MINISTRY OF EDUCATION AND RESEARCH

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE

Smart Knob in Building Automation Applications

LICENSE THESIS

Graduate: **Cosmin-Ionuț ILEA**

Supervisor: **Assoc. Prof. Eng. Roxana RUSU-BOTH, PhD**

2025

MINISTRY OF EDUCATION AND RESEARCH

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE

DEAN,
Prof. dr. ing. Vlad MUREȘAN

HEAD OF DEPARTAMENT,
Prof. dr. ing. Honoriu VALEAN

Graduate: **Cosmin-Ionuț ILEA**

Smart Knob in Building Automation Applications

1. **Project proposal:** *The thesis presents the design, implementation, and validation of a centralized smart home controller known as the "Smart Knob". This device seeks to address the issue of disjointed user control across many applications by offering a unified, straightforward interface. It will incorporate software-defined haptic feedback with sophisticated motor control and a circular display for visual indicators.*
2. **Project contents:** *Presentation page, Declaration of Authenticity, Synthesis, Chapter 1: Introduction, Chapter 2: Bibliographic study, Chapter 3: Analysis and Design, Chapter 4: Implementation and Testing, Chapter 5: Conclusions, Bibliography, Appendices.*
3. **Place of documentation:** *Technical University of Cluj-Napoca, Faculty of Automation and Computer Science.*
4. **Consultants: Assoc. Prof. Eng. Roxana RUSU-BOTH, PhD**
5. **Data of issue of the proposal:** November 1, 2024
6. **Data of delivery :** July 11, 2025

Graduate: _____

Supervisor: _____

MINISTRY OF EDUCATION AND RESEARCH

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA
FACULTY OF AUTOMATION AND COMPUTER SCIENCE

SYNTHESIS

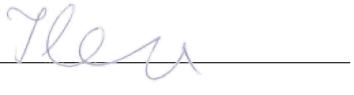
of the Diploma Thesis:

Smart Knob in Building Automation Applications

Student: **Cosmin-Ionuț ILEA**

Project supervisor: **Assoc. Prof. Eng. Roxana RUSU-BOTH, PhD**

1. **Problem definition:** The management of smart home devices is distributed among numerous applications, resulting in customer dissatisfaction. A centralized, physical controller is essential for delivering a straightforward and intuitive user experience.
2. **Application domain:** The project pertains to Building Automation and Smart Home Control. It integrates embedded technologies, advanced motor control, and IoT communication protocols to establish a tactile user interface for the management of smart home devices.
3. **Obtained results:** A prototype of a working Smart Knob was developed, which consolidates smart home management. The device employs a brushless motor with Field-Oriented Control to provide high-fidelity haptic input and has a circular display for visual cues. The system successfully interfaces with Home Assistant over MQTT to manage lighting, media, and climate management.
4. **Testing and verification:** The prototype underwent verification using comprehensive end-to-end testing scenarios. Each control mode was evaluated by manipulating the knob and verifying that the appropriate MQTT messages were transmitted, and that Home Assistant performed the associated actions. The evaluations validated the system's dependability and reactivity.
5. **Personal contributions:** The project concentrated on the design and development of the entire Smart Knob system. This encompasses hardware integration, firmware development incorporating sophisticated haptic logic, integration with Home Assistant and the bespoke mechanical design of the container.
6. **Documentation sources:** Documentation sources include academic publications, technical documentation for hardware and software libraries, and analyses of current commercial and research initiatives.

Student signature 

Project's supervisor signature: 

Contents

1. Introduction

1.1. General Context

The rapid increase in smart devices in our homes has fundamentally changed our way of living, providing more control and automation than ever before. With all these advancements, comes a new problem: the control of the devices is divided between multiple apps, voice control agents and devices. Instead of adding comfort and simplicity to the house these scattered controls only provide additional cognitive load.

One exciting direction for solving this problem comes from the use of Tangible User Interfaces (TUIs)[1]. These are systems that let the user interact with devices in a physical manner, having a response from the device in a real time allows the user to get used to the controls more quickly.

In the context of home automation this type of interaction with the house means familiar and simple gestures, like turning a knob and receiving a haptic feedback from the device feeling that the change was done.

This thesis focuses on creating the solution: a smart knob that acts as the central hub of control for a smart home. The goal is the definition of different modes that adapt to the function being controlled making the operation feel effortless, something you can learn trough instinct rather than thought. From a technical standpoint, the knob integrates haptic feedback using software defined modes using advanced motor control. Practically one well-designed knob can replace many switches, apps and other control devices from a home.

This project aims to create a tool that not only works well but also feels good and is simple to use, by combining advanced motor control techniques, home automation and human-centered design, innovating how people interact with their smart houses.

1.2. Objectives

To address the challenge of fragmented control in modern smart houses, this thesis sets out to develop a tangible and innovative solution. The work will be guided by a general goal, represented as the general objective, that will be guided by a set of specific objectives that define the functionalities and features of the project.

1.2.1. General Objective

The main objective of the work is to design, build and validate a functional smart knob that will serve as the centralized control hub for a smart house. The project aims to move beyond

screen-based and voice interaction with the smart devices in the house providing a interface that leverages TUIs making the control of various functions in the smart house feel seamless, responsive and physically interactive.

1.2.2. Specific Objectives

To achieve the general objective the following specific goals are established:

- **Design and Fabricate a Self-Contained Prototype:** Design and construct a wall mountable solution, integrating a colored circular display and a brushless motor for haptic feedback integration. Develop a custom enclosure mounted either in a surface-mounted electrical box with a standard electrical plug for power or a flush-mounted electrical box for a connection directly to the house's electrical system for a clean inside the wall look.
- **Implement a High-Fidelity Haptic Feedback:** Construct or use an advanced control technique for the BLDC motor to implement different haptic textures that will resemble the function being controlled.
- **Develop an Integrated Graphical and Tangible User Interface:** Create a seamless user interface by combining the visual cues of the circular display with the motor physical sensations. This involves designing a graphical user interface (GUI) that will display the modes and values correlated to the motor's haptic response.
- **Establish Robust Wireless Smart Home Integration:** Integrate the smart knob into a home automation ecosystem and implement wireless communication over Wi-Fi, allowing the knob to control various devices in the house (e.g. lights, temperature, media, fans).
- **Conduct System-Wide Testing and Validation:** Perform testing trough a series of practical-use case scenarios. Validating the system's reliability, responsiveness of the haptic and visual cues, proving that the system provides an intuitive and a robust integration with the home automation hub and all the equipment in the home.

1.3. Specifications

This section outlines the technical and functional specifications of the system, including requirements, constraints, and key assumptions. It defines the system's expected behavior, performance, and components without detailing implementation, which is covered in subsequent chapters.

- **Functional Requirements**
  - The smart knob must control multiple smart home devices (e.g., lights, fan, media).
  - Each rotation or press should correspond to an action or parameter change.
  - The device must generate haptic feedback contextual to the selected function.
  - Mode switching should be performed through intuitive user gestures.
- **Non-Functional Requirements**
  - The haptic feedback must feel smooth and natural, requiring high-frequency motor control.
  - The user interface must be intuitive, reducing cognitive load.

- **Technical Requirements**
  - **Hardware**
    - **Control Unit:** A dual-core microcontroller platform responsible for processing, motor control, display handling, and communication.
    - **Motor:** Brushless DC (BLDC) motor for generating configurable haptic feedback.
    - **Encoder:** High-resolution magnetic rotary encoder for precise angle tracking.
    - **Motor Driver:** A compatible 3-phase driver for controlling the BLDC motor.
    - **Display:** Circular screen used for graphical user feedback.
    - **Power Supply:** AC to DC power adapter providing 12V powering the motor.
    - **Voltage Converter:** A buck converter to step down supply voltage from 12V to 5V powering the control unit and logic components.
  - **Software**
    - The firmware is developed in C++ using the PlatformIO environment and is organized in a modular, task-driven architecture.
    - Core functionalities are distributed across well-defined software modules responsible for motor control, graphical user interface, network connectivity, and system configuration.
    - The haptic logic utilizes Field-Oriented Control (FOC) algorithms, while the graphical interface is rendered in real time using sprite-based visuals.
  - **Communication**
    - Wi-Fi communication handled by the integrated wireless module of the control unit.
    - MQTT protocol for real-time communication with the home automation broker.
- **User Interface Requirements**
  - GUI displays active mode, values, and feedback using text and progress elements.
  - TUI involves physical interaction through knob rotation and push-button, enhanced by haptic signals.
- **Constraints**
  - Requires continuous Wi-Fi and a configured MQTT broker.
  - Powered via AC supply from a wall socket or embedded wiring.
  - Number of available control modes is fixed at compile time.
  - Wireless network setup fixed at compile time.
- **Security**
  - **Network-Level Security:** The device uses WPA2/WPA3 encryption to connect securely to the local Wi-Fi network.
  - **Application-Level Security:** MQTT communication is secured using username/password credentials.
  - **Resilience:** The firmware includes watchdog and reconnect logic to prevent and recover from insecure or failed communication states.
- **Assumptions and Dependencies**
  - The smart home devices support remote control via MQTT or compatible APIs.

2. Bibliographic study

2.1. Existing Solutions and Critical Analysis

The modern smart home is ironically characterized by complexity and user frustration, despite its intended design as a seamless and intuitive environment. The two interconnected causes of this usability issue are widespread technological fragmentation and the consequent cognitive overload imposed on the user. Because the industry hasn't standardized, the end user has to do the system integration, which creates an atmosphere that goes against the convenience it claims to offer. The next parts will look closely at the current steps taken to solve this problem.

2.1.1. Commercial Solutions

Commercial solution represent the current state of the market in providing a smart home hub for the control for all the devices in the house. The most popular methods for smart home controls are voice assistants and screen-based hubs but they fail to offer a seamless usability due to their challenges and limitations.[2]

In the case of voice assistants like the Amazon Alexa or Google Assistant offer a clear benefit of hands free operation, their utility diminishes with more complex tasks. A primary weakness is their poor discoverability; users often have "limited knowledge of their full breath of capabilities" and do not know what commands are available and how to phrase them correctly.[3] Furthermore, voice is an inefficient modality for the continuous, granular control required for tasks like setting a light to a precise brightness.

Screen-based smart-home hubs such as Brilliant Smart Home Control and Google Nest Hub promise to tame "app fatigue" by concentrating all controls on a wall-mounted touchscreen.

Independent reviewers confirm that the fixed-panel format is indeed easier for guests and non-technical family members than juggling multiple phone apps or voice commands. [1]

Real-world tests reveal that these hubs expose little more than on/off or brightness sliders; accessing advanced features (e.g., Hue colour scenes or full Sonos browsing) still forces users back to each manufacturer's native app, undercutting the idea of a universal dashboard. [1] [2]

The surge of smart home gadgets has generated a need for more intuitive and tactile control interfaces that transcend voice commands and smartphone applications. A tactile

[1] Adrienne So, *Review: Brilliant Two Switch Panel*, WIRED, Nov. 20, 2018. Available at: `https://www.wired.com/review/brilliant-two-panel-switch-smart-home-controller/`. Accessed: July 1, 2025.

[2] Jennifer Pattison Tuohy, *Brilliant's New Smart Home Controller Makes It Easier to Put a Touchscreen on Your Wall*, The Verge, Aug. 22, 2023. Available at: `https://www.theverge.com/23841213/brilliant-plug-in-smart-home-control-panel-price-review`. Accessed: July 1, 2025.

smart knob, delivering haptic and visual feedback, can consolidate control and enhance user satisfaction. With custom haptic feedback, an integrated display, and broad Home Assistant compatibility, the "Smart Knob" project enters a market with many well-known commercial alternatives. The Flic Twist, Philips Hue Tap Dial, Aqara Touchscreen Dial V1, and Microsoft Surface Dial are just a few of the products researched in this analysis to put the project's innovations in perspective and determine its unique place in the modern smart home ecosystem.

The Flic Twist is a unique product in the smart controller market because it prioritizes ease of use, customization, and a small form factor that allows for flexible positioning over a wealth of built-in feedback features. Offering a simple push-button and a rotating dial that can be assigned to a variety of tasks within the Flic ecosystem is the core concept [3]. Subtle control and extensive integration with services like IFTTT, Sonos, and numerous smart lighting brands are important to the intended user, who does not need the controller to be the center of attention [4,5]. This concept is demonstrated by its small, battery-powered design, which can be attached with magnets or adhesive [3].

The Flic Twist's principal distinction from the thesis proposal resides in its feedback mechanism. The gadget provides no haptic feedback and is devoid of a screen for visual confirmation of statuses or values [3,5]. This intentional exclusion optimizes the hardware and extends battery longevity (up to two years on two AAA batteries), while imposes the cognitive burden solely on the user and the responsiveness of the managed device [3]. The rotation is fluid; but, it does not incorporate the simulated detents or end-stops that the Smart Knob provides, resulting in a less tactile experience for exact adjustments. The control operates only on functionality, depending on user memory for the dial's current configuration, which can be set for advanced dimming, scene mixing, or the selection of up to 12 distinct actions in "Selector" mode [3].

The Flic Twist needs a Flic Hub (LR or Mini) to connect the Bluetooth-enabled device to the home network [3]. This hub-based design makes it easier for the system to work with a wide range of third-party systems and devices. However, this also adds another point of failure and raises the cost of hardware. It interacts with smart home systems in a strong but indirect way. For a Home Assistant user, actions from the Twist would have to go through the Flic Hub before reaching Home Assistant. This is different from the Smart Knob, which uses MQTT to send messages directly and quickly. The Flic Twist, consequently, serves as an adaptable "remote control" for a diverse smart home, however it does not aspire to be the specialized, high-feedback interface represented by the proposal from the thesis.

The Philips Hue Tap Dial is a notable home control gadget, engineered to deliver a superior tactile experience for those immersed in the Philips Hue lighting environment. It serves as a solution for "Hue superusers" who oversee numerous lights in various rooms and need a physical controller that is more substantial and intuitive than a mere button or mobile

[3] Shortcut Labs AB, *Flic Twist*, 2024. Available at: `https://flic.io/twist`. Accessed: July 2, 2025.

[4] Jennifer Pattison Tuohy, *The Philips Hue Tap Dial is for the superusers*, The Verge, August 2022. Available at: `https://www.theverge.com/2022/8/20/23311283/philips-hue-tap-dial-switch-review`. Accessed: July 2, 2025.

[5] Matter Alpha, *Flic Twist review: the ultimate all-in-one remote, button, and dimmer*, 2023. Available at: `https://www.matteralpha.com/review/flic-twist-review`. Accessed: July 2, 2025.

application [6]. The design shows this with a heavy feel, a strong magnetic base for installation, and black or white finishes that go with the interior design [6,7]. Its main job is to give you a reliable, specialized way to choose scenes and, most importantly, control the dimming without any problems.

The Tap Dial is lauded for its superior haptic feedback, delivering a gratifying and accurate sensation during rotation [67]. This haptic input is uniform and lacks the dynamic programmability that the project's brushless motor provides. Although efficient for dimming, it cannot replicate the precise detents for fan speeds or the definitive boundaries for media management that the thesis incorporates. Moreover, the Tap Dial is devoid of a visual display, indicating that although the tactile control is commendable, the user lacks a visual indicator on the device about brightness levels or the specific functions of the four buttons associated with each zone [6]. This may result in a learning curve, particularly for visitors or in intricate multi-room configurations.

The Tap Dial's most significant strength and shortcoming lies in its profound, yet limiting, connection with the Hue ecosystem from Philips. It necessitates a Hue Bridge for operation and is totally configured using the Hue app [6]. This ensures robust reliability for managing Hue lights but restricts its inherent functionality for other devices. Although it is compatible with Apple HomeKit, its functionality is significantly restricted; the dial is inoperative within HomeKit, and the buttons are confined to single-press actions, rendering it an inadequate option as a universal HomeKit controller [67]. Signify has pledged future support for Matter, which might potentially enable management over devices from other companies; nevertheless, it currently functions as a single-ecosystem solution [6]. This categorizes it as a specialist lighting instrument, in contrast to the Smart Knob, which is fundamentally engineered as a multipurpose, multi-modal hub for a varied Home Assistant ecosystem.

The Aqara Touchscreen Dial V1 serves as a commercial equivalent to the concepts of the thesis Smart Knob project, integrating a physical dial, haptic feedback, and a dynamic display interface into one wall-mounted device. The design aims to substitute a conventional light switch, necessitating a neutral wire and expert installation, so establishing it permanently as a central control hub within a room [8]. It includes an intelligent rotary dial with haptic feedback and a 1.32-inch circular touchscreen, which can be personalized with various themes and layouts to present information such as time, weather, and inside temperature [8]. This positions it as an ambitious, comprehensive controller designed for people seeking a sophisticated and visually engaging smart home interface.

The integration of haptic and visual feedback in the Aqara Dial V1 delivers an extensive user experience. The dial provides intuitive control with haptic feedback, while the touch-

[6]Jennifer Pattison Tuohy, *The Philips Hue Tap Dial is for the superusers*, The Verge, August 2022. Available at: `https://www.theverge.com/2022/8/20/23311283/philips-hue-tap-dial-switch-review`. Accessed: July 2, 2025.

[7]Michael Ansaldo, *Philips Hue unwraps a portable Go table lamp and a revamped Tap switch*, TechHive, June 2022. Available at: `https://www.techhive.com/article/792270/philips-hue-unwraps-portable-go-table-lamp-revamped-tap-switch.html`. Accessed: July 2, 2025.

