

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

IoT-Enabled Smart Medication System for Elderly Care with Bioengineering Integration

David Oliveira Rodrigues



Master's in Bioengineering

Supervisor: Prof. Vitor Hugo Pinto

July 31, 2026

Resumo

A baixa adesão terapêutica em idosos que vivem de forma independente constitui um risco clínico significativo e uma fonte substancial de despesa evitável para o sistema de saúde. As soluções domésticas atuais, sobretudo os organizadores semanais de comprimidos, apresentam problemas práticos conhecidos, como embalagens de blister difíceis de remover e rótulos de leitura difícil. Esta dissertação apresenta o desenho, a implementação e a avaliação experimental do MedBox, um dispensador automático de medicação conectado que combina dispensa automática unitária, confirmação física da queda e da recolha do medicamento, e monitorização e alerta remotos dirigidos ao cuidador. Os requisitos do sistema foram fundamentados em entrevistas semiestruturadas a duas farmacêuticas comunitárias, com perspetivas urbana e rural complementares, que informaram diretamente decisões como a organização do dispensador por tipo de medicamento, a janela de tolerância de recolha configurável com escalonamento sonoro progressivo do alarme, e a identificação de um alçapão de segurança como funcionalidade prioritária. O mecanismo de dispensa adota um princípio de singulação por cilindro rotativo de fenda única, adaptado a uma implementação imprimível por FDM, construída em torno de um microcontrolador ESP32-C6, uma célula de carga HX711 para confirmação de queda e recolha, um sensor ambiental DHT11, e um relógio de tempo real DS3231. O sistema segue uma arquitetura de quatro camadas desacopladas, com uma Firebase Realtime Database como ponto central de sincronização entre o firmware, uma aplicação móvel em Flutter, e um servidor Node.js de vigilância responsável pelas notificações *push*. A validação experimental, conduzida ao longo de cinco fases de teste na montagem final de três compartimentos, confirmou o conceito mecânico central: o cilindro rotativo dispensa de forma fiável uma única unidade por ciclo, com 100% de sucesso na rotação mecânica nos três compartimentos. A taxa de aterragem no tabuleiro de recolha variou entre 66% e 96% consoante o peso do comprimido testado, sendo mais elevada para a unidade mais pesada. A aplicação, o modelo de dados Firebase, a resiliência *offline*, e o *pipeline* de notificações foram validados de ponta a ponta na montagem final com medicamentos reais, passando todas as dezoito verificações funcionais e os sete cenários de caso-limite testados. A fixação rígida e aparafusada da célula de carga, adotada na montagem final, garantiu uma deteção de peso estável, com um coeficiente de variação inferior a 1% para os medicamentos testados. A caracterização do comportamento de encravamento em cada compartimento revelou uma dependência clara da geometria da fenda e do modo de assentamento do comprimido no reservatório. Este protótipo contribui com uma alternativa aberta e mecanicamente robusta aos mecanismos de singulação proprietários encontrados nos dispensadores automáticos de comprimidos atuais, complementada por uma arquitetura de *software* resiliente *offline*, validada em três compartimentos com dois medicamentos reais. A metodologia experimental é documentada extensivamente, associando cada decisão de desenho aos dados que a motivaram, demonstrando um sistema funcional, testado de ponta a ponta.

Abstract

Poor medication adherence among elderly patients living independently constitutes a significant clinical risk and a substantial source of avoidable expenditure for the healthcare system. Current home-based solutions, particularly weekly pill organisers, present well-known practical problems, ranging from blister packaging that is difficult to remove to labels that are hard to read. This dissertation presents the design, implementation, and experimental evaluation of MedBox, a connected automatic medication dispenser that combines automated single-unit dispensing, physical confirmation of both drop and collection of the medication, and remote monitoring and alerting directed at the caregiver. The system requirements were grounded in semi-structured interviews with two community pharmacists, offering complementary urban and rural perspectives, which directly informed design decisions such as the organisation of the dispenser by medication type, the configurable collection tolerance window with progressive acoustic alarm escalation, and the identification of a safety hatch as a priority feature. The dispensing mechanism adopts a single-slot rotating-cylinder singulation principle, adapted into an FDM-printable implementation built around an ESP32-C6 microcontroller, an HX711 load cell for drop and collection confirmation, a DHT11 environmental sensor, and a DS3231 real-time clock. The system follows a four-layer decoupled architecture, with a Firebase Realtime Database as the central synchronisation point between the firmware, a Flutter mobile application, and a Node.js watchdog server responsible for push notifications. The experimental validation, conducted across five testing phases on the final, three-compartment assembly, confirmed the core mechanical concept: the rotating cylinder reliably dispenses a single unit per cycle, achieving 100% mechanical rotation success across all three compartments. The landing rate inside the collection tray ranged from 66% to 96% depending on the weight of the tablet tested, highest for the heaviest unit. The application, the Firebase data model, offline resilience, and the notification pipeline were validated end-to-end on the final assembly with real medications, passing all eighteen functional checks and all seven edge case scenarios tested. The rigid, screw-mounted load cell fixation adopted for the final assembly ensured stable weight detection, with a coefficient of variation below 1% for the medications tested. The characterisation of jamming behaviour across compartments revealed a clear dependence on slot geometry and on how the tablet settles within the reservoir. This prototype contributes an open and mechanically robust alternative to the proprietary singulation mechanisms found in current automatic pill dispensers, complemented by an offline-resilient software architecture, validated across three compartments with two real medications. The experimental methodology is documented extensively, linking each design decision to the data that motivated it, demonstrating a functional system, tested end-to-end.

Acknowledgements

I would like to begin by thanking my supervisor, Prof. Vitor Hugo Pinto, for his guidance, availability, and the critical input that shaped this dissertation from its earliest stages through to its completion.

I am also deeply grateful to Diogo Gomes, whose advice on the mechanical design of the dispensing mechanism was essential to this work. His willingness to discuss ideas, point out flaws in early approaches, and suggest the rotating-cylinder principle that ultimately shaped the prototype made a difference I could not have achieved on my own.

I would also like to thank Ema Paulino, President of Associação Nacional das Farmácias, and Isabel Luz, Technical Director of Farmácia Rainha do Douro, for generously sharing their time and clinical experience. Their complementary urban and rural perspectives were invaluable in ensuring that this work remained anchored to the needs of the patients it aims to serve.

To my friends, specially the Fora Dagua group, thank you for your patience, encouragement, and the moments of distraction that made this journey easier to bear.

To Sofia, my girlfriend, thank you for the support you have given me, even having only known me for part of this journey. Your presence made a real difference during the most difficult moments, and I am deeply grateful for it.

Last but not least, I want to express my gratitude to my family. In particular, to my parents, Isabel and Daniel, and to my siblings, Leonor and Tomás, thank you for your unconditional support throughout this entire process, and for everything you have given me along the way.

David Oliveira Rodrigues

“What man actually needs is not a tensionless state but rather the striving and struggling for a worthwhile goal, a freely chosen task.”

Viktor E. Frankl

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem Statement and Objectives	2
1.3	Alignment with the United Nations Sustainable Development Goals	3
1.4	Structure of the Dissertation	5
2	State of the Art	6
2.1	The Clinical and Demographic Landscape of Elderly Medication Management . .	6
2.1.1	The Global Demographic Transition and the Emergence of Population Ageing	6
2.1.2	The Epidemiological Pivot: Multimorbidity and the Healthcare Utilisation Crisis	8
2.1.3	Polypharmacy: Prevalence, Determinants, and Clinical Projections	8
2.1.4	Medication Self-Administration Errors: A Hidden Clinical and Economic Burden	10
2.1.5	The Portuguese Context	11
2.1.6	Technology Adoption Barriers in the Elderly: The Digital Divide	12
2.1.7	Summary: The Clinical and Social Case for Automated Medication Management	16
2.2	IoT Architectures for Home Healthcare Applications	16
2.2.1	Communication Technologies: WiFi, Bluetooth, and Cellular	16
2.2.2	Edge Computing and Offline Resilience	17
2.2.3	Cloud Platforms and Real-Time Data Management	18
2.3	Smart Medication Dispensers: An Analytical Survey	18
2.3.1	Device Taxonomy: Storage Regimes and Dispensing Mechanisms	19
2.3.2	Dose Validation Mechanisms: The Capability Ladder	20
2.3.3	Alert Management and Caregiver Notification Architectures	21
2.3.4	Active Fail-Safes and Physical Responses to Non-Adherence	22
2.3.5	Usability and User Acceptance Among Elderly Patients	22
2.3.6	The Portuguese Precedent: The Balvia / SelfMed Ecosystem	23
2.3.7	Synthesis: The Technology-Clinical Gap	24
2.4	Summary and Positioning of the Present Work	25
3	Experimental Development	26
3.1	Requirements Elicitation: Interviews with Pharmacists	27
3.1.1	Compartmentalisation Strategy	28
3.1.2	Scale of Polypharmacy and Number of Compartments	28
3.1.3	Dispensing Mechanism and Fail-Safes	28