[8]Aqara, *Aqara Touchscreen Dial V1*, 2024. Available at: `https://eu.aqara.com/products/aqara-touchscreen-dial-v1`. Accessed: July 2, 2025.

screen distinctly displays the manipulated elements, removing the uncertainty associated with screenless devices [8]. It is capable of managing both two hard-wired light loads and a diverse range of wireless Aqara and third-party devices [8] . Moreover, its functionality is augmented by integrated sensors for temperature, humidity, and occupancy detection, enabling it to serve as both a human controller and an automation initiator [8]. The screen can autonomously activate upon detecting an approaching individual [9]. This degree of integrated sensing and display capacity establishes a significant standard for functionality in a commercial product.

The Aqara Dial V1 exhibits considerable versatility in terms of integration. It functions via Wi-Fi and may immediately interface with third-party ecosystems such as Apple Home, Alexa, and Google Home, frequently without need a specialized Aqara hub [8]. Its compatibility with Zigbee enables it to function as a repeater, hence enhancing the mesh network [8]. Although its dial functionality is most effective within the Aqara Home app, its interoperability with major platforms, including anticipated Matter support through a bridge, renders it a widely interoperable device [8]. This serves as a compelling illustration of a commercial endeavor to develop a genuine smart home hub in a knob configuration, aligning with the basic goal of the Smart Knob, yet executed through a mass-market, in-wall switch model.

Another device similar to the thesis subject is the Microsoft Surface Dial that exemplifies enhanced haptic feedback and context-aware control, shifting the emphasis from home automation to creative productivity. It was not intended as a smart home hub, but rather as a "novel instrument for the creative process," meant to complement a mouse and pen by offering a third input method for digital artists, designers, and video editors [10,11]. The physical design is of high quality, crafted from metal, featuring a non-slip base that enables placement on a desk or directly on the screens of compatible Surface devices [10].

The primary novelty of the Surface Dial is its seamless interaction with the Windows operating system and select creative applications. The haptic feedback delivers beneficial vibrations that align with operations in software, establishing a palpable link to digital instruments. Upon activation, it displays a radial menu on the computer screen, with options that vary according to the active application [10,11]. For instance, in a sketching application, it may regulate brush size and opacity, in Spotify, it governs volume, and in Windows Maps, it might facilitate zooming and rotating the 3D view. The primary characteristic of this modality is its context-awareness. Visual feedback is completely transferred to the primary display, representing a fundamentally distinct methodology compared to the integrated screens of the Smart Knob or the Aqara Dial [10].

The Surface Dial, albeit a highly inventive peripheral, is predominantly limited to the Windows ecosystem and a select array of supported applications [11]. Native compatibility for smart home platforms such as Home Assistant, Alexa, or Google Home is absent. Although technically proficient users have devised methods for creating bespoke integrations, this neces-

[9]HomeKit News and Reviews, *Aqara MagicSwitch S1E Review*, 2023. Available at: `https://homekitnews.com/2023/01/22/aqara-magic-switch-s1e-review/`. Accessed: July 2, 2025.

[10]Microsoft, *Surface Dial*, 2024. Available at: `https://www.microsoft.com/en-us/d/surface-dial/925r551sktgn`. Accessed: July 2, 2025.

[11]Tom Warren, *Microsoft Surface Dial hands-on*, The Verge, November 2016. Available at: `https://www.theverge.com/2016/11/10/13586658/microsoft-surface-dial-hands-on`. Accessed: July 2, 2025.

sitates considerable programming work and is not an intended application. Consequently, the Surface Dial functions not as a rival in the smart home sector, but as a source of inspiration. It illustrates the efficacy of meticulously implemented haptics and comprehensive software integration in rendering digital interactions more tactile and intuitive, offering significant insights into the possibilities for sophisticated, context-sensitive haptic design applicable to a smart home environment.

The examination of existing commercial solutions, as presented in **Table 2.1**, indicates a market characterized by considerable trade-offs for each device. Currently, no device integrates genuine platform openness, distinct visual feedback, and a high-fidelity tactile interface. To facilitate hands-free or visual convenience, voice and screen-based hubs compromise accuracy and tactile engagement.

Table 2.1: Comparative Analysis of Smart Control Solutions

Solution	**Primary Control**	**Visual Feedback**	**Haptic Feedback**	**Platform Openness**	**Key Limitation**
Voice Assistants	Voice Commands	Auditory/ Smart Display	None	High (via skills)	Poor discoverability; no precision control
Screen-based Hubs	Touchscreen	Integrated Screen	None	Moderate (often shallow)	Lacks tactile feel; limited deep control
Flic Twist	Dial & Button	None	None	High (via Hub)	No feedback; relies on user memory
Philips Hue Tap Dial	Dial & Buttons	None	Standard Vibration	Low (Hue-centric)	Closed ecosystem; no visual feedback
Aqara Touchscreen Dial V1	Dial & Touchscreen	Integrated Screen	Vibration & Sound	High (Wi-Fi/Zigbee)	Fixed in-wall unit; basic haptics
Microsoft Surface Dial	Dial & Press	On-screen (PC)	Advanced Vibration	Very Low (Windows-centric)	Not a smart home device
Smart Knob (Thesis)	**Dial & Press**	**Integrated Screen**	**Motor-driven (Torque)**	**Very High (MQTT/HA)**	**DIY complexity**

Each dedicated knob controller is constrained by a fundamental trade-off, such as the closed ecosystem of the **Philips Hue Tap Dial**, the absence of feedback in the **Flic Twist**, or the basic vibration-based haptics in the fixed design of the **Aqara Touchscreen Dial V1**. Although not a smart home device, the **Microsoft Surface Dial** represents a significant standard for the capabilities of high-quality, context-aware haptics.

This environment presents a distinct potential for a technology that integrates these varied strengths. The Smart Knob initiative, as detailed in this thesis, aims to address this gap. The objective is to provide an unparalleled user experience by merging advanced motor-driven haptics with a distinct visual display and the necessary platform openness for ecosystems such as Home Assistant—a combination presently absent in the commercial sector.

2.1.2. Industrial & Research Prototypes

In addition to the business world, academic research and industrial prototyping look into more specific or new uses for knob-based interfaces. This body of work often focuses on new ways of interacting, advanced haptic representations, or solutions for specific, complex areas instead of the problems of cost and simplicity that come with mass-market products. These prototypes help us understand the basic concepts and technological possibilities that will shape the design of future tactile interfaces.

The "Internet of Tangible Things (IoTT)" is a basic idea that says the most common smartphone and web interfaces for IoT devices make it hard for people to connect with the real world [4]. Angelini et al. say that using people's natural abilities can help stop this trend. Their study outlines important features for physical interfaces, such as giving meaningful physical representations of digital states (T1), improving spatial and collaborative interactions (T4), and encouraging reflection (T7). This theory directly addresses the usability problem that commercial solutions create, in which the user's mental understanding of how a device works is often unclear [1]. The IoTT paradigm shows how important persistency (T3) is. This means that a physical interface can keep its state and keep working even if there are problems with power or communication. This is a common weakness in only digital IoT controllers. It also makes a distinction between "Ready-to-Hand" interactions, which are fluid and part of everyday life, and "Present-at-Hand" interactions, which are deliberate and encourage thought. This sets up a theoretical framework for making haptic experiences that are more nuanced and sensitive to context [4].

From a technological point of view, research has looked into different ways to make haptic input. Giraud, Amberg, and Lemaire-Semail made a haptic knob using the "squeeze film effect." In this effect, ultrasonic vibrations at a high frequency (about 38.6 kHz in their prototype) create an air cushion that controls the friction against the user's finger [5]. Their careful design process included user research to find the best ring size (38mm in diameter) and finite element analysis to find a vibration mode that made the effect stronger at the user's finger location. One of the biggest technical problems they had to solve was making a custom four-point force sensor to find out where the finger was pointing by using trigonometric calculations to figure out the forces. They made a PI control loop to keep the vibration amplitude stable, which would have changed the haptic feel if the user's finger pressure naturally dampened it [5].

Ding et al. talk about a cheap haptic knob for programming industrial robots that works better with the technology used in this thesis [6]. Their prototype, which costs 100 EUR in hardware, uses a Field-Oriented Controlled (FOC) Brushless DC (BLDC) motor to give force feedback. This shows that this cheap technology can be used for high-precision tasks. One important new thing about their work is that their knob doesn't have a built-in force/torque

sensor. Instead, it uses the impedance controller of the industrial robot. When the robot's end-effector hits something, the sensors on the robot detect the contact force and send it to the knob's motor as a resistive torque. A control interface uses a linear mapping coefficient, K^{Force}, to turn the robot's contact force (measured in Newtons) into knob torque (measured in Newton-meters). This makes it easy for the operator to connect with the machine in a direct, intuitive, and cost-effective way. This is helpful for complicated tasks like installing a car fuse box, where a precise force profile is necessary [6].

The use of haptic knobs has grown to include the area of information visualization. Zhang et al. look into how haptic engagement might help people understand large, complicated datasets in order to reduce cognitive stress and visual strain [7]. They made a full "design space" and a prototype knob that used a BLDC motor to turn data attributes into physical forces. Their technology uses a lot of different haptic feedback patterns. For example, "Barrier" makes the user feel like they're hitting a solid surface to let them know something is wrong with a scatterplot, and "Mapping" changes the knob's rotational resistance based on the data values being looked at. Their user tests showed that using haptic feedback made visualization tasks more accurate than just using visual feedback. This study shows that a haptic knob is more than just a controller; it is an exploratory tool that makes abstract data real, which opens up new uses for it in a smart home, from basic device management to showing ambient data (like visualizing energy use) [7].

On the other hand, research has looked into simple, single-purpose IoT controls. Garg et al. made a "Simple Gas Knob" to make LPG stoves safer [8]. This cheap device uses a basic Hall Effect sensor to figure out whether the knob is on or off and sends this information to a user's smartphone over an ESP8266 Wi-Fi module using the MQTT protocol. It doesn't have haptic feedback, but it shows the "Internet of Things" in its most basic form: adding connectivity to a simple object to meet a specific, practical need [8].

In the end, even though it's not a knob, relevant research on haptic feedback for the Metaverse shows how important the basic communication structure is. Yang et al. made "FingerTac," a wearable haptic glove system that gives vibrotactile feedback for interacting with objects in augmented reality [9]. One important part of their study was setting up network-level Smart Queue Management (SQM) on a custom OpenWrt router to give the haptic data stream priority. This makes sure that tactile sensations are sent with as little delay as possible, which is necessary for a believable and immersive user experience. This idea is important for any haptic device that is connected to a network, like the Smart Knob, because the quality of feedback depends on how fast and stable the data link is between the device and the system it controls [9].

XeelTech's HAPTICORE technology is a big step forward for advanced haptic interfaces. It offers a unique and specialized alternative to motor-based systems [12]. The HAPTICORE actuator is not a regular motor; it is a unique design that uses electromagnetism and special materials to create programmable haptic feedback that can only be controlled by software. This technology lets you change the amount of friction between internal parts right away,

[12]XeelTech GmbH, *HAPTICORE Smart Knob – Combining a haptic knob and a display*, 2024. Available at: `https://www.xeeltech.com/smart-knob/`. Accessed: July 3, 2025.

which creates a wide range of complex haptic effects. The HAPTICORE platform is built to help people make better Human-Machine Interfaces (HMIs) and has a library of complex, customizable haptic patterns. This includes basic detents (Ticks) and continuous resistance (Torque), as well as more complicated effects like hard stops (Barriers) and full rotational locks (Lock). All of these can be fine-tuned with software settings, such as current and speed factors. XeelTech provides integration diagrams for combining their actuator with a stationary core display. This creates a "Smart Knob" where the outer ring spins around a fixed screen. This industrial solution is a perfect example of a carefully designed, market-ready haptic module that can be used in automotive controls, high-end appliances, and industrial machinery, among other things, where a superior, reliable, and highly customizable tactile experience is needed [12]. The exploration of industrial and research prototypes reveals a landscape of haptic interfaces that move beyond commercial constraints to investigate novel technologies and specialized applications. Theoretical frameworks like the Internet of Tangible Things (IoTT) argue for physical interfaces that are more intuitive and context-aware than their screen-based counterparts.

Technologically, these prototypes showcase diverse methods for generating feedback. Some, like the knob by Giraud et al., use unconventional physics such as the "squeeze film effect" to modulate friction, while high-end industrial solutions like the XeelTech HAPTICORE use patented electromagnetic actuators to create a wide range of software-defined haptic effects for premium HMIs.

Research by Ding et al. and Zhang et al. corroborates the efficacy of cost-effective BLDC motors for sophisticated applications. Ding's research presents a control mechanism for industrial robot programming that ingeniously utilizes force input from the robot's sensors, whereas Zhang's study employs a comparable mechanism for data visualization, converting abstract data into physical forces to alleviate cognitive burden. These examples are juxtaposed with more simplistic solutions, such as the gas knob by Garg et al., which employs fundamental sensors and MQTT for a singular safety function devoid of haptic feedback. Research on networked haptics, exemplified by the FingerTac glove, highlights the essential need for a low-latency communication infrastructure to guarantee that haptic input is both responsive and believable.

2.2. Motor Control Methods for Haptic Knob Applications

This bibliographic study offers a comprehensive analysis of several motor types and control approaches pertinent to the advancement of high-fidelity haptic knobs. The emphasis is on Brushless DC (BLDC) motors because of their performance attributes, accompanied by a comprehensive examination of control methodologies, from fundamental commutation to sophisticated vector control, for delivering accurate force feedback.

2.2.1. Overview of Motor Types

In precise applications such as haptic knobs, the selection of the motor is essential. The predominant alternatives comprise Brushless DC (BLDC) motors, Permanent Magnet Synchronous Motors (PMSM), and stepper motors.

BLDC and PMSM Motors are used for haptic applications owing to their superior efficiency, exceptional speed-torque characteristics, high reliability, extended operational lifespan, and silent operation [10]. These motors forgo the mechanical brushes and commutator present in conventional brushed DC motors, thus mitigating problems related to wear and tear, sparking, and substantial electromagnetic interference (EMI) [11]. A BLDC motor is a category of permanent magnet synchronous machine, and the phrases are frequently used synonymously [12]. BLDC motors are characterized by a trapezoidal back-EMF waveform and are conventionally operated using six-step commutation, while PMSMs exhibit a sinusoidal back-EMF and are powered by sinusoidal currents [12, 13]. The elevated torque-to-size ratio of BLDC/PMSM motors renders them optimal for applications where spatial and weight constraints are paramount [12]. Their efficacy as haptic actuators is well-documented in studies with programmable force feedback knobs.

Stepper Motors are another option, particularly in low-power applications. They move in discrete steps, which allows for precise open-loop position control. However, they can suffer from lower efficiency, limited speed, and potential resonance issues, which can be detrimental to creating smooth haptic feedback. Low-voltage drives, including stepper motors, have a predominant share of low-power applications [14].

Servo Motors are not a distinct type of motor but rather a system consisting of a motor (often a BLDC or PMSM), a position sensor (like an encoder), and a sophisticated controller that enables precise control of position, speed, and torque [15]. For high-performance haptics, a BLDC or PMSM is typically operated as a servo motor.

For haptic knob applications that require smooth torque, rapid dynamic response, and high precision, the **BLDC/PMSM motor is the superior choice** when combined with an advanced control strategy.

2.2.2. Control Techniques for BLDC Motors

Trapezoidal and Sinusoidal Commutation

Trapezoidal commutation is the standard technique for operating BLDC motors. This six-step procedure involves passing current through two of the three motor phases simultaneously, with the sequence dictated by Hall-effect sensors that ascertain the rotor's location [16, 13]. This method produces a quasi-square wave current waveform [16].

- **Benefits**: The control logic is straightforward to build and necessitates less sophisticated processing [16]. The inverter efficiency may exceed that of sinusoidal approaches due to reduced overall switching losses, as only two transistors are activated simultaneously [16].
- **Constraints**: The principal disadvantage is considerable torque ripple, which arises at

every 60-degree commutation interval [13]. This ripple is detrimental for high-precision applications and can hinder the attainment of smooth haptic feedback.

Sinusoidal commutation is the conventional technique for operating permanent magnet synchronous motors (PMSMs) and can also be utilized in brushless DC (BLDC) motors to enhance performance. This approach involves continually energizing all three phases with sinusoidal currents that are phase-shifted by 120 degrees [16]. This generates a continuously revolving stator magnetic field that engages with the rotor's magnetic field to yield consistent thrust.

- **Benefits**: This methodology yields considerably smoother rotation and markedly diminishes torque ripple in comparison to the trapezoidal method [13]. This results in reduced noise and more precision, which is particularly advantageous for haptics. Simulation results indicate that sinusoidal control yields expedited and more stable speed and position tracking [15].
- **Limitations**: This approach is more intricate and computationally demanding, as it necessitates the generation and regulation of exact sinusoidal waveforms. The inverter exhibits marginally reduced efficiency owing to elevated switching losses, as all three phases are perpetually active [16].

Field Oriented Control (FOC)

A complex control technique called Field Oriented Control (FOC), also known as vector control, makes it possible to regulate a BLDC/PMSM motor and achieve dynamic performance comparable to that of an independently excited DC machine. The basic idea is to decouple the control of the motor's torque and flux. This is a well-known technique for managing haptic knobs, as it makes it possible to directly and instantly regulate torque, which is necessary for producing high-fidelity haptic effects.

Principles and Transformations FOC accomplishes decoupling by converting the stator currents from the stationary three-phase reference frame (abc) into a two-axis rotating reference frame (dq) that aligns with the flux vector of the rotor. The torque- and flux-generating components of the current are rendered autonomous by this transformation. Two fundamental mathematical transformations are used in the method:

1. **Clarke Transform:** In a stationary reference frame, this transformation converts the three-phase time-domain currents (I_a, I_b, I_c) into two orthogonal components (I_α, I_β).
2. **Park Transform:** The two stationary components (I_α, I_β) are transformed into the dq reference frame by the Park Transform, and this frame rotates in tandem with the rotor flux. The position of the rotor determines the angle for this rotation. Within this rotating framework, alternating currents transform into direct current values in the steady state, which facilitates their regulation.

Current Control In the dq frame, while the quadrature-axis current (I_q) component is perpendicular to the flux and governs the motor's torque, the direct-axis current (I_d) component is aligned with the rotor flux and regulates the strength of the rotor magnetic field. Since the rotor flux in a permanent magnet motor is constant, keeping the flux-producing current (I_d) at zero is usually the main goal. This method maximizes the torque-per-ampere ratio and boosts overall

efficiency by ensuring that the full stator current is used for torque generation. The torque is directly proportional to the I_q current. Two independent PI controllers are used to regulate I_d and I_q to their specified reference values (I_{d_ref} and I_{q_ref}). In order to generate the PWM signals for the inverter, the controllers produce voltage commands (V_d, V_q), which are subsequently converted back to the three-phase system using inverse Park and Clarke transformations. The internal current loop facilitates accurate, dynamic torque regulation and is essential to Field Oriented Control (FOC).

Sensorless Control

Sensorless control seeks to obviate the necessity for mechanical position sensors such as Hall sensors or encoders, thereby diminishing system cost, size, and complexity while enhancing dependability [12, 10]. The rotor position is inferred from the motor's electrical impulses.

Back-EMF Methods: The predominant sensorless method for BLDC motors relies on the detection of the motor's back-electromotive force (back-EMF) [12]. In the conventional six-step trapezoidal commutation, one motor phase remains unenergized (floating), with the voltage on this phase being proportional to the back-EMF [12]. The **Zero-Crossing Detection (ZCD)** technique observes this voltage and identifies the moment it intersects zero. This timing information, generally exhibiting a 30-degree electrical phase shift, is utilized to ascertain the appropriate instant for the subsequent commutation [12].

- **Pros:** The ZCD approach is straightforward and economical to execute [12].
- **Cons**: The primary limitation is its incapacity to operate at minimal speeds or when stationary, as the back-EMF correlates with speed and is insufficiently strong for detection [12]. This requires an independent open-loop initiation process to commence motor rotation prior to the sensorless algorithm assuming control [12]. Moreover, noise generated by PWM switching can distort the back-EMF signal, necessitating filtering that may create phase delays and restrict high-speed performance [12].

Observer-Based Approaches Observer-based strategies can be employed for enhanced control and improved low-speed performance. These strategies utilize a mathematical model of the motor operating concurrently with the actual motor [12]. An error signal is generated by comparing the model's estimated outputs (e.g., current) with the actual outputs from the motor [12]. This mistake serves as corrective feedback for the model, compelling the estimated states (such as rotor position and speed) to align with the real states [12].

- **Pros:** Observers can deliver continuous and precise position estimations, even at minimal velocities, and are typically more resilient to parameter fluctuations [12].
- **Cons**: These algorithms are resource-intensive and generally necessitate a Digital Signal Processor (DSP) for real-time execution [12]. Typical instances comprise the **Luenberger Observer**, **Sliding-Mode Observer (SMO)**, and the **Extended Kalman Filter (EKF)** [12].

2.2.3. Position and Torque Control Algorithms

PID Control

The Proportional-Integral-Derivative (PID) controller is a common feedback control method that is used in many industrial processes because it is simple, reliable, and works well [17, 18]. The controller changes a control output to try to reduce the difference between a measured process variable and a desired setpoint. The result is the sum of three parts:

- The **proportional (P)** phrase gives an output that is proportionate to the current error.
- The **Integral (I)** phrase takes into account mistakes that have already happened by combining them across time. This phrase is very important for getting rid of steady-state defects.
- The **Derivative (D)** term uses the current rate of change to estimate future error. This helps to slow down the system's response and keep it from overshooting.

The overall control function is given by the equation [18]:

$$C(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{2.1}$$

where $e(t)$ is the error, and K_p, K_i, and K_d are the gains.

A PID controller is optimal for the outermost position control loop in a haptic knob. The error is the discrepancy between the intended knob position and the actual measured position. The output of this PID controller often establishes the speed reference for a cascaded speed control loop [15].

Adjusting the PID gains is essential for optimal performance but may be difficult due to system nonlinearities and variations in parameters [17, 18]. Although manual approaches are available, sophisticated strategies like employing a genetic algorithm to identify the ideal gain coefficients (K_p, K_i) have been suggested to ensure resilient performance under varying operating conditions [15].

Cascaded Architectures

A cascaded control architecture is a conventional and extremely efficient framework for intricate motion control systems. A haptic knob generally entails a hierarchy of interrelated control loops. Position → Speed → Torque (Current) [15].

The structure operates in the following manner:

1. The outermost Position Loop, typically a P or PID controller, computes the discrepancy between the target and actual knob position. The output is a **speed command** utilized as the reference for the speed loop [15].
2. The intermediate **Speed Loop**, generally a PI controller, juxtaposes the reference speed with the actual motor speed [13, 14]. The output is a torque command (I_{q_ref} in FOC) provided as the reference to the innermost loop [13, 14].
3. The innermost **Current (Torque) Loop** (including PI controllers in FOC) governs the motor phase currents to provide the requisite torque [13, 14]. This is the most rapid loop

in the system.

The justification for this architecture is bandwidth segregation. Every inner loop must exhibit a markedly higher speed than the enclosing loop. The rapid current loop guarantees that the motor reacts swiftly to torque inputs, rendering it an optimal torque source for the speed loop [14]. The response of the speed loop is far swifter than the mechanical adjustments in the position loop. This decoupling enables the independent design and tuning of the loops, significantly streamlining the control architecture and maintaining stability.

2.2.4. Summary

For high-performance haptic knob applications requiring accurate and dynamic control of both torque and position, **Field Oriented Control (FOC) is the optimal method** among those evaluated. Field-Oriented Control (FOC) facilitates independent regulation of torque and flux, allowing for immediate torque modifications essential for delivering intricate haptic feedback, which is a notable benefit compared to the torque ripple-susceptible trapezoidal control [13, 11, 19].

The optimal control architecture is a cascaded structure with an outside position loop, a middle speed loop, and a rapid inner current (torque) loop. This modular design is resilient and facilitates adjustment. A **PID controller** is a widely recognized option for the position loop [17, 18].

Field-Oriented Control (FOC) simplifies control to the management of direct current (DC) quantities by converting stator currents into the dq rotating reference frame using the Clarke and Park transformations.[13] [13]. The motor must be a **BLDC or PMSM** to utilize their superior torque density and efficiency [10].

Ultimately, although sensorless control presents a means to diminish expenses and intricacy [12], high-precision haptic feedback generally derives advantages from the direct and unequivocal position data supplied by a high-resolution **encoder** [15]. The selection between sensored and sensorless control involves a compromise between optimal performance and the associated costs and complexities of the system.

2.3. Communication Protocols

Efficient communication is crucial for the functionality of any sophisticated embedded device, such as a haptic knob. This include exterior communication with other devices or networks as well as internal communication among microcontrollers and peripherals. This section examines the protocols pertinent to these duties.

[13] MathWorks, *Clarke and Park Transforms*, n.d. Available at: `https://www.mathworks.com/discovery/clarke-and-park-transforms.html`. Accessed: July 6, 2025.

2.3.1. External communication protocols

Wi-Fi

Wi-Fi is a technology for wireless networks founded on the IEEE 802.11 standards, delivering high-bandwidth communication for devices within a local area network (LAN) [20]. Within the realm of Internet of Things (IoT) and especially home automation, Wi-Fi is a prevalent option owing to its omnipresence and the extensive availability of access points (APs) [21]. Embedded systems, including those utilizing an ESP32 microcontroller, can function in various Wi-Fi modes:

- **Station (STA) Mode:** The device connects to a pre-existing Wi-Fi network created by an access point [20]. This is the predominant method for an end-device, such as a haptic knob, to connect to a local network and the internet.
- **Access Point (AP) Mode:** The device establishes an independent Wi-Fi network, enabling other devices to connect directly to it [20].
- **AP+STA Mode:** The device can concurrently establish its own network while connecting to another access point, functioning as both a hub and a client [20].

The two main levels of the Wi-Fi architecture are the Medium Access Control (MAC) layer, which handles network access and data framing, and the Physical layer (PHY) [20]. Although its elevated data rate is beneficial, Wi-Fi's greater power consumption relative to protocols such as Zigbee or Bluetooth renders it more appropriate for devices that are not exclusively battery-operated.

MQTT Protocol

Message Queuing Telemetry Transport (MQTT) is a lightweight publish/subscribe messaging protocol designed for networks with low bandwidth, high latency, or unreliability, as well as restricted devices [14]. For IoT connectivity, it has become a standard [21]. MQTT's publish/subscribe design isolates clients from one another, unlike the client-server model, which requires a client to poll a server for information.

MQTT's core components are [14]:

- **Clients:** Any device, ranging from a sensor to an application, capable of connecting to the broker. Clients may either disseminate messages or subscribe to receive them.
- **Broker:** The central server that receives messages from publishing clients and disseminates them to subscribed clients according to the relevant topic. It oversees subscription management and guarantees message delivery.
- **Topics:** The broker uses a UTF-8 text called a topic to filter communications for every client. Clients disseminate messages to designated subjects (e.g., home/livingroom/ temperature), while other clients obtain such messages by subscribing to the corresponding topic.

This design is incredibly flexible and scalable. Many sensors can send their data to a central broker in a smart home system, and different actuators or user apps can subscribe to this data

[14] HiveMQ Team, *MQTT Tutorial: An Easy Guide to Getting Started with MQTT*, 2025. Available at: `https://www.hivemq.com/blog/how-to-get-started-with-mqtt/`. Accessed: July 6, 2025.

to perform actions without needing to know the sensors themselves [21].Multiple Quality of Service (QoS) levels are available using MQTT to guarantee message delivery dependability. These levels range from QoS 0 (at most once) to QoS 2 (exactly once).[14].

2.3.2. Internal Communication Protocols

Although Wi-Fi and MQTT facilitate external connections, internal communication between a central microcontroller and its peripherals, like sensors or motor drivers, requires a particular class of protocols that are made for high-speed, short-distance data transfer on a printed circuit board (PCB).

Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol utilized in embedded systems. It operates within a master-slave architecture, in which one or more slave devices are controlled by a single master device. Because of its ease of use and high throughput, SPI is the most widely used standard for this kind of communication [22].

Four primary logic signals are used in the interface [22]:

- **SCLK (Serial Clock):** The master-driven clock signal that synchronizes data transmission.
- **MOSI (Master Out Slave In) / SPISIMO:** The data line utilized for transmitting information from the master to the slave.
- **MISO (Master In Slave Out) / SPISOMI:** This data line transmits information from the slave to the master.
- **SS (Slave Select) / SPIENA:** The signal that the master uses to identify which slave device is to be used for communication.

Full-duplex operation, which allows simultaneous data transmission and reception, is made possible by SPI's use of separate lines for data transmission and reception. Analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and specialized motor driver integrated circuits (ICs) are among the peripherals that can be connected to it due to its high-speed, synchronous characteristics.

2.3.3. Summary

A haptic knob, as a sophisticated device, necessitates the integration of various protocols.

SPI serves as the essential bus for internal, high-speed communication on the device's printed circuit board. It links the primary microcontroller to critical peripherals requiring swift data transmission, such as the motor driver, high-resolution position sensors, and graphical displays. This guarantees that low-level control signals, sensor data, and display updates are performed with minimal latency.

The device will probably utilize Wi-Fi for external connectivity to link with a local network. Alongside the Wi-Fi connection, MQTT offers a versatile and efficient messaging framework for transmitting status information (e.g., current knob position) and receiving commands (e.g., haptic effect parameters) from a host computer or other IoT devices within the

ecosystem. This stratified methodology optimizes the implementation of each protocol: SPI for internal velocity and precision, and Wi-Fi with MQTT for external communication and interoperability.

3. Analysis and Design

3.1. Theoretical foundation for the haptic feedback system

3.1.1. Mathematical Model of BLDC Motor

Brushless DC (BLDC) motors are extensively utilized in precision control applications owing to their superior efficiency, torque density, and reliability. A BLDC motor has a rotor embedded with permanent magnets and a stator featuring dispersed windings, necessitating electronic commutation contingent upon rotor position.

A three-phase BLDC motor's mathematical model is developed using the standard voltage equations for each phase.

$$V_a = Ri_a + L\frac{di_a}{dt} + e_a \tag{3.1}$$

$$V_b = Ri_b + L\frac{di_b}{dt} + e_b \tag{3.2}$$

$$V_c = Ri_c + L\frac{di_c}{dt} + e_c \tag{3.3}$$

Where:

- V_x denotes the phase voltages.
- i_x represents the phase currents.
- R and L signify the stator resistance and inductance, respectively.
- e_x refers to the back-EMF voltages, which are dependent on rotor position.

Continuous estimation or measurement of rotor position is essential for successful torque control, commonly achieved by Hall effect sensors or magnetic encoders. The back electromotive force (EMF) and the stator current's quadrature component are directly proportional to the electromagnetic torque, which may be written as follows:

$$T_e = \frac{3}{2} \cdot \frac{P}{2} \cdot \left(\psi \cdot i_q\right) \tag{3.4}$$

Where:

- T_e denotes the electromagnetic torque.
- P denotes the number of poles.
- ψ represents the flux linkage.
- i_q represents the quadrature axis current component.

3.1.2. Angle Measurement

Accurate angular position sensing is crucial in high-performance motor control systems, particularly those utilizing Field-Oriented Control (FOC), since the control strategy depends on the rotor's real-time position. Among the various methods, magnetic angle detection employing the Hall effect provides a dependable and non-contact method for ascertaining absolute position.

This technique employs a diametrically magnetized rotor magnet that generates a revolving magnetic field as the shaft turns. A stationary sensor, located in proximity, consists of two orthogonal Hall components that measure the magnetic flux density in the horizontal plane, designated as B_x and B_y. These components represent the projection of the magnetic field onto the sensor's axis.

As the magnet rotates, B_x and B_y fluctuate sinusoidally with a phase difference of 90 degrees, resulting in a circular vector trajectory in the B_x–B_y plane. This attribute facilitates the utilization of trigonometric functions to reconstruct the angle of rotation. The angle θ is determined using the two-argument arctangent function:

$$\theta = \arctan\left(\frac{B_y}{B_x}\right) \tag{3.5}$$

This expression yields a continuous and definitive angular location throughout the whole 360° spectrum, regardless of the direction of rotation. The accuracy of this measurement depends on the linearity of the Hall sensors, the consistency of the magnetic field, and the spatial alignment between the sensor and the magnet.

This method, unlike incremental encoders, does not rely on the detection of transitions or edges, hence avoiding quantization jitter and ambiguity. Additionally, it provides excellent resolution and repeatability, making it suitable for precise feedback in real-time control systems.

The theoretical basis of this sensing technology, centered on vector field analysis and trigonometric reconstruction, efficiently meets the requirements of modern sensor-based control systems in robotics, automation, and embedded mechatronics.

3.1.3. Field-Oriented Control Strategy Incorporating Position Loop

Field-Oriented Control (FOC) is a vector control technique that facilitates precise decoupled management of torque and flux in alternating current machines, such as Permanent Magnet Synchronous Motors (PMSM) or Brushless Direct Current Motors (BLDC) . It accomplishes this by mapping the stator currents onto a revolving reference frame that is aligned with the rotor's magnetic field.

Clarke and Park Transformations

The control procedure begins with the Clarke transformation, which converts three-phase currents into two orthogonal components inside a stationary reference frame:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{3.6}$$

The Park transformation subsequently turns the stationary frame to align with the rotor's magnetic field, resulting in the direct (i_d) and quadrature (i_q) current components:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{3.7}$$

Space Vector PWM

A three-phase inverter's six operational switching states are depicted in the space vector diagram. Numbered 1 through 6, each vector is positioned 60 degrees apart within the stationary $\alpha\beta$ reference frame and represents a unique combination of inverter switch states. The six equivalent sectors used for space vector modulation are indicated by these vectors.

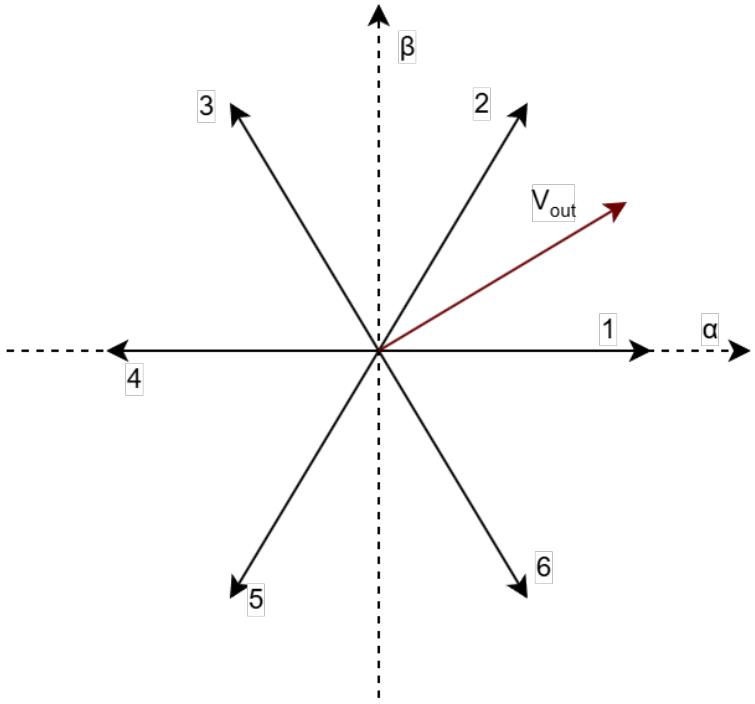


Figure 3.1: Space vector diagram showing Sector 1's output voltage vector V_{out} and switching vectors

The arrow V_{out} in Figure 3.1 represents the desired output voltage vector. In this case, V_{out} is located between vectors 1 and 2 in **Sector 1**. The SVPWM algorithm uses the following to estimate this vector for every PWM cycle:

- Active vector 1 for duration T_1,
- Active vector 2 for duration T_2,
- A null vector for the remaining duration T_0.