3.1.4	Alarm Behaviour and Escalation	29
3.1.5	Storage Conditions	29
3.1.6	Limitations of Current Weekly Pillbox Solutions	29
3.1.7	Choice of Test Medication	30
3.1.8	Personalisation of Forgetfulness-Mitigation Strategies	30
3.1.9	Therapeutic Adherence as a Need Statement	31
3.2	System Architecture	31
3.2.1	System Overview	31
3.2.2	User Profiles and Use Cases	34
3.2.3	System Requirements	36
3.2.4	Summary	38
3.3	Mechanical Development	39
3.3.1	Physical Requirements of the Dispensing Mechanism	39
3.3.2	Approaches Considered	39
3.3.3	Compartmentalisation Strategy: Mechanical Implications	43
3.3.4	Scaling to Additional Compartments: Fixed versus Medication-Dependent Parameters	45
3.3.5	Final Mechanical Design	47
3.3.6	3D Printing Process and Setbacks	53
3.3.7	3D-Printed Pills as a Test Proxy	54
3.3.8	Surface Treatment: Rationale for Omitting Resin Coating	55
3.3.9	Three-Compartment Implementation	55
3.4	Electronics and Firmware	56
3.4.1	Selection and Justification of Components	56
3.4.2	Firmware-Level Component Description	58
3.4.3	Firmware Development Phases	64
3.4.4	Firmware Architecture	65
3.4.5	Firestore Realtime Database Integration	67
3.4.6	Challenges and Solutions in Firmware Development	68
3.4.7	Components Considered but Not Implemented	70
3.5	Mobile Application and Backend	70
3.5.1	Mobile Application Structure	70
3.5.2	Medication Catalogue Database	74
3.5.3	Recurring Dose Management and Database Indexing	75
3.5.4	Notification Architecture	75
4	Testing, Results and Discussion	77
4.1	Test Plan	77
4.2	Test Environment	78
4.2.1	Historical Note: Early Platform Iterations	79
4.3	Phase 1: Servo Rotation Test — Empty Cylinder	80
4.4	Phase 2: Load Cell Calibration and Validation	81
4.4.1	Rationale and Protocol	81
4.4.2	Results	83
4.5	Phase 3: Dispensing Test, Landing Rate and Detection Stability	84
4.5.1	Protocol	84
4.5.2	Results	85
4.5.3	Detection Stability Under Real Dispensing Conditions	86
4.6	Phase 4: Jamming Characterisation and Mitigation Attempt	87

4.6.1	Observed Cause of Jamming	87
4.6.2	Mitigation Attempt	88
4.6.3	Protocol	88
4.6.4	Results	89
4.7	Phase 5: End-to-End System Test	92
4.7.1	Preliminary Bench Validation (Coin, App and Firmware Logic Only) . .	92
4.7.2	Final Validation (Real Medications, Three Compartments)	93
4.8	Discussion	98
4.8.1	Mechanical Validation	98
4.8.2	Multi-Pill Dispensing and Shared Drop Path	99
4.8.3	System-Level Assessment	100
4.9	Limitations	100
5	Future Work and Conclusions	103
5.1	Future Work	103
5.1.1	Mechanical	103
5.1.2	Electronics and Firmware	104
5.1.3	Software and Notifications	105
5.1.4	Clinical and Usability Validation	106
5.2	Conclusions	106
	References	108
A	Detailed Use Case Descriptions	114
B	System Requirements	117
C	SolidWorks Technical Drawings	119
C.1	Rotating Cylinder	119
C.2	Shaft	121
C.3	Body	123
C.4	Cap	125
C.5	Upper Compartment Support	127
C.6	Main Tunnel	129
C.7	Secondary Collector Funnel	131
C.8	Lower Support and Load Cell Mounting	133
C.9	Load Cell Platform	135
D	Firmware Code and Test Logs	137
D.1	Load Cell Calibration Sketch	137
D.2	Dispensing Weight Test Sketch	142
D.3	Jamming Mitigation Test Sketch	144
D.4	Final Production Firmware	146
D.5	Example Drop Detection Log (Ben-u-ron 1000 mg)	166
D.6	End-to-End Grouped Dispensing Log	167

List of Figures

2.1	MedReady rotary carousel dispenser [1].	19
2.2	LiveFine 28 automated pill dispenser [2].	19
2.3	MedMinder connected pill dispenser [3].	20
2.4	Evondos pouch-roll medicine dispenser [4].	20
2.5	Hero bulk-canister dispenser [5].	20
2.6	MedaCube multi-bin automatic dispenser [6].	20
3.1	The final, three-compartment MedBox prototype: rotating-cylinder dispensing mechanism, shared collection platform with load cell, and control electronics. . .	27
3.2	Example of a weekly pillbox organiser of the type currently used to prepare medication for elderly patients, with small individually-labelled daily compartments [71].	30
3.3	<i>Overview of the layered architecture of the MedBox system: Hardware (Edge), Persistence and Communication (Cloud), Application (Client), and Service (Back-end).</i>	32
3.4	<i>End-to-end data flow of the MedBox system, from schedule configuration by the caregiver to the dispensing event and the resulting push notification in case of an anomaly. Numbered steps are described in the accompanying text. Dashed arrows represent one-off reads rather than persistent listeners.</i>	33
3.5	Use case diagram of the MedBox system. The dashed arrow indicates that an anticipated dispensing request (UC4) directly triggers the dispensing routine (UC6), bypassing the regular schedule.	36
3.6	The eighteen functional and non-functional requirements of the MedBox system, grouped into eight thematic categories.	37
3.7	Early concept sketch of the flap-hatch dispensing mechanism, made during the initial ideation phase.	40
3.8	Informal design sketch of the operating principle of the rotating cylinder mechanism: in its resting position, the cylinder's slot is aligned with the funnel and receives a medication unit from the reservoir; the servo then rotates the cylinder by approximately 180°, carrying the unit to the bottom of the housing where it is released onto the collection platform, before the cylinder returns to its resting position.	42
3.9	The rotating cylinder of the original compartment, shown here sized for the 3D-printed pill substitute: the slot (<i>fenda</i>) sized for one medication unit, and the lateral cut matching the servo horn profile, are both visible. (a) SolidWorks design and (b) the corresponding 3D-printed part.	48

3.10	Side view of the SG90 servo motor used to actuate the rotating cylinder. The distance from the body of the servo to the tip of the horn, visible here, is the dimension that had to be accounted for when sizing the thickness of the body's rear wall.	49
3.11	The body of the dispensing mechanism, with the asymmetric funnel embedded in its top face: (a) SolidWorks design and (b) the corresponding 3D-printed part. . .	50
3.12	The main tunnel and secondary collector funnel: (a) SolidWorks assembly showing only the main tunnel, the secondary collector funnel, and the upper compartment supports, without the lower supports or pillars described below; (b) a close-up photograph of the fully assembled prototype, showing the same funnels in their final position.	51
3.13	The lower support piece below Compartment 1, showing the groove into which one end of the load cell is screwed, the only point of contact between the load cell and this fixed part of the structure: (a) SolidWorks design and (b) the corresponding 3D-printed part.	52
3.14	The final load cell platform, onto which medication units fall regardless of which compartment dispensed them: (a) SolidWorks design and (b) the corresponding 3D-printed part.	53
3.15	Comparison, for reference, between a real paracetamol tablet (Pharmakern) and the 3D-printed pill substitute used for the initial single-compartment tests, shown in the same cylinder slot: (a) the real tablet, and (b) the printed substitute.	54
3.16	Breadboard circuit connecting the ESP32-C6 to the HX711 load cell amplifier, DHT11 sensor, DS3231 RTC, buzzer, and servo motors.	57
3.17	Wiring diagram of the ESP32-C6 and the peripherals listed in Table 3.4, complementing the breadboard photograph of Figure 3.16 with an explicit view of the GPIO connections.	58
3.18	Sequence of firmware steps executed during a grouped dispensing event, from the initial tare to the final collection outcome. The decision point loops back to a fresh baseline reading for each additional unit or medication in the same event; collection is assessed only once, after every unit has been dispensed. Annotated examples are drawn from the three-medication session discussed in Section 4.7.2.	62
3.19	Incremental firmware development phases, each validated independently before integration of the next.	64
3.20	Firmware architecture: the Arduino <code>loop()</code> function drives four independent software timers. When the dispensing check (5 s timer, highlighted) detects a scheduled dose, it triggers <code>dispensarDosesAgrupadas()</code> , which executes synchronously and suspends the remaining background timers for its full duration.	65
3.21	Additional firmware mechanisms, beyond the timer-based polling architecture, described individually below.	66
3.22	Caregiver interface: (a) Firebase Authentication login screen; (b) Agenda tab showing scheduled doses by day with status indicators; (c) dose creation modal with fields for medication, compartment, quantity, time, tolerance window, and day-of-week frequency selection.	72
3.23	Inventory and settings: (a) compartment-to-medication association; (b) Infomed catalogue search using a <code>search_slug</code> prefix query over 6000+ entries; (c) manual configuration of environmental alert thresholds.	73

3.24	Monitoring and history: (a) Monitor tab showing connectivity status, live temperature and humidity readings, next scheduled dose, most recent dispensing outcome, and active alert banners — all updated via <code>onValue</code> RTDB streams; (b) History tab with chronological dispensing log, colour-coded outcome labels, and both the scheduled and actual collection times.	73
4.1	Early platform iterations and their provisional load cell placement: (a) initial flat platform with edge mounted screws; (b) concave platform with centred screws. Both used the same groove and adhesive tape fixation for the load cell.	80
4.2	Empty cylinder rotation test, close-up on the slot area: the servo cylinder assembly performing the full $0^\circ \rightarrow 180^\circ \rightarrow 0^\circ$ cycle after the inner body surface was trimmed to eliminate friction induced binding. (a) Resting position, with the slot visible and aligned with the funnel opening; (b) rotated position, with the slot turned away and hidden inside the housing.	81
4.3	Setup used to weigh individual medication units on the final platform, with the calibration sketch running over serial.	82
4.4	Dispensing test scenario: servo cylinder assembly, funnel, and final platform with rigidly mounted load cell, used across the 50 dispensing cycles of Phase 3.	85
4.5	Desired and undesired orientation of a Ben-u-ron 1000 mg tablet inside the cylinder slot: (a) the tablet lies flat within the slot without obstructing the surrounding cavity; (b) the tablet is wedged against the edge of the slot, preventing the cylinder from completing its rotation and causing a jam.	88
4.6	Funnel loaded with 5 medication units at the start of a jamming trial.	89
4.7	Example of a jam in the Ben-u-ron 1000 mg compartment: a tablet in the undesired orientation, with a second tablet partially occupying the remaining space, blocking rotation.	90
4.8	Ben-u-ron 500 mg compartment: tablets crowding close to the funnel outlet, due to the free space left around the slot by the shared cylinder body and the rounded slot profile matching the tablet's circular geometry.	91
4.9	Preliminary bench setup used to validate the application and firmware logic, with a 1 euro coin used as a stand in reference object.	93
4.10	Final end-to-end test setup, shown from three angles: ESP32-C6 development board with full sensor circuit, final 0.2 g resolution load cell, three compartment dispensing mechanism, and mobile app running on the caregiver's device.	94
4.11	Examples of push notifications received on the caregiver's device via Firebase Cloud Messaging, with the application closed: a device connectivity-loss alert (top) and a temperature-exceeded alert (bottom).	97

List of Tables

1.1	Alignment between the present work and the United Nations Sustainable Development Goals.	4
2.1	Evolution of healthcare utilization indicators (U.S.), 2010–2024. [12]	7
2.2	Impact of polypharmacy and concurrent cognitive decline on clinical outcomes. .	10
2.3	Barrier categories, underlying physiological/cognitive decline, and resulting operational friction in technology use.	13
2.4	Universal Design Principles, description and example in practice [46].	15
2.5	Comparison of wireless communication technologies for home healthcare IoT applications [48].	17
2.6	Visual syndthesis of the technology-clinical gap across representative smart adherence architectures. (✓ Fully Addressed ~ Partially Addressed ✗ Failed / Critical Risk)	24
3.1	Mechanical trade-offs between per-type and per-dose compartmentalisation strategies.	44
3.2	Mechanical design parameters: fixed (medication-independent) versus medication-dependent dimensions.	46
3.3	Tablet dimensions (caliper-measured) and corresponding cylinder cavity dimensions chosen for each compartment.	56
3.4	Pin assignment for the ESP32-C6 in the MedBox prototype.	59
3.5	Firmware constants governing drop and collection detection.	61
3.6	Specification comparison between the DHT11 and DHT22 sensors [74, 75]. . . .	63
3.7	Firebase Realtime Database nodes accessed by the firmware.	68
4.1	Summary of testing phases and objectives.	78
4.2	Load cell calibration, 1 euro coin (7.5 g), final platform, rigid fixation.	83
4.3	Validation against a real medication unit (Ben-u-ron 1000 mg), final platform. . .	83
4.4	Validation against a real medication unit (Ben-u-ron 500 mg), final platform. . . .	84
4.5	Landing rate per compartment, 50 dispensing cycles each.	86
4.6	Position of the first jam (dispense attempt, out of 5 loaded units) across 5 loading trials, per compartment, with the mitigation routine active.	89
4.7	Functional validation results, pass/fail per system feature, final assembly, real medications.	95
4.8	Edge case test scenarios, three compartment prototype.	98
A.1	Detailed description of UC6: Dispense Medication.	115
A.2	Detailed description of UC4: Request Anticipated Dispensing.	116

B.1 Functional requirements of the MedBox system. 117

B.2 Non-functional requirements of the MedBox system. 118

List of Acronyms

ADE	Adverse Drug Event
AIM	<i>Autorização de Introdução no Mercado</i> (Marketing Authorisation)
AAL	Ambient Assisted Living
ANF	<i>Associação Nacional das Farmácias</i> (Portuguese National Pharmacies Association)
API	Application Programming Interface
BLE	Bluetooth Low Energy
CAS	Computer Attitude Scale
DHT11	Digital Humidity and Temperature sensor (model 11)
DHT22	Digital Humidity and Temperature sensor (model 22)
DigComp	Digital Competence Framework
ED	Emergency Department
eHEALS	eHealth Literacy Scale
ESP32-C6	Espressif System-on-Chip, C6 variant (RISC-V, Wi-Fi)
EU	European Union
FCM	Firebase Cloud Messaging
FDA	Food and Drug Administration
FDM	Fused Deposition Modelling
FR	Functional Requirement
GPIO	General-Purpose Input/Output
GSM	Global System for Mobile Communications
HR	Hazard Ratio
HX711	24-bit Analogue-to-Digital Converter for Load Cells (model HX711)
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
INE	<i>Instituto Nacional de Estatística</i> (Statistics Portugal)
Infarmed	<i>Autoridade Nacional do Medicamento e Produtos de Saúde</i> (Portuguese National Authority of Medicines and Health Products)
IRR	Incidence Rate Ratio
ISP	Internet Service Provider
LTE	Long-Term Evolution
MC38	Magnetic Contact reed switch (model MC38)
MDPQ	Mobile Device Proficiency Questionnaire
MQTT	Message Queuing Telemetry Transport
MSE	Medication Self-Administration Error
NASA-TLX	NASA Task Load Index
NCD	Non-Communicable Disease

NFC	Near-Field Communication
NFR	Non-Functional Requirement
NTP	Network Time Protocol
OR	Odds Ratio
PLA	Polylactic Acid
PWM	Pulse-Width Modulation
REST	Representational State Transfer
RFID	Radio-Frequency Identification
RH	Relative Humidity
RISC	Reduced Instruction Set Computer
RTC	Real-Time Clock
RTDB	Realtime Database
SDK	Software Development Kit
SG90	Standard micro servo motor (model SG90)
SHARE	Survey of Health, Ageing, and Retirement in Europe
SIM	Subscriber Identity Module
SUS	System Usability Scale
TAM	Technology Acceptance Model
UC	Use Case

Chapter 1

Introduction

This chapter introduces the present dissertation. It begins by motivating the work in the context of population ageing and the daily challenge of medication management, before formulating the central problem addressed and the specific objectives pursued to answer it. The chapter then positions the contribution of this work within the United Nations Sustainable Development Goals (Section 1.3), and closes by outlining the structure of the remainder of the dissertation.

1.1 Motivation

Population ageing is reshaping the demands placed on healthcare systems worldwide, and nowhere is this more visible than in the daily management of medication. As life expectancy rises and chronic disease becomes the norm rather than the exception among older adults, an increasing share of the population is expected to manage complex, multi-drug regimens largely on their own, at home, without direct clinical supervision. This reality is particularly acute in Portugal, one of the most aged countries in the European Union, where polypharmacy and non-adherence rates among community-dwelling elderly patients are alarmingly high, as detailed in Chapter 2.

The consequences of this gap are not abstract. Forgetting a dose, taking the wrong tablet, or losing track of a complex daily schedule are common, low-visibility errors that nonetheless drive a disproportionate share of avoidable hospital admissions, adverse drug events, and healthcare costs [7, 8]. These errors are not the result of carelessness, but of a structural mismatch: medication regimens have grown more complex precisely as the cognitive and physical capacity to manage them independently tends to decline with age [9]. At the same time, many of the digital tools designed to help, smartphone apps, reminder platforms, complex caregiver dashboards, assume a level of technological fluency that a large fraction of the elderly population simply does not have [10, 11].