This results in an average voltage vector that approximates the desired V_{out} as shown in Equation 3.8:

$$\vec{V}_{out} = \frac{1}{T_s}\left(T_1 \cdot \vec{V}_1 + T_2 \cdot \vec{V}_2\right) \tag{3.8}$$

where T_s is the total duration of one PWM cycle, and $T_0 = T_s - T_1 - T_2$ is the time during which a null (zero) vector is applied.

The inverter creates an average voltage vector that closely resembles V_{out} by combining these vectors in a time-weighted manner. With each PWM cycle, this process is repeated, producing a continuously spinning voltage vector that ensures efficient torque and flux control in the motor.

SVPWM is a better technique than traditional sinusoidal PWM because of its ability to minimize harmonic distortion and maximize DC bus utilization. A higher-level torque control loop can be introduced by implementing the inner current control loop using FOC and SVPWM. In order to accurately regulate the motor's torque output in accordance with dynamic system demands, this loop operates on a slower timeframe and provides the FOC block with the i_q^* reference.

Torque Control Loop

The torque control loop constitutes an essential component of the comprehensive cascaded control framework in Field-Oriented Control (FOC). Situated above the inner current control loop, it is tasked with generating the quadrature-axis current reference $i_{q_{ref}}$, which directly regulates the electromagnetic torque generated by the motor.

As depicted in Figure 3.2, the control system quantifies all three motor phase currents (i_a, i_b, i_c). The Clarke transformation block receives these inputs to calculate the two orthogonal components i_α and i_β in the stationary reference frame. This comprehensive three-phase measurement enables enhanced precision in current estimation relative to inferring the third phase from the other two.

Thereafter, the Park transformation translates the stationary frame currents into the rotating d-q reference frame, yielding i_d and i_q. These components denote the currents that generate flux and torque, respectively.

In most applications, the direct-axis current i_d is kept at zero to maximize torque per ampere, hence assuring efficient current utilization and preventing superfluous magnetizing flux. The quadrature-axis current i_q is utilized to control the electromagnetic torque.

Two independent PI controllers are utilized to accomplish this:

- The i_d controller manages the current that generates flux and is generally supplied with a reference value of zero.

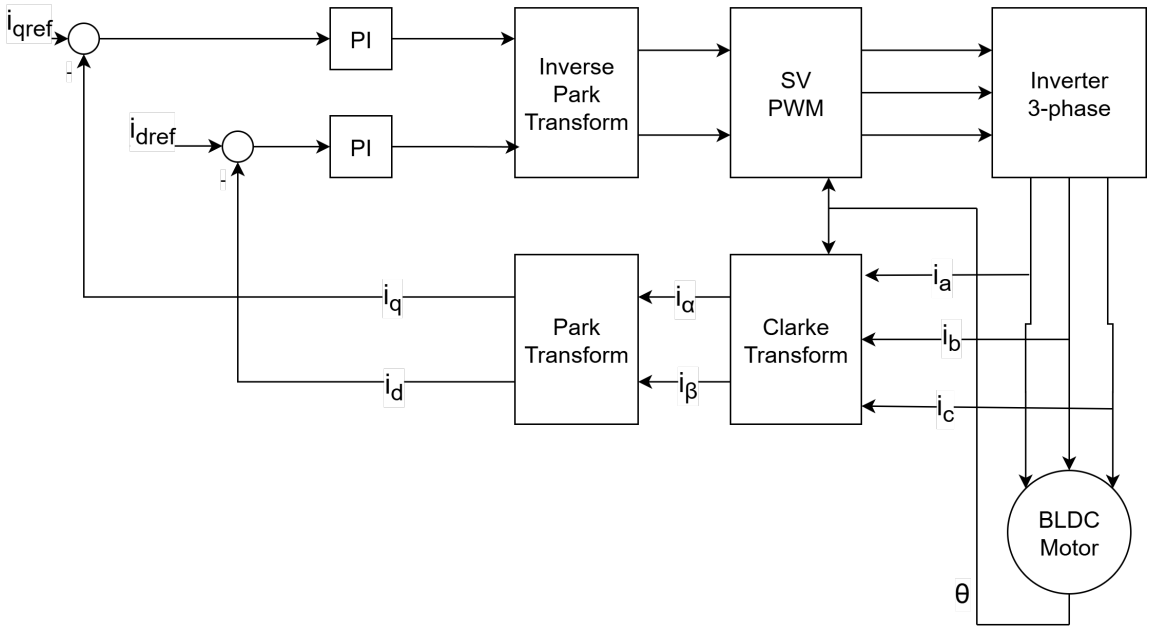


Figure 3.2: Field-Oriented Control method incorporating the torque loop and current regulation utilizing three-phase current sensing

- The i_q controller monitors the torque command issued by the outer loop and modifies v_q accordingly.

Each PI controller juxtaposes the reference ($i_{q_{ref}}$ or $i_{d_{ref}}$) with the measured value and produces the corresponding voltage reference (v_q or v_d). The voltages are subsequently converted back into the stationary $\alpha\beta$ frame by the inverse Park transformation, yielding v_α and v_β.

The resultant voltage vector is transmitted to the SVPWM block, which determines the necessary switching times for the inverter according to the current sector of the space vector diagram. The control loop is completed when the inverter provides the motor windings with the proper three-phase voltages.

This closed-loop approach facilitates accurate and independent regulation of torque and flux. The torque control loop functions at a reduced frequency compared to the inner current loop, yet it supplies the dynamic torque reference $i_{q_{ref}}$ essential for sustaining performance amid fluctuating loads or motion profiles.

Cascaded Control Loops

The overall control architecture is structured in a cascaded manner, comprising three nested loops:

- **Torque control loop (inner):** implemented via Field-Oriented Control (FOC), this loop directly regulates the quadrature-axis current i_q to control the torque produced by the motor. It operates at the highest frequency and acts directly on the motor's three-phase currents.
- **Velocity control loop (middle):** determines the torque-producing current reference i_q^* based on the velocity error. The reference angular velocity ω_{ref} is compared with the

measured angular velocity ω, typically using a PID controller:

$$i_q^* = K_{p\omega} e_\omega + K_{i\omega} \int e_\omega \, dt + K_{d\omega} \frac{de_\omega}{dt}, \quad \text{where } e_\omega = \omega_{\text{ref}} - \omega \tag{3.9}$$

- **Position control loop (outer):** generates a velocity reference ω_{ref} by comparing the desired position θ_{ref} with the actual rotor angle θ:

$$\omega_{\text{ref}} = K_{p\theta} e_\theta + K_{i\theta} \int e_\theta \, dt + K_{d\theta} \frac{de_\theta}{dt}, \quad \text{where } e_\theta = \theta_{\text{ref}} - \theta \tag{3.10}$$

The Smart Knob program incorporates an external position loop that manages user engagement, encompassing detents and end-stops, while the FOC loop ensures precise and fluid physical response at the motor level. The velocity loop may be activated or bypassed for rapid response, contingent upon the mode.

This hierarchical structure, shown schematically in Figure 3.3, supports high bandwidth inner loops for torque control while enabling intuitive haptic behaviors at the position loop level.

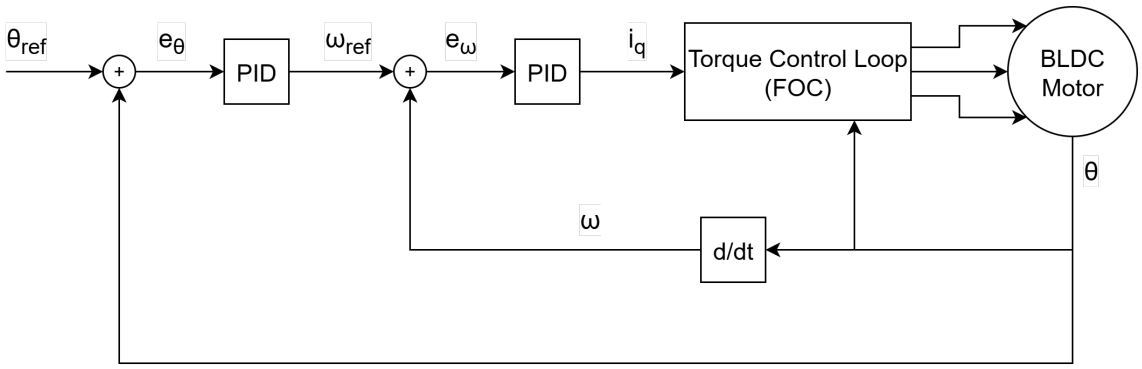


Figure 3.3: Cascaded control architecture for position and velocity regulation

This layered architecture allows the Smart Knob to deliver improved responsiveness and genuine haptic feedback while maintaining system stability. Control commands are generated at the highest level (position loop) and conveyed lower to torque commands, which are meticulously executed by the FOC loop.

3.2. Analysis of Software Solutions: Methods and Technologies

This chapter delineates the technical study and architectural design of the Smart Knob project. It explains the reasoning for the choice of particular software and hardware components, delineating their integration to fulfill the system's fundamental requirements. The principal design objectives were to establish a highly responsive haptic interface, guarantee reliable wireless connection for smart home integration, and develop a modular and extensible system. The subsequent sections will analyze these design choices, first with an examination of the software solutions, methodologies, and technologies utilized.

3.2.1. Platform and Framework Selection

The project utilizes the **ESP32** microcontroller platform, specifically the ESP32-WROOM-32 development board. This platform was selected for its exceptional ability to meet the project's stringent criteria. The primary benefits encompass:

- **Dual-Core Architecture:** The dual-core Xtensa LX6 processor of the ESP32 is essential for attaining responsive haptics. It facilitates the total segregation of the time-critical motor control task from the primary application logic, so averting jitter or lag in the feedback loop that could otherwise diminish the user experience.
- **Integrated Wi-Fi:** Native compatibility with 802.11 b/g/n Wi-Fi is crucial for the primary communication with Home Assistant through the MQTT protocol. This integration streamlines the hardware architecture by obviating the necessity for an external Wi-Fi module.
- **FreeRTOS Support:** The provision of a real-time operating system (RTOS) facilitates preemptive multitasking, which is utilized to concurrently and efficiently manage motor control, display updates, and network connectivity.

The firmware was created utilizing the **PlatformIO IDE** in Visual Studio Code. PlatformIO was selected instead of the conventional Arduino IDE due to its enhanced project management, library dependency oversight, and efficient build/upload procedure, which are essential for a project of this intricacy. The foundational code is developed with the **Arduino framework**, which offers an extensive array of high-level APIs for hardware interface, hence expediting the development process.

3.2.2. Core Software Libraries

The operation of the Smart Knob is significantly dependent on a collection of specialized, open-source libraries. Each was chosen for its performance, attributes, and strong community support, which were crucial for the effective execution of the project's intricate features. The libraries are encapsulated in Table 3.1.

3.2.3. Real-Time Processing and Multitasking

To guarantee a flawless and persuasive user experience, haptic feedback must be produced in real-time without any noticeable latency. A delay of even a few milliseconds between the actual turn of the knob and the resultant haptic feedback would disrupt the perception of a physical mechanism, rendering the response feel disjointed and artificial.

To satisfy this essential real-time need, the firmware utilizes the ESP32's dual-core design by assigning the most time-critical activities to a certain processor core. The motor control loop, tasked with reading the sensor, doing the FOC calculations, and changing the motor's state, operates solely on Core 0 with the highest RTOS priority. This preemptive scheduling ensures that its execution is not hindered by other, less critical operations. All remaining application logic, including button click handling, display updates, and network connectivity management, is executed on Core 1. The task to core distribution across the ESP32 Cores are depicted in Figure3.4.

Table 3.1: Core Software Libraries and Their Purpose

Library	Version	Purpose and Justification
Simple FOC	2.3.3	Employs Field-Oriented Control (FOC) for the Brushless Direct Current (BLDC) motor. Crucial for the seamless, high-torque, and accurate regulation.
SimpleFOCDrivers	1.0.7	Delivers hardware-specific driver support for Simple FOC, ensuring compatibility with the DRV8313-based driver board utilized in this project.
PubSubClient	-	An efficient and dependable client for MQTT communication. This library manages all interactions with the Home Assistant broker.
TFT_eSPI	-	An efficient graphics library for operating the TFT display. Its capability for sprites facilitates flicker-free rendering of the user interface.
WiFi	-	A component of the ESP32 core utilized for managing the Wi-Fi connection necessary for MQTT communication.

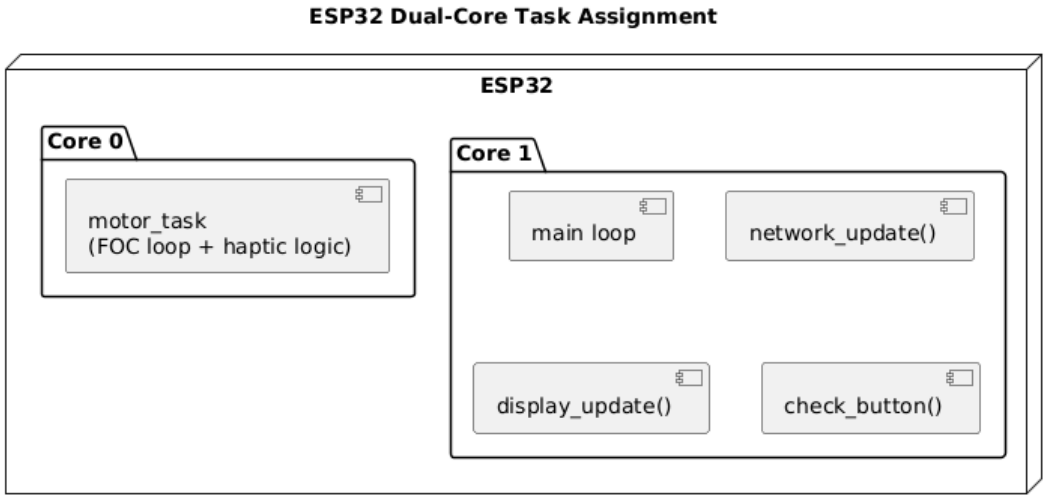


Figure 3.4: Task Distribution Across ESP32 Cores

3.3. Software Architecture

The firmware features a modular architecture that facilitates a distinct separation of concerns, hence enhancing maintainability, debuggability, and extensibility of the code. The system consists of multiple essential software modules, each assigned a specific function, as depicted in Figure 3.5.

3.3.1. Modular Firmware Architecture

Each module's primary responsibilities are as follows:

- `main.cpp`: The application's main point of entry is this file. It starts the main application

loop, initializes all hardware and software components, and uses the FreeRTOS API to distribute and assign specific tasks to the appropriate cores.

- `motor.cpp`: The haptic engine's core component is this file. The AS5048A encoder is continuously read, the SimpleFOC library is initialized, and the state machine that controls the behavior of each control mode—such as Light Brightness and Fan Speed—is controlled. Depending on the current mode, it calculates and applies the necessary voltages to the motor driver to produce the desired haptic effects.
- `display.cpp`: All visual output to the circular TFT display is controlled by this module. It consists of routines that generate the unique user interface for every mode, which includes colored forms, arcs, and text. It uses a sprite buffer, a common technique in embedded graphics programming, to ensure that updates are displayed on the screen smoothly and without flickering.
- `network.cpp`: All network-related operations are managed by this module. Connecting to the specified Wi-Fi network, establishing a connection to the MQTT broker, and sending state changes from the knob to the relevant MQTT topics for Home Assistant to use are the primary responsibilities.
- `config.h`: All of the crucial system parameters are compiled in this primary header file. It details MQTT topics, Wi-Fi and MQTT credentials, hardware pin configurations, and crucial haptic tuning parameters (like PID gains). Without changing the core application logic, this approach makes it easier to configure and modify the system's functionality.

These modules function interdependently, constituting a unified system with distinct dependencies. The 'main.cpp' module serves as the primary orchestrator, invoking update routines within the motor, display, and network modules. The 'motor.cpp' module serves as the principal data producer, creating state information, such as the current angle or mode, which is utilized by the 'display.cpp' module for UI rendering and the 'network.cpp' module for publishing MQTT updates. This establishes a unidirectional control flow from the main component and a distinct data flow from the core haptic engine to the output modules, as illustrated in Figure 3.5.