Automated medication dispensers have emerged as a promising response to this problem, and the commercial and academic landscape already offers a wide range of devices, from simple rotary carousels to sophisticated multi-canister systems with cloud connectivity. However, as the analysis in Chapter 2 shows, this landscape is fragmented along several axes that matter clinically: whether

a device can actually confirm that a dose was physically collected by the patient, rather than merely dispensed by the machine; whether it takes any physical action when a dose is left uncollected; whether its safety and notification features are bound to a recurring subscription fee; and whether it can keep operating reliably when the home internet connection is unstable, a condition that is disproportionately common in the rural and low-density regions where the need for this kind of support is greatest.

Given this landscape, the present work adopts a deliberately scoped approach. Rather than attempting to simultaneously perfect every layer of the system, including a fully featured interface for the elderly patient, this dissertation prioritises the dimension that the state of the art identifies as the most consistently neglected and clinically consequential: giving the caregiver reliable, granular control over and visibility into the dispensing process itself. The elderly-facing interaction is therefore reduced to its essential, lowest-friction form, physically collecting a dose that the system has already validated as correctly dispensed, while the caregiver is treated as the primary digital user of the system: configuring schedules, managing inventory, and receiving escalated alerts. A dedicated, fully developed elderly-facing application, built on the universal design principles discussed in Chapter 2, is identified as a direction for future work rather than a deliverable of the current prototype.

1.2 Problem Statement and Objectives

This dissertation addresses the following central problem: *how can a cost-effective, home-deployable medication dispenser reliably guarantee that a prescribed dose is not only mechanically released but physically collected by the patient, while giving the caregiver reliable control and visibility over the process and remaining resilient to intermittent connectivity?*

To answer this question, the work pursues the following objectives:

- To design and implement a single-unit dispensing mechanism capable of reliably singulating one dose at a time from a bulk medication store, minimising the risk of jamming or double-dispensing.
- To implement a gravimetric closed-loop validation pipeline, using a load cell, that confirms both the dispense event and the patient's physical collection of the dose, closing a gap identified across existing commercial and academic systems.
- To design a notification architecture that escalates alerts to a caregiver only when adherence is genuinely at risk, avoiding both silent failures and alert fatigue, without relying on a paid subscription service.
- To ensure the system remains functional during connectivity outages, through a local schedule cache and a hardware real-time clock, rather than depending entirely on a remote cloud service.

- To design a caregiver-facing mobile application that provides full configuration, inventory, and monitoring control over the device, establishing the dual-interface model, with a dedicated elderly-facing experience, as the natural next iteration of the system rather than a requirement of the present prototype.

1.3 Alignment with the United Nations Sustainable Development Goals

The present work is also framed within the 2030 Agenda for Sustainable Development, adopted by all United Nations Member States in 2015. Of the seventeen Sustainable Development Goals (SDGs), three are directly and substantively addressed by the MedBox prototype and the broader contribution of this dissertation, rather than tangentially invoked: SDG 3 (Good Health and Well-being), SDG 9 (Industry, Innovation and Infrastructure), and SDG 10 (Reduced Inequalities). Table 1.1 summarises this alignment, identifying the specific target each goal addresses and the concrete element of the present work that contributes to it.

Table 1.1: Alignment between the present work and the United Nations Sustainable Development Goals.

SDG	Goal	Relevant Target	Contribution of the Present Work
3	Good Health and Well-being	3.4: by 2030, reduce premature mortality from non-communicable diseases through prevention and treatment, and promote mental health and well-being.	Directly targets a documented driver of avoidable hospital admissions and adverse drug events among elderly patients (Chapter 2): the automated, physically validated dispensing and collection of medication, with caregiver alerting when a dose is at risk of being missed.
9	Industry, Innovation and Infrastructure	9.5: enhance scientific research and upgrade the technological capabilities of industrial sectors, encouraging innovation, particularly in developing countries.	Adapts an industrial-scale, two-decade-old singulation principle (the rotating-slot mechanism of patent CA2417103C) into a cost-effective, open, FDM-printable implementation built around commodity electronics, in contrast to the proprietary, mechanically complex singulation mechanisms found in current consumer devices (Section 2.3).
10	Reduced Inequalities	10.2: by 2030, empower and promote the social, economic and political inclusion of all, irrespective of age, disability, or other status.	Adopts a subscription-free, open cloud architecture (Section 2.2), removing the recurring financial barrier that ties the safety and notification features of premium commercial dispensers to a paid subscription (Section 2.3), and targets the offline resilience required for deployment in the rural and low-density regions where automated medication management is most needed (Section 2.1.5).

It is worth noting what this alignment does, and does not, claim. The contribution of the present work to SDG 3 is direct and is the central motivation of the dissertation, established in Section 3.1 and evaluated experimentally in the chapters that follow. The contributions to SDG 9 and SDG 10 are architectural and design choices, an open, cost-effective mechanical design and a subscription-free software architecture, rather than claims of broad industrial or economic impact; their relevance is scoped to the proof-of-concept prototype described in this dissertation, and their full realisation would depend on the production-readiness work identified as future work (Section 5.1). Other SDGs with a more indirect or speculative connection to this work, such as those concerning sustainable production of the 3D-printed materials used, are not claimed here, in keeping with the evidence-based approach adopted throughout this dissertation.

1.4 Structure of the Dissertation

The remainder of this dissertation is organised as follows. Chapter 2 presents the state of the art, establishing the clinical and demographic case for automated medication management, reviewing the IoT architectures and communication technologies relevant to home healthcare devices, and conducting a critical, axis-by-axis analysis of existing smart medication dispensers that identifies the specific technological gap this work addresses. Chapter 3 describes the system requirements and overall architecture, and the mechanical, electronic, and software design and implementation of the MedBox prototype. Chapter 4 presents the experimental validation of the dispensing and notification mechanisms, discusses the results obtained, and closes with the limitations of the current prototype and directions for future work. Chapter 5 closes the dissertation with a discussion of the limitations of the current prototype and directions for future work.

Chapter 2

State of the Art

The effective management of medication in elderly populations represents a multifaceted and growing challenge for healthcare systems worldwide. As demographic ageing accelerates and polypharmacy becomes the clinical norm, the gap between what existing solutions offer and what vulnerable patients genuinely need has become both clinically significant and economically costly. This chapter surveys the landscape systematically: beginning with the clinical and demographic context that establishes the urgency of the problem, narrowing to the specific situation in Portugal, examining the technological adoption barriers that constrain any proposed solution, then surveying the IoT architectures and communication technologies available, and concluding with a critical analysis of existing smart medication dispensers and the specific gaps that the present work addresses.

2.1 The Clinical and Demographic Landscape of Elderly Medication Management

This section establishes the clinical and demographic foundations that motivate the present work. It traces the global demographic transition driving population ageing and the consequent epidemiological pivot towards multimorbidity, examines the resulting rise in polypharmacy and the burden of medication self-administration errors among elderly patients, situates this picture within the specific context of Portugal, and discusses the technology adoption barriers that limit the reach of purely digital solutions among older adults. The section closes with a summary that draws these threads together into the clinical and social case for automated medication management.

2.1.1 The Global Demographic Transition and the Emergence of Population Ageing

Contemporary population ageing is an immediate and unprecedented reality that profoundly re-configures social, economic, and healthcare structures on a global scale. Although many European nations have already reached their peak population size and entered a post-transitional phase of

population decline, their societies will continue to age rapidly [12]. This continuous demographic ageing is driven by two primary factors. First, these countries have sustained “ultra-low” fertility rates, often fewer than 1.4 live births per woman for several decades [13]. This long-term scarcity of births shrinks the youth population and significantly reduces the number of women of reproductive age, creating a negative population momentum in which the number of births will continue to decrease even if fertility rates suddenly rebounded [13]. Second, continuous progress in reducing mortality has significantly increased life expectancy, allowing a much larger segment of the population to survive into advanced ages [12, 13]. Consequently, the age structure in these nations increasingly resembles an “inverted pyramid”, causing the share of individuals aged 65 and older in these peaked countries to nearly double from 17% in 2024 to an estimated 33% by 2054, despite the overall shrinking of their total populations [12]. Consequently, the age structure in these nations increasingly resembles an “inverted pyramid”, causing the share of individuals aged 65 and older in these peaked countries to nearly double from 17% in 2024 to an estimated 33% by 2054, despite the overall shrinking of their total populations [12]. This demographic shift is already reflected in measurable strain on healthcare systems: Table 2.1 summarises the change in key hospital utilisation indicators over the past decade and a half, illustrating the scale of the burden that an ageing population places on emergency and inpatient care.