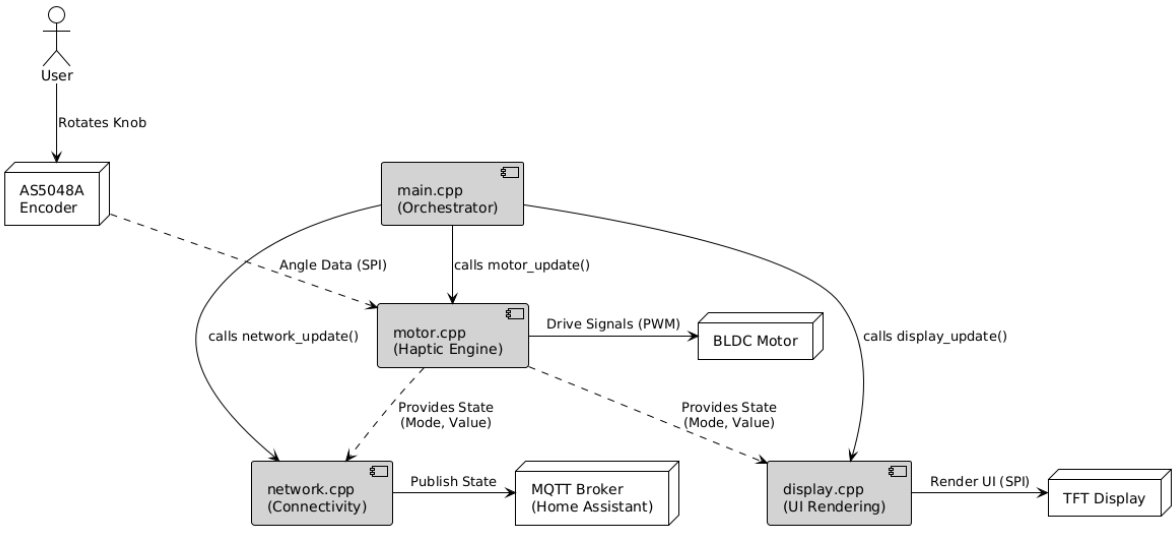


Figure 3.5: Software Module Interaction Diagram

This modular, event-driven architecture renders the system exceptionally extensible. Introducing a new control mode essentially entails including an additional state into the mode enumeration, implementing its logic within the motor C++ state machine, and integrating a suitable user interface for ‘display.cpp‘, necessitating minor alterations to the main framework.

3.3.2. Home Assistant Integration and Network Communication

The Smart Knob’s lightweight MQTT communication allows for seamless integration with the Home Assistant platform. Every functional mode, such as temperature, brightness, and volume, has a corresponding MQTT topic. The system publishes an update to the relevant topic whenever a major state change occurs, like turning a knob or pressing a button.

Home Assistant can subscribe to these topics and carry out automation logic based on the received values thanks to this decoupled architecture, which eliminates the need for direct firmware control over the knob.

For consistency and ease of integration with Home Assistant’s `configuration.yaml` and `automations.yaml` logic, the MQTT topics are defined statically in the `config.h` file.

All connectivity to the MQTT broker is managed by the underlying library `PubSubClient`. High reliability is ensured by automatically handling reconnection attempts and continuously monitoring connection status.

The communication flow between the Smart Knob device, the MQTT broker, and the Home Assistant is depicted in Figure 3.6. Home Assistant receives and processes published messages from user interactions on the device to initiate actions like controlling media playback or lighting.

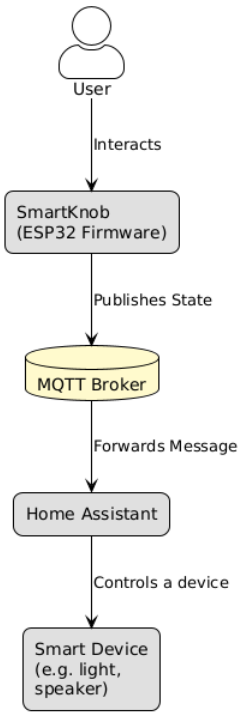


Figure 3.6: Smart Knob Communication Flow via MQTT and Home Assistant

Future extensibility and system modularity are encouraged by this loosely coupled architecture. With little effort, the Smart Knob can be added to other MQTT-based smart home ecosystems or expanded with new modes.

3.4. Hardware Analysis and Design

The hardware for the Smart Knob was chosen after a thorough evaluation, with each component picked to meet specific performance, form factor, and integration requirements. The aim was to create a high-performance, autonomous, and reliable gadget. This section elucidates the rationale for each component selection and delineates their electrical and mechanical integration to constitute the final system.

3.4.1. Component Selection

The hardware is divided into three main subsystems: the core logic, the haptic and visual feedback system, and the power system.

The Core Logic: ESP32 Development Board

The ESP32 platform was chosen as the principal controller above traditional Arduino boards (e.g., Uno, Mega) or STM32 microcontrollers. The ESP32 offers an impressive set of features essential for this project: its powerful dual-core processor is necessary for simultaneously handling real-time haptics and application logic, a task unmanageable by a single-core controller. Furthermore, its built-in Wi-Fi and Bluetooth capabilities are crucial for seamless smart home integration via MQTT, eliminating the need for additional, cumbersome modules often associated with various STM32 or Arduino boards. The comprehensive integration on a single, cost-effective chip made it the ideal choice.

Haptic and Visual Feedback Mechanism

This subsystem enables primary user interaction by combining tactile feedback with a graphical interface. It comprises:

- **iPower GM3506 Gimbal Motor:** A brushless DC (BLDC) motor is employed due to its ability to provide high torque, operate quietly, and provide the precise control required for producing genuine haptic effects. The iPower GM3506 was selected for its crucial mechanical feature: a hollow shaft with an internal diameter adequate to accommodate the display cable. This was a crucial need, since it enables the routing of display wires directly through the motor's axis of rotation, allowing for unrestricted, 360-degree movement of the knob without the risk of entanglement or mechanical failure.
- **ams AS5048A Rotary Encoder:** The ams AS5048A magnetic rotary encoder was utilized for precise position tracking. The unit was supplied as a pre-integrated assembly with the GM3506 motor, guaranteeing optimal alignment between the magnet and the sensor, a common point of failure in custom systems. Choosing a magnetic encoder over a mechanical one ensures a prolonged operational lifespan free from physical deterioration. The 14-bit resolution (16,384 steps per revolution) provides the accurate data required for

the effective operation of the FOC algorithm and the creation of intricate, high-fidelity haptic effects.

- **SimpleFOC Mini Motor Driver:** The choice of the motor driver was closely associated with the software library. The SimpleFOC Mini board was selected for its specialized design and certification, ensuring optimal interoperability with the Simple FOC library. This guarantees compatibility and optimizes the development process. The board, employing the Texas Instruments DRV8313 IC, is optimal for the application: it supports a 12V power supply, provides adequate current (2.5A peak) for the motor's haptic needs, and includes built-in protection features, ensuring dependable and safe operation.
- **1.28-inch Round TFT LCD Display:** The display was necessary to provide clear, real-time visual feedback aligned with the device's physical design. The 1.28" circular TFT display was chosen for its size and shape, which aesthetically complement a control knob. It provides a vibrant, full-color interface for displaying modes and values, and its standard SPI interface enables effortless connectivity with the ESP32.
- **Tactile Pushbutton:** A tactile pushbutton is utilized to allow the user to traverse various control modes. This button is attached directly to the external electrical enclosure housing the project. The custom-designed 3D-printed exterior shell surrounding the motor features a precisely constructed internal expansion. Upon the user pressing the knob, this extension operates as an actuator, conveying the force to the stationary button on the box, enabling a simple yet robust contact mechanism.

Power System

The power system was designed to be robust and self-sufficient, providing energy to all components from a single wall outlet.

- **12V, 60W AC-DC Power Supply:** This was chosen to meet the peak power demands of the BLDC motor, which can draw significant current while generating strong haptic feedback. Maintaining a stable, high-current 12V supply is crucial for the motor driver's efficacy and the overall system functionality.
- **LM2596 Buck Converter:** This device is a crucial element of the power architecture, as it efficiently converts the primary 12V supply to a stable 5V rail. The 5V output is thereafter delivered directly to the ESP32 development board. The development board's internal voltage regulator transforms the 5V supply into the 3.3V necessary for the ESP32 microcontroller, the AS5048A encoder, and the TFT display, thereby creating a reliable and efficient power distribution system.

3.4.2. System Integration and Architectural Design

This section outlines the physical and electrical integration of the chosen components into a working and small embedded system.

Electrical Integration

The electrical design was developed to minimize wiring complexity and ensure clear separation between power and logic domains. The block diagram of the system (Figure 3.7) demonstrates that the ESP32 microcontroller serves as the principal controller, interfacing with

all peripheral devices. Essential connections comprise the common SPI bus for the encoder and display, the three PWM outputs for the motor driver, and the digital input for the push-button. This centralized control design guarantees synchronous operation.

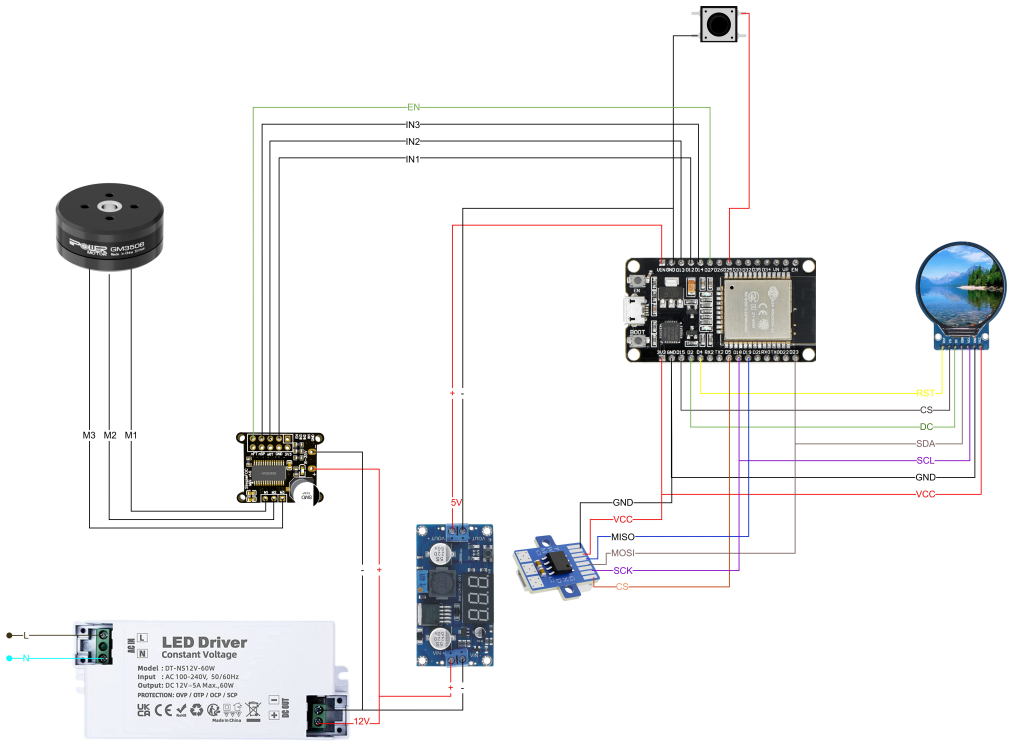


Figure 3.7: Electrical Block Diagram of the System

The electrical design of the Smart Knob was developed to ensure minimal wiring complexity, noise resistance, and clear delineation between power and logic domains. The system functions using a single 12V DC adapter, providing power to both the high-power motor driver and the logic-level components using a buck converter. The ESP32 microcontroller serves as the principal controller, interfacing with all peripheral devices as depicted in the system's block diagram (Figure 3.7). Essential relationships comprise:

- The AS5048A encoder interfaces using SPI, providing high-resolution angular position data.
- The BLDC motor driver is controlled through three PWM outputs.
- The circular TFT display also connects via the common SPI bus.
- The central push-button is observed via a digital GPIO pin.

Figure 3.7 illustrates the data and power linkages. The ESP32 operates as the SPI master for both the encoder and the display, utilizing distinct Chip Select lines for each device. The motor controller is operated via specific PWM channels. Ground planes are employed for both the logic and power domains, supplemented by suitable decoupling to maintain signal integrity.

Mechanical Housing and Assembly

The Smart Knob container is designed for compactness and mechanical dependability. All components are contained within a custom 3D-printed enclosure tailored to fit a typical surface-mounted electrical box. The exploded mechanical perspective (Figure 3.8) illustrates

that the design consists of several principal components: a motor mount with encoder support, a rotating screen carrier, and a back shell for housing the essential electronics. This modular and pragmatic method enables efficient construction and maintenance while maintaining a tidy, user-friendly appearance. The components were designed and the view was rendered in Fusion 360.

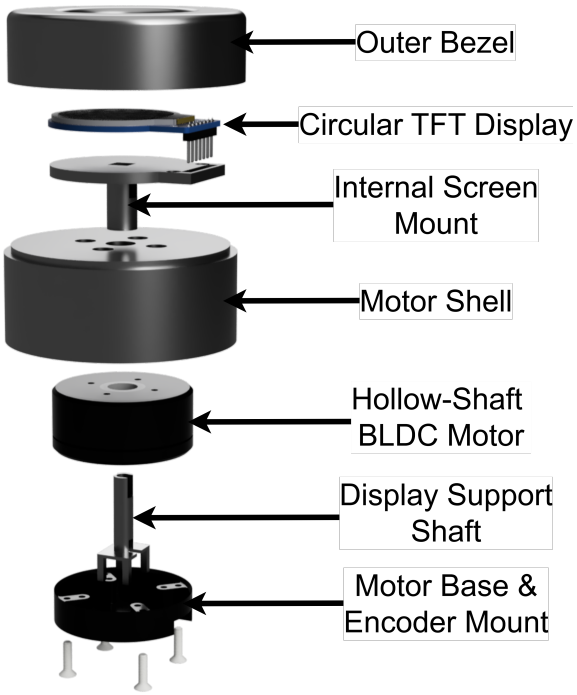


Figure 3.8: Exploded Mechanical Representation of the Smart Knob Assembly (CAD Render)

The Smart Knob enclosure is designed for compactness, user accessibility, and mechanical dependability. All components are contained within a custom 3D-printed enclosure designed to fit into a standard surface-mounted electrical box. The exploded mechanical view (Figure 3.8) illustrates that the design consists of three principal components:

1. **Motor mount and encoder support**: Ensures precise alignment between the motor and its embedded magnetic encoder.
2. **Screen mounting system**: This is a multi-part assembly designed for stability and a clean aesthetic.
    - **Inner Stabilizer and Screen Holder:** A primary 3D-printed mount is fitted inside the motor's rotating part. This piece serves two functions: it stabilizes itself by fitting around the stationary encoder, and it provides a secure platform to hold the TFT display, ensuring it is perfectly centered and stable during rotation.
    - **Outer Cosmetic Bezel:** A second part, a thin bezel, is placed over the screen holder. Its purpose is purely aesthetic: it precisely covers the display's green PCB, leaving only the circular screen visible to the user. This creates a much cleaner, more professional finish than an exposed circuit board.

The employment of a hollow shaft BLDC motor was a crucial design decision. This allows the ribbon wire of the TFT screen to traverse the axis of rotation, thereby preventing entanglement or damage during prolonged use. The push-button is mechanically pressed by the outer shell of the motor. This modular and pragmatic method enables assembly and maintenance while maintaining a clean, user-friendly appearance. The components were engineered in Fusion 360 and fabricated using PLA.

4. Implementation and Testing

4.1. Implementing the Prototype

Implementing the Smart Knob prototype involved two simultaneous activities: assembling the hardware into a unified whole and developing the software that controls the device's operation. The hardware assembly and firmware architecture that manage haptics, display rendering, and network connectivity are explained in this section.

4.1.1. Hardware Integration

A custom 3D-printed knob assembly serving as the user interface and a commercially available surface-mounted electrical box housing the electronics make up the prototype's two main parts. This technique combines the robustness of a traditional electrical component with a unique interface design. The electrical box's internal assembly is shown in Figure 4.1.

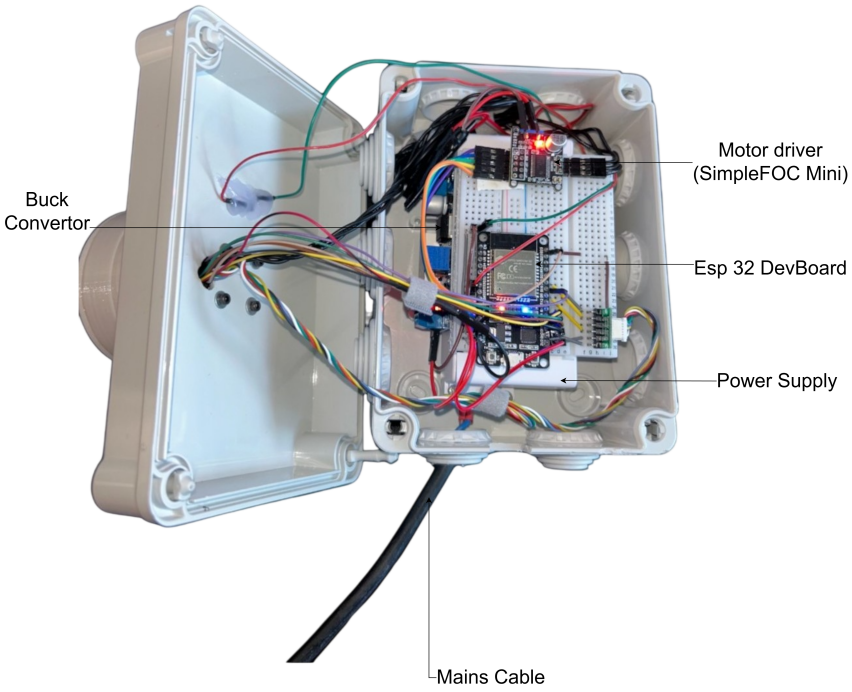


Figure 4.1: The Smart Knob prototype's internal configuration, highlighting the Buck Converter, SimpleFOC Mini motor driver, and ESP32 DevBoard housed in the electrical enclosure.