Table 2.1: Evolution of healthcare utilization indicators (U.S.), 2010–2024. [12]

Health Indicator	2010	2024	Percentage Change
Total Hospital Admissions (U.S.)	5.2 million	6.5 million	+25%
Emergency Room Visits	3.8 million	5.3 million	+40%
Hospital Readmission Rate	15.8%	19.0%	+20%
Average Length of Stay	5.4 days	6.2 days	+15%

In contrast to the European trajectory, the United States population is projected to continue growing through mid-century, a trend largely sustained by international immigration [12]. Despite this overall growth, the US is simultaneously experiencing a rapid acceleration in its ageing demographics. For example, between 2010 and 2020, the US centenarian population grew by 50.2%, vastly outpacing the growth rate of the general population [14]. This extreme ageing trend introduces distinct healthcare and logistical challenges; currently, nearly two-thirds of female centenarians in the US live alone or in group quarters such as nursing homes [14, 15]. This high rate of social isolation drastically increases the risk of medication non-adherence and domestic incidents [15], establishing a critical need for automated home healthcare solutions.

On a broader scale, the global population is expected to reach a peak of around 10.3 billion in the mid-2080s [12]. The shifting age structure is a universal phenomenon: by the late 2070s, the global number of individuals aged 65 and older is projected to reach 2.2 billion, officially surpassing the total number of children under 18 worldwide [12]. Furthermore, by the mid-2030s, the highly vulnerable demographic of adults aged 80 or older will reach 265 million globally, outnumbering infants [12]. The continuous increase in life expectancy and this rising proportion of older

adults place a substantial and unprecedented burden on healthcare systems globally [16]. This profound planetary shift dictates that all nations must urgently prepare their healthcare systems and adopt new technologies to support a fundamentally older society [16]. Consequently, there is an established critical need for Ambient Assisted Living (AAL) and smart home technologies to maintain safety, autonomy, and continuous monitoring in domestic environments [17].

2.1.2 The Epidemiological Pivot: Multimorbidity and the Healthcare Utilisation Crisis

Extended longevity has brought with it an epidemiological transition, where infectious diseases have been largely replaced by non-communicable diseases (NCDs), frequently occurring simultaneously in the same individual, a phenomenon known as multimorbidity [sastri_2025][18]. The presence of two or more chronic conditions affects about 40% of individuals aged 65 and older, and this prevalence increases linearly with advanced age [18]. This complex clinical scenario requires intensive therapeutic regimens, resulting in a drastic increase in the demand for healthcare services and specialised care [19].

Mortality among the elderly population is increasingly linked to pathologies such as Alzheimer, dementias, cardiovascular diseases, and cancer, in addition to systemic conditions like frailty [19]. Longitudinal data between 2010 and 2024 show alarming trends in the increase of age-adjusted mortality rates and the volume of hospital care utilisation [19]. Additionally, the 40% increase in mortality from Alzheimer's and dementia, combined with the 45% growth in deaths related to frailty and multi-morbidity, underlines the inability of traditional care models to manage patients experiencing simultaneous cognitive and physical declines [19]. Frailty, in particular, increases vulnerability to adverse events, making simple acts such as the self-administration of drugs without awareness of adverse reactions high-risk tasks [19].

2.1.3 Polypharmacy: Prevalence, Determinants, and Clinical Projections

Polypharmacy, typically defined as the concurrent use of five or more medications, represents a highly prevalent clinical reality among older adults, carrying substantial implications for therapeutic safety and efficacy. According to research on polypharmacy management among community-dwelling individuals, 37.1% of men and 36% of women between the ages of 75 and 85 take five or more prescription medications [20]. Furthermore, the rate of hyper-polypharmacy (defined as taking ten or more medications) has seen a dramatic escalation. Population database analyses in Scotland showed that its prevalence among older adults more than tripled over a 15-year period, rising from 4.9% in 1995 to 17.2% in 2010 [21].

Global, European, and US Prevalence Trends. Population-level data from the National Health and Nutrition Examination Survey reveals a dramatic upward trajectory in the United States. The prevalence of polypharmacy among US adults aged 65 and older significantly increased from 24% to 39% between 1999 and 2012 [21]. The systemic implications are stark, with

the average number of prescribed medications for Medicare enrollees rising alongside a 192% increase in annual prescription costs per beneficiary over a single decade [20]. The most recent data from the Survey of Health, Ageing, and Retirement in Europe (SHARE Wave 9), which evaluated community-dwelling adults aged 65 and older, highlights a severe European landscape, showing an overall polypharmacy prevalence of 36.2% [18]. In parts of Asia, the numbers are even higher: the prevalence of polypharmacy in adults aged 65 and older has been reported at 86.4% in South Korea, while in Taiwan, it reached 84% specifically among frail older adults requiring long-term care [21].

Determinants of Polypharmacy. While multimorbidity is the primary driver, several independent risk factors exacerbate the problem [22]. Older age and poor self-perceived health strongly predict multiple medication use [18][22]. Furthermore, segmented healthcare — where patients consult multiple specialists without a unified medical record — often results in duplicative prescriptions and therapeutic conflicts [20]. This often leads to a “prescribing cascade”, where additional medications are prescribed to treat or control the adverse effects caused by initial drugs, independently driving up the daily pill burden [20][22]. When this uncoordinated accumulation of prescriptions meets the physiological vulnerabilities of ageing, it inevitably triggers a cascade of negative clinical outcomes.

Clinical Projections and Adverse Effects. The intersection of polypharmacy and age-related physiological changes (such as declining renal and hepatic clearance) exponentially increases the risk of systemic harm [20]. Patients subjected to complex polypharmacy regimens present significantly higher chances of suffering adverse events, an effect that is drastically compounded by the presence of cognitive impairments like dementia [22][23]. Polypharmacy is strongly associated with falls, hospitalisation, and mortality in geriatric populations, as summarized in Table 2.2. Seniors with polypharmacy face nearly three times higher odds of an adverse drug event, an emergency department visit, or a hospital admission than their non-polypharmacy peers [23], with the presence of dementia compounding hospitalization risk to nearly nine-fold [23]. This elevated risk extends to long-term outcomes: patients on ten or more medications face a 50% higher rate of falls over a five-year period [20], and one-year all-cause mortality risk more than doubles under polypharmacy [24], an effect that persists and intensifies under continuous or hyper-polypharmacy regimens [25]. (Table 2.2)

Table 2.2: Impact of polypharmacy and concurrent cognitive decline on clinical outcomes.

Clinical Outcome	Baseline Risk Metric (Polypharmacy)	Impact of Concurrent Cognitive Decline (e.g., Dementia)
Adverse Drug Events (ADEs)	OR 2.9 [23]	Risk rises to OR 3.7 [23]
Emergency Department (ED) Visits	OR 2.5 [23]	Risk rises to OR 5.4 [23]
Avoidable Hospitalization	OR 3.0 [23]	Risk drastically rises to OR 8.5 [23]
Falls (rate ratio, ≥ 10 drugs, 5-year risk)	IRR 1.50 [20]	–
All-Cause Mortality (1-year risk)	HR 2.37 [24]	Continuous/hyper-polypharmacy elevates risk to OR 2.57 [25]

Note: OR = Odds Ratio; HR = Hazard Ratio; IRR = Incidence Rate Ratio. While not directly interchangeable, all three express the relative increase in risk compared to a non-polypharmacy baseline.

2.1.4 Medication Self-Administration Errors: A Hidden Clinical and Economic Burden

Prevalence and Phenomenology. As the global population ages and polypharmacy becomes standard practice, the domestic setting has emerged as a high-risk environment for Medication Self-Administration Errors (MSEs). A cross-sectional study conducted among people aged 60 and above indicates that up to 69.2% of illiterate or low-literate community-dwelling older adults commit at least one self-administration error over a six-month period [26]. The most frequently reported errors are simple forgetfulness or dose omission (up to 37.3% of domestic errors), incorrect dosing (30.4%), and improper timing (13.87%) [26]. Furthermore, older adults frequently engage in unauthorised medication modifications, such as the inappropriate splitting of tablets, inadvertently destroying active ingredients and risking severe adverse effects [9][7].

Transitions of Care. The vulnerability to MSEs is acutely exacerbated during care transitions, particularly when patients are discharged from hospital to home. Systematic reviews demonstrate that a median of 53% of adult and elderly patients experience at least one medication error post-discharge, and nearly 20% suffer an adverse drug event [27]. Consequently, poor home medication management directly drives hospital returns: 16% of all unplanned 30-day hospital readmissions are medication-related, with 40% deemed potentially preventable [8]. Within these preventable readmissions, transition errors account for 30% of cases, while direct patient non-adherence and self-administration errors account for 35% [8].