The crucial hardware elements found inside the electrical enclosure include:

- **Microcontroller:** The main CPU is an ESP32 development board, which also manages

the Wi-Fi connection, motor control, and display handling.
- **Motor Driver and Power:** A buck converter lowers the voltage for the logic circuits, and a SimpleFOC Mini motor driver controls the haptic motor.

The user interaction systems are included in the custom knob unit that is attached to the electrical box's cover:

- **Haptic System:** The driver is connected to a brushless DC (BLDC) gimbal motor (iPower GM3506) to generate haptic feedback that can be adjusted. A pre-integrated ams AS5048A magnetic rotary encoder is used to track the exact location.
- **Visual Feedback:** The real-time graphical user interface is displayed on a 1.28-inch circular TFT display with a resolution of 240x240.
- **Input:** By pressing the knob's outer shell, a tactile pushbutton that allows the user to select between control modes is mechanically activated.

The exploded CAD rendering in Chapter 3 (refer to Figure 3.8) illustrates the modular mechanical assembly of the knob unit. The hollow shaft of the BLDC motor is a crucial component that facilitates the routing of the display's individual wires along the axis of rotation, enabling unrestricted 360-degree movement without the potential for entanglement or mechanical failure.

The outer bezel, internal screen mount, motor shell, and display shaft support were created in CAD and produced using PLA filament through FDM 3D printing. These components provide the structural base of the custom haptic interface and fulfill specific mechanical and functional roles:

- **Outer Bezel:** A decorative encasement that obscures the display's PCB, revealing solely the active screen area. It has a sleek and minimalist frontal design for the knob.
- **Internal Screen Mount:** A bespoke mount that firmly secures the circular display. It features a hollow post aligned with the motor shaft, facilitating the unobstructed passage of display wires through its center while preserving the overall circular form factor.
- **Motor Shell:** A protective casing for the BLDC motor that functions as a structural foundation for affixing the outer bezel. It is affixed directly to the motor housing via M2 screws, ensuring mechanical stability. Furthermore, it features a protrusion that mechanically activates the internal pushbutton utilized to cycle through control modes upon pressing the knob.
- **Display Shaft Support:** A robust internal framework that secures the display mount to the motor base. It extends along the motor shaft and has four stabilizing legs arranged around the encoder. A notch at the base permits the display wires to exit towards the ESP32 board without obstructing the spinning assembly.

4.1.2. Software Stack Overview

The Arduino framework in the PlatformIO IDE is used to create the firmware in C++. It was chosen for its efficient build process and library management, both of which were essential for managing the complexity of the project. The implementation follows the modular architecture described in Chapter 3 (see Figure 3.5), which was crucial to the development

process. In addition to making debugging easier, the separation of concerns into distinct modules allowed for a more organized workflow.

The following crucial software modules make up the system:

- `main.cpp`: The main entry point of the application. Importantly for implementation, it uses FreeRTOS to allocate the time-sensitive motor control task to a particular CPU core after initializing all hardware and software components. This ensures that, despite other tasks like networking or display refreshes, haptic feedback is always responsive and fluid.
- `motor.cpp`: The haptic engine's core component. This module is responsible for reading the sensor location, implementing the state machine that defines the tactile experience of each control mode, and running the FOC algorithm.
- `display.cpp`: Uses the `TFT_eSPI` library to render the unique user interface for each mode and manages all visual output to the circular TFT display. A sprite buffer, a common and effective technique in embedded graphics programming, was used to ensure flicker-free rendering.
- `network.cpp`: Controls all network functions, such as setting up a Wi-Fi connection and managing the MQTT connection for Home Assistant communication via the `PubSubClient` library.
- `config.h`: A primary header file that includes all of the necessary system parameters, such as hardware pin configurations, MQTT topics, Wi-Fi credentials, and haptic tuning constants (like PID gains). In order to easily modify the knob's functionality during testing without changing the core application logic, this design choice was crucial.

4.1.3. Communication Layer

Through Wi-Fi and the MQTT protocol, the Smart Knob connects to the vast smart home ecosystem. Because of its effectiveness in separating the control knob from the devices it regulates, this lightweight publish-subscribe model was chosen to create a scalable and reliable architecture. The `network.cpp` module, which uses the `PubSubClient` library to handle the complexities of the MQTT protocol, contains all network-related code.

There are several stages to the communication stack's execution. The `network_init()` function is used in the initial setup. Using the credentials listed in `config.h`, this function invokes `WiFi.begin(WIFI_SSID, WIFI_PASSWORD)` to create a connection to the local network. Until a successful Wi-Fi connection is made (`WiFi.status() == WL_CONNECTED`), the device then stays in a loop. It uses `client.setServer()` to configure the MQTT server connection after a Wi-Fi connection has been made.

The primary network loop employs a watchdog-like mechanism to guarantee a consistent and reliable connection. Through `client.connected()`, the `network_update()` method continuously checks the device's connection to the MQTT broker. The `reconnect_mqtt()` method is called when the connection is lost. Every five seconds until the connection is successfully restored, this function starts a blocking loop in which it repeatedly tries to reconnect to the MQTT broker using the given credentials. This ensures that the knob can recover on its own without human assistance in the event of broker restarts or network outages. You can examine the entire implementation in Appendix A.1.

4.2. Position Control Implementation and Testing

This section describes how the various haptic feedback modes that make up the Smart Knob's user experience are implemented. Every mode is designed to provide a distinct tactile experience that naturally complements the digital function under control. The concept of a "virtual angle," which distinguishes the knob's actual physical position from the value it represents, is a key feature of this design. This technique enables smooth mode changes without requiring unnecessary mechanical movement of the knob, which is essential for a user-friendly interface.

Every firmware iteration calculates the virtual angle by adding a mode-specific offset to the motor shaft's real-time physical angle: `currentVirtualAngle = physical + modeOffset[current_mode];`. When the user switches between modes, true innovation takes place. The firmware first saves the last known virtual angle for the mode being exited into the `modeMemory[]` array. After that, it retrieves the stored angle for the new mode and calculates a new offset for it. To ensure that the new virtual angle matches its saved value at the knob's current physical position, this offset is carefully calibrated. As a result, the user feels no torque or movement when switching modes, and the knob is instantly ready for operation from its previously saved state. Instead of using the physical angle, this virtual angle is used to calculate the haptic feedback for all modes. Appendix A.2 provides more details on how this virtual angle system is implemented.

The study will look at the fundamental code that generates a variety of effects, from simple detents to more complex, confined spring-like pressures. We'll also talk about how to evaluate each mode's responsiveness and tactile sensation.

4.2.1. Continuous Control: Brightness, Volume, and Media

The modes for controlling light brightness, media volume, and media track selection are functionally grouped together because of their similar haptic interaction model. This model provides the user with an analog-style, continuous rotation within a predetermined range, which is terminated by a virtual spring-like boundary. Since the brightness mode is both aesthetically and functionally similar, Figure 4.2 only shows the volume mode's user interface for the sake of conciseness.

The implementation of this haptic feedback is managed entirely within the main motor control loop in `motor.cpp`, as detailed in Appendix A.3. The logic operates on the calculated virtual angle and is broken down into two distinct physical sensations:

1. **Damped Free-Spin:** When the knob's virtual angle is turned within its valid angular range (0 to 180 degrees), the firmware creates a mild damping effect by making the motor's voltage inversely proportional to its current rotational speed: `voltage = -HAPTIC_KD_DAMPING * motor.shaft_velocity;`. This gives the user a smooth, fluid movement sensation, similar to a high-quality physical dial.
2. **Virtual End-Stops:** When the user pushes the virtual angle beyond its predetermined boundaries, true haptic innovation takes place. The control logic switches from basic

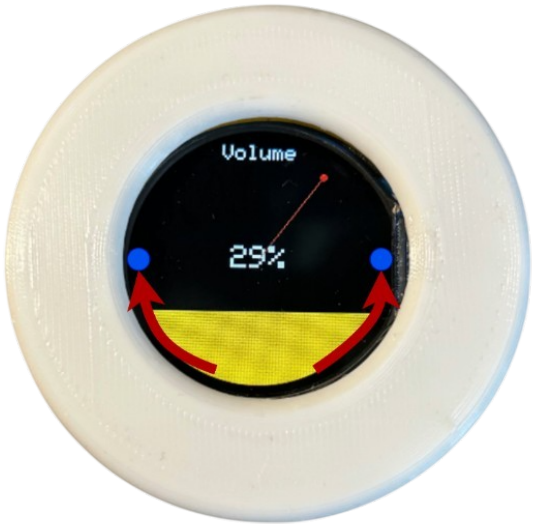


Figure 4.2: The user interface for Volume Control.

damping to an active-resistance mode, engaging a PID controller (`voltage = haptic_pid(...)`). This creates a spring sensation. Importantly, the calculated error rises as the user applies more force against the spring, which causes the motor to produce a correspondingly stronger opposing torque. The red arrows in Figure 4.2 illustrate this behavior, where the restoring force increases as the user moves deeper into the restricted zone. The intensity of this end-stop force is tuned by the `HAPTIC_KP_ENDSTOP` constant in the `config.h` file.

This particular haptic model is skillfully reused in the **Media Control** mode. Although the visible feedback changes, the underlying physics remains the same, as shown in Figure 4.3. When the boundary is reached, the `network_publish_media_control()` function is triggered, sending a "NEXT" or "PREVIOUS" command. To enhance the user experience for this mode, the display shows a white dot that moves horizontally, giving the impression that a horizontal slider is being adjusted to its limits rather than a dial being turned.

Testing and Validation

Both qualitative and quantitative assessments were used to validate these modes. To verify that the damping and end-stops were "natural" and "responsive," users were asked to describe their impressions. Quantitatively, the MQTT messages for brightness, volume, and media control were monitored in Home Assistant to ensure that state changes were reported promptly and accurately when the knob was operated, especially when triggering end-stop actions in Media Control mode.

4.2.2. Discrete Mode: Fan Speed

In contrast to the continuous modes, the Fan Speed mode provides a discrete interaction model with distinct "snap-points" or detents. This haptic feedback simulates the feel of a conventional mechanical selector switch, where each position corresponds to a specific speed

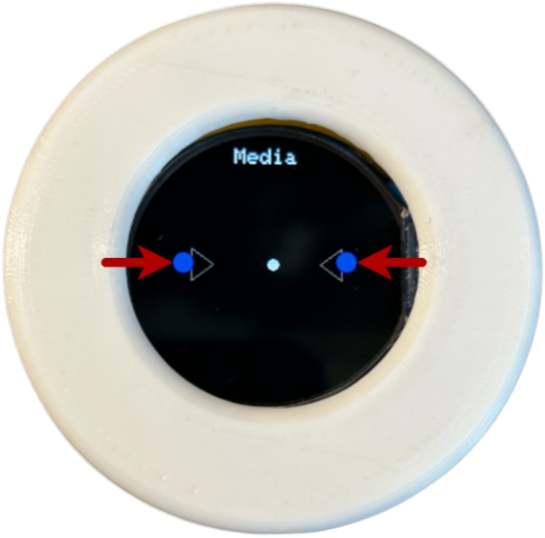


Figure 4.3: The user interface for Media Control.

level.

The physical sensation is achieved by defining four stable angular positions, or detents, spaced 90 degrees apart. The firmware continuously determines which of these detents, visualized by the blue dots in Figure 4.4, is closest to the knob's current virtual angle. A PID controller is then engaged to generate an attractive force, safely drawing the knob into that stable position.

Figure 4.4: The user interface for Fan Speed.

The `findDetent()` helper function, which calculates the closest target angle based on the virtual angle, is the central component of this logic. When the user turns the knob past the halfway point between two detents, this function updates the target attractor angle to the next position. The PID controller then immediately redirects its force, creating the crisp "snap"

sensation illustrated by the red arrow in Figure 4.4. The `HAPTIC_KP_DETENT` constant controls the snap's intensity, and the PID controller's Derivative (D) gain is purposefully set to zero to create a sharper, more mechanical click without any damping. The complete implementation can be reviewed in Appendix A.4.

Testing and Validation

The user's perception of the detents was the main focus of this mode's evaluation. The primary qualitative goal was ensuring the "snap" was perceived as distinct and satisfying, not soft or weak. The motor's torque had to be strong enough to hold the position against accidental bumps but light enough to be turned purposefully without excessive force. To verify that the correct speed level (0, 1, 2, or 3) was published as the knob settled into each of the four detents, the `smartknob/fan/speed` MQTT topic was quantitatively monitored.

4.2.3. Bounded Detent Mode: Temperature Control

The Temperature Control mode uses a hybrid haptic model that combines the hard boundaries of the continuous modes with the discrete detents of the Fan Speed mode. This allows the user to feel each "click" as a precise temperature increment (0.5°C), while preventing the selection of a value outside a predefined comfortable range. The user interface, depicted in Figure 4.5, displays the selected temperature and a cyan arc that shows the current position within the allowed range.

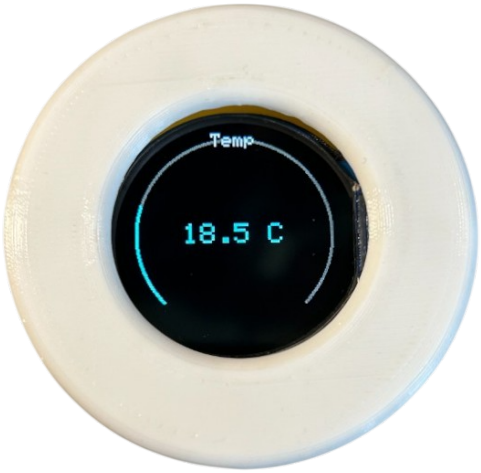


Figure 4.5: The user interface for Temperature Control.

The internal logic in `motor.cpp` first checks if the knob's virtual angle falls within the allowed operating arc, defined by `TEMP_MIN_ANGLE` and `TEMP_MAX_ANGLE`.

- **Within the Range:** If the virtual angle is inside the valid arc, the firmware uses the `findDetent()` function to find the closest stable position based on the `TEMP_DETENT_ANGLE`. Then, using the same crisp `HAPTIC_KP_DETENT` gain as in the Fan Speed mode, a PID controller generates a "snap" force toward that detent.

- **Outside the Range:** When the user tries to rotate past the minimum or maximum angle, the logic ignores the detents and engages the PID controller to create a firm, spring-like end-stop at the boundary, preventing further rotation.

This dual functionality ensures both precise, incremental adjustment and clear physical boundaries. The full implementation can be reviewed in Appendix A.5.

Testing and Validation

The validation of this mode assessed both haptic models. The end-stops were evaluated for firmness and the detents for clarity. Quantitatively, the accuracy of the mapping from angle to Celsius was verified by comparing the temperature on the screen with the knob's position. The `smartknob/temperature` MQTT topic was monitored to ensure that temperature updates were published for every detent and that the value remained within the defined `TEMP_CELSIUS_MIN` and `TEMP_CELSIUS_MAX` bounds.

4.2.4. Cyclic Detent Mode: Color Picker

The Light Color mode provides a unique tactile experience designed for continuously cycling through a preset palette of options. It has no start or end-stops; the knob can be freely spun 360 degrees. The user can feel each color option while turning the knob thanks to the haptic feedback's distinct detents. The visual interface, shown in Figure 4.6, displays a 4x4 grid with 16 preset colors, and a white border highlights the currently selected color.

Figure 4.6: The user interface for the Color Picker.

The implementation of this mode applies continuous rotation and uses the same detent logic as the Fan Speed and Temperature modes. To ensure a smooth haptic effect over the 360-degree boundary, the firmware uses the `fmodf()` function to "wrap" the virtual angle.

An interesting aspect of the implementation is the mapping between the physical detents and the available colors. The motor provides 18 physical detents (one every 20 degrees, as per

COLOR_MODE_DETENT_ANGLE) in a full rotation. However, the color palette contains only 16 colors. To correlate the physical rotation with the 16-color grid, the firmware uses the modulo operator (% 16). This means that as the user turns the knob past the 16th color, the selection automatically cycles back to the first color, allowing for infinite rotation through the palette. The PID controller settings are identical to the other detent-based modes to ensure a crisp, mechanical feel for every selection. The full implementation can be reviewed in Appendix A.6.

Testing and Validation

The accuracy of the color selection and the consistency of the haptic feedback were the main focus of this mode's validation. The detents were assessed to verify their uniqueness and consistent spacing throughout the entire 360-degree rotation. The smartknob/light/color_hs MQTT topic was monitored as the main quantitative test. The corresponding hue and saturation payload was checked against the expected values defined in the color_palette array in display.cpp after the knob was turned to each of the 16 positions. This ensured that the color sent to Home Assistant matched the color selected on the knob.

4.3. Display Function Implementation and Testing

An intuitive user experience is facilitated by the visual feedback provided by the Smart Knob's circular TFT display. The current control mode's status must be accurately shown on the display in real-time. This section describes how each mode's visual interface is rendered, how performance is taken into account to ensure smooth updates, and how the display's functionality is assessed.

4.3.1. Visual Interface Rendering

The TFT_eSPI graphics library is used to render the visual output, and each control mode has a unique layout that is implemented within a switch statement in the display_update() function of display.cpp. Each mode's implementation is described below.

Brightness and Volume UI

The brightness and volume modes' user interface is clear and simple, showing a percentage value that corresponds to the knob's position. The implementation, shown in Figure 4.2, calculates the percentage based on the motor's angle with respect to the ANALOG_MODE_MAX_ANGLE. Two graphic elements are then created using this percentage:

- A central text string is rendered using sprite.drawString() to show the numerical value.
- A vertical progress bar is rendered using sprite.fillRect(), providing a clear visual indicator with the height of the bar directly corresponding to the calculated percentage.

The code for this implementation is available in Appendix A.7.

Fan Speed UI

By displaying the discrete speed level (0, 1, 2, or 3), the Fan Speed mode's display reinforces the haptic detents. By dividing the current angle of the knob by the `FAN_DETENT_ANGLE` and rounding the quotient to the closest integer, the current speed level is calculated. As shown in Figure 4.4, this number is then displayed as a large green string in the center of the screen. The code snippet is in Appendix A.8.

Temperature Control UI

The temperature user interface, shown in Figure 4.5, provides two kinds of feedback. First, it calculates the temperature in Celsius by mapping the knob's angular position within its designated range (`TEMP_MIN_ANGLE` to `TEMP_MAX_ANGLE`) to the corresponding temperature range (`TEMP_CELSIUS_MIN` to `TEMP_CELSIUS_MAX`). This value is then displayed as text. Second, a cyan arc is rendered around the edge of the screen using `sprite.drawArc()` to display the current setting visually within the overall range. Specifics on the implementation can be reviewed in Appendix A.9.

Media Control UI

The Media Control user interface, seen in Figure 4.3, is designed to look like a horizontal slider. Two stationary triangles are rendered on the left and right sides of the screen to achieve this. After that, a small white circle is drawn, and its horizontal position is established by mapping the angle of the knob to the distance between the two triangles. The corresponding triangle fills with a solid color when the user presses against the haptic end-stops, signifying that the "NEXT" or "PREVIOUS" command has been triggered. For the code, see Appendix A.10.

Color Picker UI

The most graphically complex interface is the Color Picker, shown in Figure 4.6. Using `sprite.fillRect()`, the implementation creates a 4x4 grid of colored squares by iterating through the 16-color `color_palette` array. It creates a highlighting effect by drawing a white rectangle around the corresponding square after checking the active color index (determined by the `motor_get_color_index()` function) to indicate the current selection. Appendix A.11 contains the code used to render this grid.

4.3.2. Dynamic Updates and Performance

All rendering operations are carried out on an in-memory sprite buffer (`TFT_eSprite`) rather than directly on the display to give the user a responsive and flicker-free experience. This buffer is cleared at the start of every update cycle. The entire sprite is pushed to the physical display in a single operation using `sprite.pushSprite(0, 0)` after every element for the current mode has been rendered to the buffer. To prevent visual tearing and flickering that would result from rendering objects sequentially, this double-buffering technique is crucial. The `DISPLAY_UPDATE_MS` constant limits the refresh rate of the display in order to save CPU cycles for the high-priority haptic control task.

4.3.3. Testing and Validation

For every mode, haptic validation was carried out concurrently with display testing. Each user interface element was carefully checked to:

- Accurately match the value and state obtained from the firmware logic (for example, the displayed percentage matched the value sent over MQTT).
- Update smoothly and without noticeable lag (within the 50ms target) in response to knob movement.
- Be clearly readable and understandable in a variety of ambient lighting conditions.

This ensured that the visual feedback was not only accurate but also improved the overall user experience.

4.4. Home Assistant Integration

The integration of the Smart Knob into the Home Assistant ecosystem is the final and most important stage of the implementation. Through this integration, the knob is transformed from a standalone gadget into a powerful, centralized controller for a number of smart home appliances. Only the MQTT protocol is used to establish the connection, allowing for a highly decoupled and reliable architecture. This section describes how to set up Home Assistant to interpret knob messages and take appropriate action.

4.4.1. Entity Mapping via MQTT Sensors

Notifying Home Assistant of the information being published by the SmartKnob is the first step in the integration process. This is accomplished by defining a series of MQTT sensors in the main `configuration.yaml` file. Every sensor is configured to track a specific MQTT topic that corresponds to one of the knob's functions.

For example, the "SmartKnob Mode" sensor is a subscriber to the `smartknob/mode/state` topic. When the knob is pressed, the firmware publishes the new mode's name (e.g., "Light Brightness") to this topic, and the state of the "SmartKnob Mode" sensor in Home Assistant updates accordingly. Similarly, sensors are assigned to monitor a different topic for temperature, color, brightness, volume, and fan speed. This creates a direct link between the status of the knob and a corresponding set of Home Assistant entities. The full sensor configuration can be examined in Appendix A.12.

4.4.2. Automation Logic

The next step after creating the sensor entities is to create the logic that translates changes in these sensors into actions. The `automations.yaml` file contains a set of automations that are used to manage this. These automations act as a link between the input from the knob and the control of real devices.

A typical automation follows a simple trigger-condition-action model. The automation

for controlling a light's brightness is set up as follows:

- **Trigger:** Anytime the state of `sensor.smartknob_brightness_value` changes, the automation is triggered.
- **Condition:** The action is only executed if `sensor.smartknob_mode` is currently set to "Light Brightness." This crucial feature prevents brightness variations when the user modifies, for example, the volume.
- **Action:** The automation sets the brightness of the target light to the new value that was obtained from the knob by calling the `light.turn_on` service.

A more complex example is the media control automation, which uses a `choose` action to implement different sequences depending on the MQTT payload. It calls the `media_player.media_next_track` service if the payload is "NEXT," and the `media_player.media_previous_track` service if the payload is "PREVIOUS."

Figure 4.7 illustrates this decoupled architecture, where the knob only reports its status and Home Assistant chooses the proper response. The complete automation configuration can be reviewed in Appendix A.12.

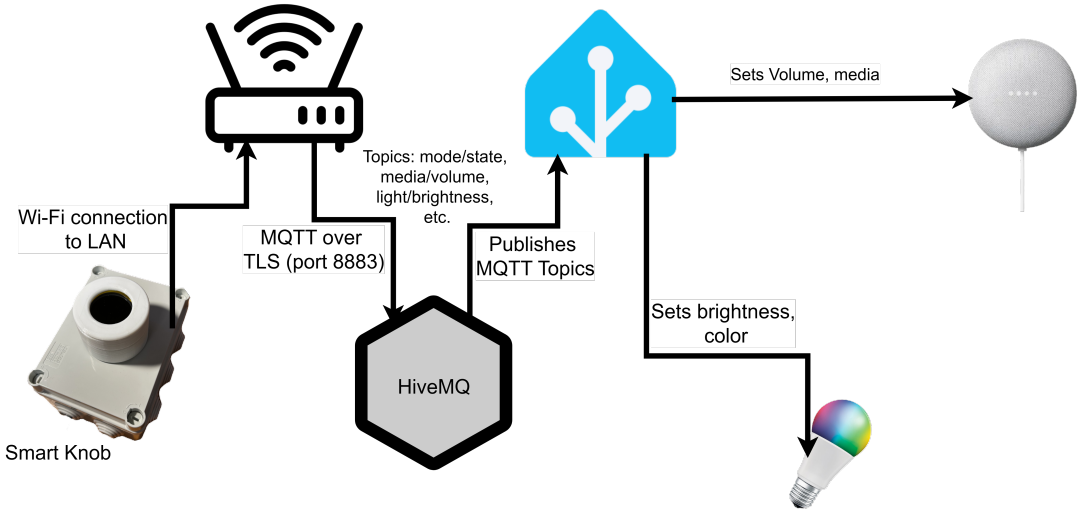


Figure 4.7: Communication flow between the Smart Knob, MQTT Broker, and Home Assistant.

In addition to publishing sensor values, Home Assistant automations facilitate the transmission of the SmartKnob's input to actual devices within the smart home ecosystem.

In this configuration, two physical devices are amalgamated for demonstration and validation purposes:

- **Google Nest Mini** (`media_player.office_speaker`)**:** This device utilizes the `media_player.volume_set`, `media_next_track`, and `media_previous_track` services for volume and playback control in Home Assistant. These operations are activated when the SmartKnob is configured to *Media Volume* or when a flick gesture transmits a `NEXT` or `PREVIOUS` command over the `smartknob/media/control` topic.
- **LEDVANCE Intelligent Illumination Device** (`light.light`)**:** This RGB bulb obtains brightness updates via the `light.turn_on` service, with the brightness value received from the `sensor.smartknob_brightness_value`. Moreover, color modifications are

managed by transmitting HSV values over the `smartknob/light/color_hs` topic and transforming them into `hs_color` instructions via Home Assistant.

Each automation guarantees that the correct device is updated solely when the knob is in the designated mode. This conditional gating facilitates multi-modal control through a singular physical interface. An thorough compilation of all automation and MQTT sensor definitions is provided in the Appendix A.12.

4.5. System Testing Scenarios

The Smart Knob system's end-to-end functionality was verified through a series of real-world test scenarios. The purpose of these tests was to confirm that a physical interaction with the knob would reliably cause Home Assistant to perform the desired action. This was achieved by verifying the entire communication and automation process by monitoring the device's own serial output and the pertinent logs in Home Assistant. In each scenario, the knob was operated while watching the Home Assistant Logbook for incoming state changes and service calls and the Arduino Serial Monitor for outgoing MQTT messages.

Scenario 1: Adjusting Light Brightness

The continuous control model for brightness modification is validated by this test.

- **Objective:** To determine whether rotating the knob in "Light Brightness" mode correctly changes the brightness of a target Home Assistant light.
- **Procedure:**
  1. The knob was set to the "Light Brightness" mode.
  2. The knob was rotated until approximately 50% appeared on the display.
  3. After that, the knob was turned until 100% appeared on the screen.
- **Results:** The test produced favorable findings. The publication of the correct brightness value to the `smartknob/light/brightness` topic was confirmed by the device's serial output, as shown in Table 4.1. At the same time, the Home Assistant Logbook confirmed the appropriate automation was triggered, calling the `light.turn_on` service with the correct brightness setting.

Table 4.1: Test Logs for Light Brightness Scenario

Arduino Serial Monitor Output	**Home Assistant Logbook Entry**
`Published Mode: Light Brightness`	`Message 8918 received on smartknob/mode/state at 5:40 PM: Light Brightness`
`Published Brightness: 127`	`Message 8919 received on smartknob/light/brightness at 5:40 PM: 127`
`Published Brightness: 255`	`Message 8920 received on smartknob/light/brightness at 5:41 PM: 255`

Scenario 2: Media Track Change via End-Stop

The event-driven interaction framework for media management is validated by this test.

- **Objective:** To confirm that the "next track" and "previous track" commands are activated when the knob is pushed against its haptic end-stop in "Media Control" mode.
- **Procedure:**
  1. By pressing the knob, it was set to "Media Control" mode.
  2. The "NEXT" command was activated by fully turning the knob to the right.
  3. The "PREVIOUS" command was then activated by fully turning the knob to the left.
- **Results:** The test went well. The serial output of the device confirmed that the correct payloads were published to the `smartknob/media/control` topic. The Home Assistant Logbook confirmed that these messages were received, as shown in Table 4.2.

Table 4.2: Test Logs for Media Track Change Scenario

Arduino Serial Monitor Output	**Home Assistant Logbook Entry**
`Published Mode: Media Control`	`Message 8925 received on smartknob/mode/state at 5:51 PM: Media Control`
`Published Media Control: NEXT`	`Message 8926 received on smartknob/media/control at 5:51 PM: NEXT`
`Published Media Control: PREVIOUS`	`Message 8927 received on smartknob/media/control at 5:52 PM: PREVIOUS`

Summary of Other Scenarios

The same procedure was used to evaluate the remaining modes, and every test passed successfully:

- **Controlling Media Volume:** The knob successfully sent percentage values to the `smartknob/media/volume` topic, which caused Home Assistant to adjust the volume of the target media player appropriately.
- **Selecting Fan Speed:** After snapping into each detent, the knob published the discrete levels (0, 1, 2, 3) to the `smartknob/fan/speed` topic, which updated the corresponding helper entity in Home Assistant.
- **Temperature Adjustment:** The knob successfully transmitted temperature data to the `smartknob/temperature` topic with a precision of 0.5°C and correctly constrained the values within the designated minimum and maximum range.
- **Selecting Light Color:** The knob correctly published the hue and saturation values (e.g., "0,100" for red) to the `smartknob/light/color_hs` topic for each of the 16 detents, and the target light changed color as expected.

These tests successful completion validates that the haptic modes are precisely implemented in the firmware and are seamlessly and reliably integrated with the Home Assistant automation engine.

5. Conclusions

5.1. Obtained Results

5.1.1. General Conclusion

A functional smart knob that acts as a centralized control center for a smart home was successfully designed, built, and verified by the project. The thesis looks at the pervasive issue of user annoyance and cognitive overload brought on by smart home controls being spread across several platforms and applications. The "Smart Knob" goes beyond conventional screen-based and speech engagements with its tactile, user-friendly interface that uses advanced motor control to provide software-defined haptic feedback. The final prototype has a brushless DC (BLDC) motor, a high-resolution magnetic encoder, and a circular color display within a self-sufficient, wall-mountable unit. The knob creates a variety of high-fidelity haptic effects, such as continuous damping, discrete detents, and constrained spring-like boundaries, using Field-Oriented Control (FOC) on a dual-core ESP32 microcontroller. Each effect is tailored for a particular purpose, such as controlling temperature, volume, or lights. Through the MQTT protocol, the system communicates with the Home Assistant automation platform with ease, demonstrating its effectiveness in a real-world smart home scenario.

5.1.2. Fulfillment of Proposed Objectives

The project successfully met every goal set forth at the outset:

- **Create and Build an Autonomous Prototype:** Using a circular color display and a brushless motor for haptic input, a wall-mountable prototype was created. The completed assembly satisfies design requirements and is housed inside a standard surface-mounted electrical box with a specially made 3D-printed enclosure.
- **Use High-Fidelity Haptic Feedback:** To control the BLDC motor, the project used advanced Field-Oriented Control (FOC) techniques. In order to replicate the feel of many physical controllers, this made it easier to incorporate a variety of complex haptic textures, such as discrete detents for fan control, limited detents for temperature, and spring-like end-stops for continuous values.
- **Create an Integrated Graphical and Tangible User Interface:** By coordinating haptic feedback with visual cues on the circular display, a unified user interface was created. In order to provide a consistent user experience, each control mode has its own unique graphical user interface (GUI) that displays relevant values and states and accurately reflects the motor's physical feedback.
- **Establish Robust Wireless Smart Home Integration:** Wi-Fi and the MQTT protocol were used to successfully integrate the smart knob into a Home Assistant environment.

In order to enable a decoupled yet reliable architecture for controlling a variety of smart home devices, including lights, fans, and media players, the firmware was designed to broadcast state changes to specific MQTT topics.

- **Complete System Validation and Testing:** The system was thoroughly tested using a variety of real-world use-case scenarios. From physical knob interaction to the efficient execution of commands in Home Assistant, the tests validated the system's dependability, integration resilience, and feedback responsiveness.

5.1.3. List of Contributions

Personal contributions included in this thesis include:

- **System Integration:** A comprehensive combination of hardware and software components, including a circular TFT display, an ESP32 microcontroller, a magnetic rotary encoder, and a BLDC gimbal motor with a hollow shaft, that are assembled into a functional prototype.
- **Advanced Haptic Model Implementation:** Using Field-Oriented Control (FOC), a sophisticated haptic feedback system is designed and implemented. This required creating a number of distinct, software-defined haptic modes (continuous, discrete, bounded, and cyclic) that were tailored for different scenarios involving smart home management.
- **Virtual Angle and Mode-Switching Logic:** Development of a new "virtual angle" system that distinguishes between the knob's actual location and the value it represents. This makes it easier to switch between multiple control modes without unintentionally moving or losing the previous state for each mode.
- **Modular and Real-Time Firmware Architecture:** A dual-core, modular firmware architecture that isolates application and network logic from time-sensitive motor control functions has been developed. This ensures jitter-free instantaneous haptic feedback, which is crucial for a reliable tactile interface.
- **Home Assistant Integration Framework:** Using MQTT sensors and automations, a modular and scalable integration with Home Assistant is developed. This includes a complete setup for linking knob states to Home Assistant objects and triggering actions on various smart devices.
- **Custom mechanical design:** This process involves designing and 3D printing a custom enclosure and mechanical assembly that allows the display cable to pass through the hollow shaft of the motor, allowing for unrestricted 360-degree rotation, while producing an end product that is both aesthetically pleasing and easy to use.

5.2. Future Development Directions

Several possible development paths could be investigated to enhance the Smart Knob's capabilities in light of the project's outcomes and limitations:

- **Dynamic Mode Configuration:** Rather than depending on a set number of modes at build time, future versions might allow users to dynamically add, remove, or reconfigure

control modes through a web interface or Home Assistant integration. This would greatly increase user customization and adaptability.

- **Advanced Network Configuration:** Wi-Fi and MQTT credentials are hardcoded into the current firmware during compilation. A "provisioning" mode (for example, creating a temporary Wi-Fi access point) might be included in a later version to allow users to adjust network settings from a computer or smartphone without having to flash the firmware again.
- **Bi-Directional Communication and State Synchronization:** While the knob currently communicates its current state to Home Assistant, bi-directional communication would be a more complete solution. This would ensure that the knob's display and tactile position always reflect the device's actual state by allowing Home Assistant to update the knob's status (for example, if a light is turned on via a different interface).
- **Hardware and Power System Optimization:** The prototype currently uses standard development boards and a breadboard that are enclosed in an electrical enclosure. Building a custom Printed Circuit Board (PCB) to combine all electronic components is one possible approach. This would simplify assembly, improve dependability, and reduce overall dimensions.
- **Pressure-Sensitive Input:** Incorporate a force-sensing resistor or alternative pressure sensors beneath the knob assembly. This would provide a novel interaction paradigm wherein the knob might react variably depending on the pressure exerted by the user. A gentle press may modify a parameter gradually, whereas a hard press could expedite the rate of change, introducing an additional dimension of intuitive control.