Economic Burden. This high prevalence of domestic errors translates into an astronomical clinical and economic burden. Globally, the World Health Organization estimates that medication

errors consume approximately 1% of total health expenditure, equating to \$42 billion annually [7]. In England, avoidable adverse drug events cost the National Health Service up to £1.6 billion per year, consuming 3.8 million hospital bed-days and contributing to over 22,000 deaths [28][7]. Major economic models frequently exclude administration errors committed by patients and care-givers at home [7], meaning the true indirect burden remains heavily underestimated. Roughly 18% of older adults who commit a medication error at home experience subsequent adverse clinical events, creating a silent pipeline of morbidity that feeds back into costly acute care systems [26][29].

The Impact of Low Health Literacy. A critical driver of this hidden epidemic is low health literacy, which profoundly impairs an individual's ability to access, comprehend, and apply health-related information [9]. It is estimated that only 3% of adults aged 65 and older possess proficient health literacy skills [9]. Being completely illiterate significantly predicts higher MSE rates, increasing the likelihood of errors by 1.62 times compared to those with basic literacy [26]. Unable to read complex prescription labels or warning brochures, low-literate older adults are forced to rely entirely on the physical appearance of their pills [9], drastically increasing the risk of medication confusion, duplicate dosing, and accidental poisoning.

2.1.5 The Portuguese Context

Portugal occupies a particularly acute position within the global ageing landscape. According to data from Statistics Portugal (INE), the proportion of the population aged 65 and over reached 24.3% in 2024, making Portugal the second most aged country in the European Union, surpassed only by Italy (24.7%) and well above the EU27 average of 21.6% [30]. The national ageing index — which compares the population aged 65 and over with the population aged 0 to 14 — reached 192.4 elderly per 100 young people in 2024, up from 147.6 in 2015 [30]. The median age of the resident population rose to 47.3 years in 2024, one of the highest in Europe. These trends are projected to intensify: current projections indicate that by 2050, more than one third of the Portuguese population will be aged 65 or older.

This demographic context is directly reflected in the clinical burden. Portugal is among the most severely affected nations in Europe with respect to polypharmacy, with a prevalence of 51.3% among community-dwelling adults aged 65 and older, according to SHARE Wave 9 data [18] — significantly above the European average of 36.2%. A nationwide cross-sectional study conducted across all five Portuguese healthcare administrative regions confirmed that polypharmacy is pervasive in primary care, with a substantial fraction of older patients simultaneously prescribed drugs with a high potential for interactions and adverse events [31]. Among institutionalised older adults in Portugal, the figures are even more alarming: a 2022 study found that major polypharmacy (7 or more drugs per day) affected 80.8% of the studied population, with an average of 7.6 medications taken daily, and that 79.3% were simultaneously taking at least one potentially inappropriate medication according to the Beers Criteria [32].

Adherence among polymedicated elderly patients in Portugal is equally concerning. A cross-sectional study conducted in primary care centres in the central region of Portugal, involving 1089

participants aged 65 and over, found that 47.7% were considered non-adherent [33]. The most commonly cited reasons for non-adherence were forgetfulness (38.8%), difficulties in managing medication (14.3%), concerns about side effects (10.7%), and the price of medication (9.2%) [33]. Difficulties in managing medication were independently associated with non-adherence in regression models, pointing to the need for solutions that actively reduce the cognitive and physical burden of the daily medication routine.

Health literacy compounds this picture. A study focused on low-income older adults in the Beira interior region of Portugal found that approximately 61% of the Portuguese population has problematic or inadequate health literacy [34], consistent with European averages [35] but particularly impactful in the context of a highly aged and rural population. Portugal also has significant rural and low-density regions, such as the interior, Alentejo, and Trás-os-Montes, where the concentration of elderly individuals living alone [36], combined with reduced access to pharmacy services, makes automated home medication management both more necessary and more challenging to implement [37].

In 2023, Order No. 263/2023 authorised community pharmacists in Portugal to renew prescriptions for chronic conditions [38], a measure that partially acknowledges the pressure that polypharmacy places on clinical infrastructure [21]. While this represents a step forward, it does not address the core problem of reliable daily self-administration at home, particularly for cognitively impaired or functionally limited patients [7].

2.1.6 Technology Adoption Barriers in the Elderly: The Digital Divide

Assessment Methods and the Spectrum of the Digital Divide. To accurately evaluate the technological capabilities of older adults, the scientific literature proposes multifaceted assessment methods. Validated instruments such as the eHealth Literacy Scale (eHEALS), the Mobile Device Proficiency Questionnaire (MDPQ), and the Computer Attitude Scale (CAS) are frequently employed to measure digital aptitude. Theoretical models, including the European Commission's Digital Competence (DigComp) Framework [39], evaluate users across multiple dimensions: information and data literacy, communication, digital content creation, safety, and problem-solving. This comprehensive analysis reflects the three layers of digital exclusion: the Access Divide (lack of physical access to devices or networks), the Use Divide (lack of digital skills and functional use), and the Knowledge Divide (inability to extract tangible benefits from digital usage) [10].

Statistical analysis of these divides reveals an alarming reality. In the United States, approximately 27% of adults aged 65 and older do not use the internet at all, and more than 35% lack a home broadband connection [11]. In Europe, 2024 data highlights that while 92.83% of the general population actively uses the internet, this rate drops to 76.27% for the 65+ demographic [10]. The abyss deepens regarding mobile internet, utilised by 86.31% of the general European public but only 60.5% of the elderly. Although the European Union established a target of ensuring that 80% of adults possess basic digital skills by 2030, current figures stagnate at 55.6%, heavily dragged down by acute exclusion among older age groups with lower educational attainment [10].

The Psychological Barrier: Technophobia and the Cycle of Exclusion. Contrary to the popular belief that economic cost is the primary barrier to adoption (cited by only 23% to 25% of European non-users), the lack of digital skills (45%) and lack of interest (45%) are the most significant obstacles [40]. At the root of this reluctance is often profound technology anxiety. Many older adults harbour a deep-seated fear of making catastrophic operational errors that could lead to equipment damage, personal data leaks, or financial fraud. Overcoming this barrier requires a departure from standard design: solutions must incorporate multimodal feedback (visual, auditory, and tactile) and foster intergenerational digital learning.

Table 2.3: Barrier categories, underlying physiological/cognitive decline, and resulting operational friction in technology use.

Barrier Category	Specific Physiological/Cognitive Decline	Resulting Operational Friction (Impact on Tech Use)
Motor & Physical	Decreased fine motor dexterity, loss of finger joint flexibility, and hand tremors [41]	High rate of accidental touches; inability to operate small, densely arranged touch targets on flat screens lacking tactile feedback [41]
Sensory	Visual impairment (reduced acuity) and hearing loss [41]	Difficulty recognizing small-font text and low-contrast interface elements; missing critical audio prompts or feedback [41]
Cognitive	Decline in fluid intelligence, limited working memory capacity, and naturally increased reaction times [42]	Cognitive overload from complex, multi-level menus; inability to memorize or recall intricate passwords; repeated failures in timed biometric authentications [42]
Psychological	“Technology anxiety” and fear of catastrophic errors (e.g., equipment damage, financial loss, or data leaks) [43]	Active digital exclusion and reluctance to complete necessary setups (such as binding accounts or entering verification codes) [43]

Operational Friction and Biological Decline. Digital exclusion is severely aggravated by the physiological and biological barriers inherent to ageing. The natural decline of fine motor dexterity, loss of finger joint flexibility, and the onset of hand tremors result in high rates of accidental touches on modern touchscreens, which inherently lack mechanical, tactile feedback. Simultaneously, older adults experience a decline in fluid intelligence (the capacity to process new information quickly) and working memory, which clashes directly with the constant updates and complex functional hierarchies of modern applications [44]. This neurobiological context naturally increases reaction times and error rates, turning the navigation of complex app menus into deeply frustrating and anxiety-inducing experiences.

Geographical Inequalities. Physical and psychological barriers to technology adoption are drastically amplified by geographical context. Regular internet access rates sit at merely 54% in rural populations, compared to 66% among urban residents [10]. Elderly individuals living in rural areas face particular challenges due to underdeveloped broadband infrastructure and the progressive closure of local analogue services. Furthermore, the migration of younger demographics to cities dissolves traditional intergenerational support networks, leaving older rural adults with limited direct assistance. Empirical models confirm that the digital divide affecting older adults is significantly more pronounced in rural communities [45]. This phenomenon is highly relevant in Portugal, where rural and interior regions present the highest concentrations of elderly populations in the country [37].

The Mandate for Inclusion: Universal Design. Expecting older adults to overcome age related physical and cognitive constraints to adapt to complex digital interfaces is a flawed approach. True digital equity requires accessible solutions that respect the diverse capabilities of this population. To achieve this, healthcare technology must adopt the principles of Universal Design [46], summarized in Table 2.4, ensuring that products are intuitively usable, require low physical effort, offer high tolerance for error, and avoid age related stigmatisation. Interestingly, when technologies are correctly designed according to these principles, they can act as a cognitive support framework. Recent meta-analyses indicate that regular and supported technology use is associated with a significantly reduced risk of cognitive decline, potentially building a form of technological cognitive reserve [47]. Given the severe infrastructural deficits in rural areas and the prevalent technology anxiety among older adults, it is critical that essential health systems, such as automated medication dispensers, are capable of functioning seamlessly in both online and offline environments.