Bibliography

[1] O. Shaer, E. Hornecker *et al.*, "Tangible user interfaces: past, present, and future directions," *Foundations and Trends® in Human–Computer Interaction*, vol. 3, no. 1–2, pp. 4–137, 2010.

[2] M. Traviss. (2024) Evaluating usability issues with AI-assisted smart speakers. Innovation News Network. [Online]. Available: https://www.innovationnewsnetwork.com/evaluating-usability-issues-with-ai-assisted-smart-speakers/49288/

[3] PricewaterhouseCoopers LLP. (2018) Prepare for the voice revolution: The impact of voice assistants on consumer behavior. PwC Consumer Intelligence Series. [Online]. Available: https://www.pwc.com/us/en/services/consulting/library/consumer-intelligence-series/voice-assistants.html

[4] L. Angelini, E. Mugellini, O. Abou Khaled, and N. Couture, "Internet of tangible things (iott): Challenges and opportunities for tangible interaction with iot," in *Informatics*, vol. 5, no. 1. MDPI, 2018, p. 7.

[5] G. Frédéric, M. Amberg, and B. Lemaire-Semail, "Design and control of a haptic knob," *Sensors and Actuators A: Physical*, vol. 196, p. 78–85, 07 2013.

[6] J. Ding, X. Fu, T. Wei, and A. Perzylo, "A cost-efficient foc-controlled haptic knob for industrial robot programming with force feedback," in *2024 IEEE 22nd International Conference on Industrial Informatics (INDIN)*. IEEE, 2024, pp. 1–7.

[7] G. Zhang, G. Sun, Z. Sun, J. Tang, L. Jiang, and R. Liang, "Towards better utilization of haptic interaction in visualization: Design space and knob prototype," *International Journal of Human–Computer Interaction*, vol. 41, no. 9, pp. 5384–5405, 2025.

[8] S. Garg, J. M. Chatterjee, and R. KumarAgrawal, "Design of a simple gas knob: An application of iot," in *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*. IEEE, 2018, pp. 1–3.

[9] Y. Yang, M. Ma, T. V. Doan, T. Hulin, H. P. Frank Fitzek, and G. T. Nguyen, "Touch the metaverse: Demonstration of haptic feedback in network-assisted augmented reality," in *2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2024, pp. 379–381.

[10] D. D. Kumar, M. Rambabu, B. Yogesh, and G. Likhitha, "Control techniques for bldc motor: A review," *International Journal of Research Publication and Reviews*, vol. 3, no. 11, pp. 294–302, November 2022. [Online]. Available: http://www.ijrpr.com

[11] V. Bist, "Field oriented control (foc) made easy for brushless dc (bldc) motors using ti smart gate drivers," *Application Brief*, 2021.

[12] J. C. Gamazo-Real, E. Vázquez-Sánchez, and J. Gómez-Gil, "Position and speed control of brushless dc motors using sensorless techniques and application trends," *sensors*, vol. 10, no. 7, pp. 6901–6947, 2010.

[13] M. HARSHAVARDHAN, "Field oriented control of a brushless dc motor," Ph.D. dissertation, INDIAN INSTITUTE OF TECHNOLOGY MADRAS, 2017.

[14] N. P. Quang, "50 years field oriented control of three-phase ac drives in the practice-the state-of-the-art," *Journal of Computer Science and Cybernetics*, vol. 40, no. 1, pp. 1–22, 2024.

[15] E. Bekiroglu and A. Dalkin, "Comparison of trapezoidal and sinusoidal pwm techniques for speed and position control of pmsm," *Advances in Electrical and Electronic Engineering*, vol. 18, no. 4, p. 207, 2020.

[16] M. A. A. Hassan, A. R. Abdullah, N. Bahari, and M. I. M. Sabri, "Efficiency comparison of trapezoidal and sinusoidal method for brushless dc motor drive," *Applied Mechanics and Materials*, vol. 785, pp. 248–252, 2015.

[17] M. Mahmud, M. R. Islam, S. Motakabber, M. D. A. Satter, K. E. Afroz, and A. A. Habib, "Control speed of bldc motor using pid," in *2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA)*. IEEE, 2022, pp. 150–154.

[18] M. Mahmud, S. Motakabber, A. Z. Alam, and A. N. Nordin, "Control bldc motor speed using pid controller," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 3, 2020.

[19] M. A. W. Begh and H.-G. Herzog, "Comparison of field oriented control and direct torque control," *Authorea Preprints*, 2018.

[20] Espressif Systems, *Wi-Fi - ESP32 — ESP-IDF Programming Guide v5.4.2*, https://docs.espressif.com/projects/esp-idf/en/v5.4.2/esp32/api-guides/wifi.html, Espressif Systems, 2025, accessed: 2025-07-06.

[21] I. Froiz-Míguez, T. M. Fernández-Caramés, P. Fraga-Lamas, and L. Castedo, "Design, implementation and practical evaluation of an iot home automation system for fog computing applications based on mqtt and zigbee-wifi sensor nodes," *Sensors*, vol. 18, no. 8, p. 2660, 2018.

[22] L. Gao and G. Li, "Design and implementation of smart home control system based on stm32," in *INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS, NETWORKING AND APPLICATIONS*. Springer, 2023, pp. 112–125.

A. Relevant Code sections

A.1. Communication and Reconnection Logic

```
// From network.cpp

void reconnect_mqtt() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect("SmartKnobClient", MQTT_USER, MQTT_PASSWORD))
        {
            Serial.println("connected");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}

void network_init() {
    Serial.print("Connecting to ");
    Serial.println(WIFI_SSID);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nWiFi connected.");

    espClient.setInsecure();
    client.setServer(MQTT_SERVER, MQTT_PORT);
    Serial.println("Networking Initialized.");
}

void network_update() {
    if (!client.connected()) {
        reconnect_mqtt();
```

```
    }
    client.loop();

    // ... (rest of the publishing logic) ...
}
```

A.2. Haptic Logic and Virtual Angle Implementation

```
// From motor.cpp

// Haptic logic based on the current virtual angle
void motor_update() {
    motor.loopFOC();

    float physical = motor.shaft_angle;
    currentVirtualAngle = physical + modeOffset[current_mode];

    float angle_rad = currentVirtualAngle;
    float angle_deg = rad2deg(angle_rad);
    float voltage = 0.0f;

    switch (current_mode) {
        // ... (case statements for each mode as below) ...
    }
    motor.move(voltage);
    modeMemory[current_mode] = currentVirtualAngle;
}

// Mode switching logic calculating the offset
void motor_set_mode(KnobMode new_mode, bool force) {
    if (!force && new_mode == current_mode) return;

    modeMemory[current_mode] = currentVirtualAngle;

    float stored = modeMemory[new_mode];
    float physical = motor.shaft_angle;

    if (stored == 0.0f && new_mode != MODE_LIGHT_BRIGHTNESS) {
        stored = default_memory_for_mode(new_mode, physical);
    }

    stored = clamp_to_mode(new_mode, stored);
    modeMemory[new_mode] = stored;
```

```
    modeOffset[new_mode] = stored - physical;
    current_mode = new_mode;

    // ... (PID profile switching logic) ...
}
```

A.3. Haptic Logic for Continuous Modes

```
// From motor_update() in motor.cpp
case MODE_LIGHT_BRIGHTNESS:
case MODE_MEDIA_VOLUME: {
    if (angle_deg > ANALOG_MODE_MAX_ANGLE) {
        current_attractor_angle = deg2rad(ANALOG_MODE_MAX_ANGLE);
        voltage = haptic_pid(current_attractor_angle - angle_rad);
    } else if (angle_deg < 0) {
        current_attractor_angle = 0;
        voltage = haptic_pid(current_attractor_angle - angle_rad);
    } else {
        voltage = -HAPTIC_KD_DAMPING * motor.shaft_velocity;
    }
    break;
}
case MODE_MEDIA_CONTROL: {
    float min_r = deg2rad(TRACK_MODE_MIN_ANGLE);
            float max_r = deg2rad(TRACK_MODE_MAX_ANGLE);

            if (angle_rad > max_r) {
                current_attractor_angle = max_r;
                voltage = haptic_pid(current_attractor_angle - angle_rad);
                if (last_media_action != 1) {
                    network_publish_media_control("NEXT");
                    last_media_action = 1;
                }
            } else if (angle_rad < min_r) {
                current_attractor_angle = min_r;
                voltage = haptic_pid(current_attractor_angle - angle_rad);
                if (last_media_action != -1) {
                    network_publish_media_control("PREVIOUS");
                    last_media_action = -1;
                }
            } else {
                voltage = -HAPTIC_KD_DAMPING * motor.shaft_velocity;
```

```
                last_media_action = 0;
            }
            break;
}
```

A.4. Haptic Logic for Fan Speed Mode

```
// From motor_update() in motor.cpp
case MODE_FAN_SPEED: {
    current_attractor_angle = //
    findDetent(angle_rad, deg2rad(FAN_DETENT_ANGLE));
    voltage = haptic_pid(current_attractor_angle - angle_rad);
    break;
}
```

A.5. Haptic Logic for Temperature Control Mode

```
// From motor_update() in motor.cpp
case MODE_TEMPERATURE_CONTROL: {
    float min_r = deg2rad(TEMP_MIN_ANGLE);
    float max_r = deg2rad(TEMP_MAX_ANGLE);
    if (angle_rad > max_r) {
        // ... end-stop logic ...
    } else if (angle_rad < min_r) {
        // ... end-stop logic ...
    } else {
        float relative = angle_rad - min_r;
        float detent_w = deg2rad(TEMP_DETENT_ANGLE);
        current_attractor_angle = \\
        findDetent(relative, detent_w) + min_r;
        voltage = haptic_pid(current_attractor_angle - angle_rad);
    }
    break;
}
```

A.6. Haptic Logic for Color Picker Mode

```
// From motor_update() in motor.cpp
case MODE_LIGHT_COLOR: {
    float wrapped = fmodf(angle_rad, 2 * PI);
```

```
    if (wrapped < 0) wrapped += 2 * PI;
    current_attractor_angle = //
      findDetent(wrapped, deg2rad(COLOR_MODE_DETENT_ANGLE));
    voltage = haptic_pid(current_attractor_angle - wrapped);

    int spinner_pos = //
      roundf(current_attractor_angle / deg2rad(COLOR_MODE_DETENT_ANGLE));
    current_color_index = spinner_pos % COLOR_PALETTE_SIZE;
    break;
}
```

A.7. Display Logic for Brightness and Volume

The following C++ code from `display.cpp` renders the UI for the Brightness and Volume modes.

```
// From display_update() in display.cpp
case MODE_LIGHT_BRIGHTNESS:
case MODE_MEDIA_VOLUME: {
    float displayAngleDeg = constrain(angleDeg, 0, ANALOG_MODE_MAX_ANGLE);
    float percent = (displayAngleDeg / ANALOG_MODE_MAX_ANGLE) * 100.0;
    int bar_height = (percent / 100.0) * sprite.height();
    sprite.fillRect(0, sprite.height() - bar_height, //
      sprite.width(), bar_height, TFT_ORANGE);

    sprite.setTextDatum(MC_DATUM);
    sprite.setTextSize(3);
    sprite.drawString(String((int)percent) + "%", centerX, centerY);
    break;
}
```

A.8. Display Logic for Fan Speed

```
// From display_update() in display.cpp
case MODE_FAN_SPEED: {
    // Background dots for detents
    int radius = 100;
    for (int i = 0; i < 4; i++) {
        float detentRad = i * FAN_DETENT_ANGLE * DEG_TO_RAD;
        sprite.fillCircle(centerX + cos(detentRad) * radius, //
        centerY - sin(detentRad) * radius, 5, TFT_BLUE);
    }
```

```
    // Central text for speed level
    float constrainedAngle = constrain(angleDeg, 0, 359);
    int speed_level = round(constrainedAngle / FAN_DETENT_ANGLE);
    sprite.setTextDatum(MC_DATUM);
    sprite.setTextSize(3);
    sprite.setTextColor(TFT_GREEN);
    sprite.drawString(String(speed_level), centerX, centerY);
    break;
}
```

A.9. Display Logic for Temperature Control

```
// From display_update() in display.cpp
case MODE_TEMPERATURE_CONTROL: {
    // Background arc
    sprite.drawArc(centerX, centerY, 105, 102, TEMP_MIN_ANGLE, //
      TEMP_MAX_ANGLE, TFT_DARKGREY, TFT_BLACK, false);
    float constrained_angle = //
      constrain(angleDeg, TEMP_MIN_ANGLE, TEMP_MAX_ANGLE);
    if (constrained_angle > TEMP_MIN_ANGLE) {
        sprite.drawArc(centerX, centerY, 105, 102, TEMP_MIN_ANGLE, //
          constrained_angle, TFT_CYAN, TFT_BLACK, false);
    }

    // Central text for temperature
    float range_degrees = TEMP_MAX_ANGLE - TEMP_MIN_ANGLE;
    float range_celsius = TEMP_CELSIUS_MAX - TEMP_CELSIUS_MIN;
    float temp = TEMP_CELSIUS_MIN + ((angleDeg - TEMP_MIN_ANGLE) *  //
      range_celsius / range_degrees);
    float rounded_temp = round(temp * 2.0) / 2.0;
    rounded_temp = constrain(rounded_temp, TEMP_CELSIUS_MIN, //
      TEMP_CELSIUS_MAX);
    sprite.setTextDatum(MC_DATUM);
    sprite.setTextSize(3);
    sprite.setTextColor(TFT_CYAN);
    sprite.drawString(String(rounded_temp, 1) + " C", centerX, centerY);
    break;
}
```

A.10. Display Logic for Media Control

```
// From display_update() in display.cpp
case MODE_MEDIA_CONTROL: {
    // Draw static arrows
    sprite.drawTriangle(centerX - 50, centerY, centerX - 70, //
      centerY - 15, centerX - 70, centerY + 15, TFT_DARKGREY);
    sprite.drawTriangle(centerX + 50, centerY, centerX + 70, //
      centerY - 15, centerX + 70, centerY + 15, TFT_DARKGREY);

    // Fill triangle if at end-stop
    if (angleDeg <= TRACK_MODE_MIN_ANGLE) {
        sprite.fillTriangle(centerX - 50, centerY, centerX - 70, centerY -
15, centerX - 70, centerY + 15, TFT_CYAN);
    }
    if (angleDeg >= TRACK_MODE_MAX_ANGLE) {
        sprite.fillTriangle(centerX + 50, centerY, centerX + 70, centerY -
15, centerX + 70, centerY + 15, TFT_CYAN);
    }

    // Draw moving dot
    float needle_pos_x = map(angleDeg, TRACK_MODE_MIN_ANGLE, //
      TRACK_MODE_MAX_ANGLE, centerX - 30, centerX + 30);
    needle_pos_x = constrain(needle_pos_x, centerX - 30, centerX + 30);
    sprite.fillCircle(needle_pos_x, centerY, 5, TFT_WHITE);
    break;
}
```

A.11. Display Logic for Color Picker

```
// From display_update() in display.cpp
case MODE_LIGHT_COLOR: {
    int selected_index = motor_get_color_index();
    int cols = 4, rows = 4, box_size = 40, padding = 10;
    int start_x = centerX - (cols * (box_size + padding) - padding) / 2;
    int start_y = centerY - (rows * (box_size + padding) - padding) / 2;

    for (int i = 0; i < COLOR_PALETTE_SIZE; i++) {
        int col = i % cols, row = i / cols;
        int box_x = start_x + col * (box_size + padding);
        int box_y = start_y + row * (box_size + padding);
        sprite.fillRect(box_x, box_y, box_size, box_size, //
```

```
          color_palette[i].tft_color);

        // Draw highlight rectangle if selected
        if (i == selected_index) {
            sprite.drawRect(box_x - 2, box_y - 2, box_size + 4, //
              box_size + 4, TFT_WHITE);
        }
    }
    break;
}
```

A.12. Home Assistant Configuration

A.12.1. MQTT Sensor Configuration

```
# From configuration.yaml
mqtt:
  sensor:
    - name: "SmartKnob Mode"
      state_topic: "smartknob/mode/state"
      icon: mdi:sync

    - name: "SmartKnob Brightness Value"
      state_topic: "smartknob/light/brightness"
      unit_of_measurement: " "

    - name: "SmartKnob Volume Value"
      state_topic: "smartknob/media/volume"
      unit_of_measurement: "%"

    # ... other sensors ...

    - name: "SmartKnob Light Color HS"
      state_topic: "smartknob/light/color_hs"
      icon: mdi:palette
```

A.12.2. Automation Configuration

```
# From automations.yaml

# Automation for controlling brightness
- alias: "Sync Simulated Brightness to Living Room Light"
  id: sync_simulated_brightness_to_living_room_light
```

```
  trigger:
    - platform: state
      entity_id: sensor.smartknob_brightness_value
  condition:
    - condition: state
      entity_id: light.light # Target light entity
      state: 'on'
  action:
    - service: light.turn_on
      target:
        entity_id: light.light # Target light entity
      data:
        brightness: "{{ trigger.to_state.state | int(0) }}"

# Automation for controlling media player
- alias: "SmartKnob Skip Media Track"
  id: smartknob_skip_media_track
  trigger:
    - platform: mqtt
      topic: "smartknob/media/control"

    action:
    - choose:
        - conditions:
            - condition: template
              value_template: "{{ trigger.payload == 'NEXT' }}"
          sequence:
            - service: media_player.media_next_track
              target:
                entity_id: media_player.office_speaker
        - conditions:
            - condition: template
              value_template: "{{ trigger.payload == 'PREVIOUS' }}"
          sequence:
            - service: media_player.media_previous_track
              target:
                entity_id: media_player.office_speaker
```