Table 2.4: Universal Design Principles, description and example in practice [46].

Universal Design Principle	Description	Example in Practice
1. Equitable Use	Useful and marketable to people with diverse abilities; avoids stigmatizing any user group.	Doors that automatically open for everyone.
2. Flexibility of Use	Accommodates a wide range of individual preferences and abilities (e.g., left/right-handed).	ATMs with buttons far enough apart to be pressed accurately.
3. Simple and Intuitive Use	Easy to understand, regardless of user's experience, knowledge, language skills, or cognitive level.	Providing instructions in a series of clear illustrations instead of complex text.
4. Perceptible Information	Communicates necessary information effectively, regardless of ambient conditions or the user's sensory abilities.	Software that relays information via multiple modes (visually through text/pictures, and audibly).
5. Tolerance for Error	Minimizes hazards and the adverse consequences of accidental or unintended actions.	Hallways that return to common areas rather than stop in dead ends.
6. Low Physical Effort	Can be used efficiently and comfortably with a minimum of fatigue.	Bottle caps that are easy to grip and require only a small range of motion to open.
7. Size/Space for Approach/Use	Appropriate size and space provided for approach, reach, manipulation, and use regardless of body size, posture, or mobility.	Wall-mounted components that are visible, easy to reach, and easy for all hand sizes to use.

Although originally formulated in 1997, these seven principles remain the canonical reference framework in contemporary accessibility research precisely because they are technology-agnostic: they specify properties a design must exhibit (equitable, intuitive, error-tolerant, low-effort) rather than prescribing specific interfaces, allowing them to be applied unchanged to touchscreens, voice assistants, or embedded medical devices that did not exist at the time of their publication. Their continued citation in recent literature on age-friendly digital interfaces [41] confirms their ongoing relevance.

2.1.7 Summary: The Clinical and Social Case for Automated Medication Management

The preceding sections establish a convergent and compelling need statement. A population that is ageing faster than any previous generation, concentrated in geographically isolated settings, experiencing polypharmacy at rates above 50% in countries like Portugal [31], and already exhibiting nonadherence rates approaching 50% [33], cannot be adequately served by passive weekly pill-boxes or by digital solutions that require a level of technological literacy that the majority of its members do not possess. Any effective intervention must guarantee reliable daily medication delivery without requiring the patient to read small print, open rigid packaging, or interact with a complex digital interface, and it must do so even when the internet connection is intermittent or unavailable.

2.2 IoT Architectures for Home Healthcare Applications

Internet of Things (IoT) technology has emerged as a foundational enabler for home healthcare systems, providing the sensing, actuation, communication, and cloud integration capabilities required to bridge the gap between the patient's home and the healthcare network. This section examines the key architectural components relevant to an automated home medication dispenser: the wireless communication technologies available, the balance between local (edge) and remote (cloud) processing, and the cloud platforms used for real-time data management and notification delivery.

2.2.1 Communication Technologies: WiFi, Bluetooth, and Cellular

The selection of a communication technology for an IoT healthcare device is a critical architectural decision involving tradeoffs between data rate, range, power consumption, infrastructure dependency, and cost [48, 49]. Three technologies dominate the short to medium range segment of home healthcare IoT: WiFi, Bluetooth, and cellular networks.

WiFi (IEEE 802.11 b/g/n) offers high data throughput (up to 600 Mbps in the 802.11n standard), moderate range (typically 30 to 50 m indoors), and seamless integration with the residential broadband infrastructure that is already present in most homes [50]. Its primary limitations in IoT contexts are relatively high power consumption compared to low power alternatives, and a dependence on a functioning residential router and internet service provider connection. For a home medication dispenser that is permanently powered and requires bidirectional synchronisation with a cloud database, WiFi is the natural choice, since it eliminates the need for any additional communication hardware or SIM subscription.

Bluetooth (IEEE 802.15.1) and Bluetooth Low Energy (BLE) operate at shorter ranges (typically under 10 m) with significantly reduced power consumption, making them appropriate

for body worn sensors, smart scales, and peripheral devices that communicate with a nearby smart-phone gateway rather than directly with the internet [49]. In a multi device medication management ecosystem, Bluetooth could be used to link wearable vital sign sensors to a central dispenser unit, offloading internet connectivity to the phone.

Cellular (GSM/LTE) provides the widest coverage area and does not depend on residential WiFi infrastructure, making it the technology of choice for devices deployed in locations without broadband access [50]. Its disadvantages are higher power consumption and the recurring cost of a SIM data plan, which is particularly relevant in the context of fixed income elderly patients. Table 2.5 summarises the key characteristics of these three technologies in the context of home healthcare IoT.

Table 2.5: Comparison of wireless communication technologies for home healthcare IoT applications [48].

Parameter	WiFi (802.11n)	Bluetooth / BLE	GSM / LTE	ZigBee
Data rate	Up to 600 Mbps	Up to 3 Mbps / 1 Mbps	0.1 Mbps to 1 Gbps	250 kbps
Range (indoor)	30 to 50 m	1 to 10 m	Nationwide	10 to 100 m
Power consumption	Medium	Very low	High	Very low
Infrastructure	Router + ISP	Smartphone gateway	SIM / operator	Coordinator node
Recurring cost	None (beyond ISP)	None	SIM plan	None
Best use case	Mains powered home device	Wearable sensors	Remote / rural areas	Smart building sensors

2.2.2 Edge Computing and Offline Resilience

A critical architectural consideration for home healthcare IoT is the degree to which the device can continue to operate in the absence of an Internet connection. Cloud-dependent architectures, in which all logic resides on a remote server, are vulnerable to network outages, router failures, and ISP disruptions, all of which are disproportionately common in the rural and low-density areas where the need for automated medication management is most acute.

Edge computing addresses this vulnerability by performing processing and decision-making locally, at the “edge” of the network, i.e., on the device itself or on a local gateway, rather than on a remote cloud server [51, 52]. In the context of a medication dispenser, edge-first design means maintaining a local copy of the daily schedule, using a hardware real-time clock (RTC) as a fallback time source, and executing the dispensing logic on the microcontroller even when the RTDB is unreachable. State changes that occur during offline operation are buffered locally and synchronised to the cloud once connectivity is restored.

Fog computing extends this concept by introducing an intermediate processing layer between the edge device and the cloud, typically implemented on a local gateway (such as a residential Wi-Fi router or a smartphone) that aggregates data from multiple IoT devices and performs pre-processing before forwarding to the cloud [51]. While fog computing adds architectural complexity, it reduces cloud bandwidth and latency, and can be particularly valuable in multi-device setups, for example, a future MedBox with multiple compartments, each equipped with its own sensor suite, where a local gateway aggregates readings before cloud upload.

For the single-compartment MedBox prototype, a pure edge-first approach, in which the ESP32-C6 itself handles all local logic and the cloud is used for persistence and notification only, provides the necessary offline resilience without the additional complexity of a fog layer.

2.2.3 Cloud Platforms and Real-Time Data Management

The cloud layer of a home medication management system must fulfil three principal functions: persistent storage of the medication schedule and historical data; real-time synchronisation of state changes between the device and the caregiver’s mobile application; and delivery of push notifications to the caregiver’s smartphone when anomalous conditions occur.

Firebase, developed by Google, has emerged as a dominant platform for IoT and mobile health applications requiring these capabilities [53, 54]. Its Realtime Database (RTDB) uses a WebSocket-based streaming protocol that pushes state changes to all connected clients within milliseconds, without requiring any polling. This reactive model is particularly well-suited to medication management, where latency requirements are tight (the caregiver should be aware of a missed dose within seconds, not minutes) and where the number of simultaneous clients is small (typically one dispenser and one or a few mobile clients). Firebase Cloud Messaging (FCM) provides a free, low-latency push notification delivery service that integrates natively with Android and iOS, allowing critical alerts to reach the caregiver’s device even when the mobile application is closed, a requirement that client-side notification architectures cannot reliably satisfy due to Android’s Doze Mode battery management policies [55].

The choice of Firebase for home medication management prototypes is therefore grounded in its combination of native real-time streaming, integrated push notification delivery, and a free tier that accommodates the data volumes of a single-patient home device without subscription cost.

2.3 Smart Medication Dispensers: An Analytical Survey

This section surveys the current landscape of in-home smart medication dispensers, spanning consumer-commercial products, academic prototypes, and connected (app-enabled) devices. Rather than providing a simple product catalogue, this analysis interrogates existing solutions along fixed technical and clinical axes to isolate a defensible technological gap and position the contribution of the present work. The authoritative quantitative baseline for this population is established by Faisal, Ivo, and Patel (2021), who screened 51 smart medication adherence products (38 commercial, 13 academic prototypes). Their systematic taxonomy provides the ground-truth parameters

for current device features: 74.5% feature alarms, 88.2% notify the patient, 47.1% notify a care-giver, 43.1% offer a locking mechanism, 62.7% operate under a multidose framework, and 23.5% utilise a unit-dose regime [56].

2.3.1 Device Taxonomy: Storage Regimes and Dispensing Mechanisms

Before analysing device behaviours, the physical organisation of medication must be classified, as it constrains storage capacity, refill burden, mechanical complexity, and the feasibility of automated single-dose control. Following the classifications of Faisal et al. (2021), the market splits into two core regimes:

- **Unit-Dose / Pre-Sorted Regime:** Each compartment holds the complete combination of pills for a single scheduled intake. Rotary carousels (e.g., MedReady [1], LiveFine 28 [2]), the MedMinder [3] cup keyboard, and the Evondos [4] medicine dispenser belong to this category. The advantage is mechanical simplicity and low singulation risk, as the device merely exposes or ejects a pre-filled unit. The cost is rigid capacity and high manual labour; a standard 28-slot carousel supporting four daily intakes yields only seven days of autonomy before requiring a complete human refill.
- **Bulk / Singulation Regime:** Each container stores multiple units of a single medication type, and the device mechanically separates, counts, and dispenses the required units at runtime. Devices like Hero [5] (up to ten canisters) and MedaCube [6] (up to 16 bins) exemplify this paradigm. While offering flexible scheduling and long autonomy, this regime introduces high mechanical and sensing complexity, as reliable singulation must tolerate vast variations in pill geometries and surfaces.

Representative examples of these dispensing architectures are shown in Figures 2.1- 2.6.

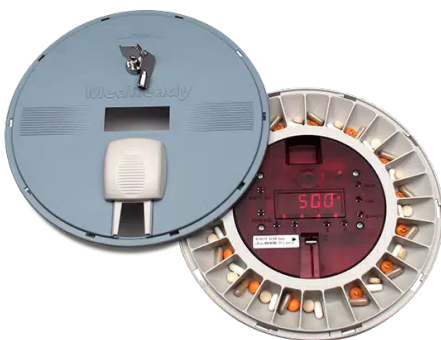


Figure 2.1: MedReady rotary carousel dispenser [1].



Figure 2.2: LiveFine 28 automated pill dispenser [2].



Figure 2.3: MedMinder connected pill dispenser [3].



Figure 2.4: Evondos pouch-roll medicine dispenser [4].



Figure 2.5: Hero bulk-canister dispenser [5].



Figure 2.6: MedaCube multi-bin automatic dispenser [6].

2.3.2 Dose Validation Mechanisms: The Capability Ladder

Validation mechanics dictate how or whether a device confirms that the therapeutic compound physically left the machine and reached the patient. This axis exposes a clear capability ladder across existing architectures:

- **Open-Loop Indexing:** Carousel devices (MedReady [1], LiveFine [2]) register only that the internal tray advanced. They cannot verify if the pills were extracted or left in the slot, creating a structural risk for the dangerous accumulation of consecutive missed doses.

- **Binary Open-Sensing:** Drawer or lid switches (e.g., Wong et al., 2019) register a positive adherence event whenever the cabinet is opened. This constitutes a structural source of false positives, failing to prove that the pill was actually removed.
- **Optoelectronic Counting:** Premium systems like Hero count individual pills as they fall down an internal chute. While confirming mechanical release from the bulk canister, they do not confirm patient collection, allowing the pills to sit indefinitely exposed in an open delivery cup.
- **Single-Point Gravimetric Sensing:** Some academic prototypes attach a load cell directly to the pillbox compartment or bottle to track the remaining medication mass over time [57, 58]. This approach can flag that mass changed, but it typically validates a single dispensing or removal event in isolation, rather than closing the loop between the dispense action and the collection action for each individual dose.
- **Gravimetric Closed-Loop:** A load cell on the collection platform confirms both that mass increased on dispense and that it decreased on collection. This is the only mechanism that closes the loop on the patient side rather than the machine side. This dual-event verification approach remains largely absent from current commercial and academic systems [56], representing a critical gap in physical adherence verification.

As observed by [59], internet connectivity in recent adherence devices is heavily oriented toward remote monitoring rather than enabling automated, device-side interventions, a limitation that directly cripples the validation layer of most modern devices.

2.3.3 Alert Management and Caregiver Notification Architectures

Telemetrical routing to the caregiver spans cellular, Wi-Fi, Bluetooth, NFC, and RFID links, with 47.1% of products offering remote notifications [56]. However, analysing the connected segment reveals a significant accessibility barrier regarding how these notifications are managed.

Currently, advanced notification management is largely restricted to premium proprietary ecosystems. High-end devices (Hero [5], MedMinder [3], MedaCube [6]) route telemetry through closed vendor back-ends, yielding polished caregiver portals and sophisticated alert escalation. By notifying remote caregivers only when doses are missed or hardware fails, these systems successfully combat *alert fatigue*, the cognitive desensitisation that causes caregivers to ignore critical warnings due to notification overload [60, 61].

While highly effective, this model fundamentally ties crucial systemic oversight to a recurring subscription fee [56], imposing a financial barrier that excludes a significant portion of the target population. An optimal architecture must democratise these refined notification models, adopting the accessibility of an open, subscription-free cloud paradigm [59] while implementing the smart notification behaviour of premium devices. By utilising asymmetric, cascading escalation, where routine adherences are silently logged and active push notifications are strictly reserved for critical

events, a system can preserve caregiver attention [60] and ensure patient safety without imposing a financial paywall.

2.3.4 Active Fail-Safes and Physical Responses to Non-Adherence

Analysing the existing literature and commercial landscape reveals three behavioural tiers regarding physical fail-safes after a dose is dispensed but not collected:

- **Passive (No Action):** The majority of systems, including rotary carousels, recent academic prototypes, and premium devices like Hero [5], take no physical action at the hardware level [56]. The uncollected dose remains accessible in an open cup or slot. For cognitively impaired patients, this creates a severe iatrogenic risk of accidental overdose or late ingestion outside the therapeutic window [7, 26].
- **Locked / Human-Gated:** A minority of devices physically restrict the missed dose but remain entirely dependent on human intervention. Systems such as Evondos [4] lock missed-dose compartments, with only authorised care staff able to access the interior to remove uncollected doses [56]. Although safer than passive accumulation, these systems are not autonomous.
- **Autonomous Removal (The Structural Gap):** Currently, no surveyed device, neither commercial nor academic, autonomously retracts or sequesters an uncollected dose to render it inaccessible without human action [56, 59]. This represents the most critical unaddressed gap in the physical safety layer of existing systems.

2.3.5 Usability and User Acceptance Among Elderly Patients

The usability of electronic medication adherence products is an under-investigated dimension of the design space. A prospective mixed-methods study by Patel et al. (2020), published in the *Journal of Medical Internet Research*, evaluated 21 electronic medication adherence products using the System Usability Scale (SUS) and the NASA Task Load Index (NASA-TLX) with a sample of older adults, caregivers, and clinicians [62]. The results revealed substantial variability: mean SUS scores ranged from 28.63 (SD 21.24) for the MedReady 1700 automated dispenser to 78.67 (SD 15.57) for the TimerCap, on a scale where scores above 70 are considered acceptable and above 85 excellent. Critically, participants were able to complete all required interaction steps without assistance only 55.3% of the time, meaning that nearly half of interactions with these devices required external help, in a population for whom independent use is the entire value proposition. These findings align with the theoretical prediction from the Technology Acceptance Model (TAM) that perceived ease of use is a primary determinant of technology adoption among elderly users [63]. For medication management devices specifically, the consequence of poor usability is not merely user frustration but clinical non-adoption: a device that the patient cannot reliably operate independently provides no meaningful improvement over no device at all. A separate proof-of-concept study by Zijp, Touw and van Boven (2020) evaluated a novel smart pill bottle prototype using the

SUS and found a mean usability score of 79.3 (range: 57.5–97.5) and an overall dispense adherence rate of 88% [64]. The study concluded that usability and reliability must be established as first-class design constraints, not afterthoughts, for medication adherence devices, particularly for the elderly population, reinforcing usability as a central evaluation axis for the present analysis.

2.3.6 The Portuguese Precedent: The Balvia / SelfMed Ecosystem

Given that Section 2.1.5 established Portugal as one of the most acutely affected countries in this domain, it is important to situate the present work relative to comparable national initiatives rather than international products alone. The most directly comparable Portuguese effort is **Balvia**, a commercial medication-management ecosystem that originated from the academic project *Self-Med* and is incubated at the Instituto Politécnico de Bragança (IPB) through its Research Centre in Digitalisation and Intelligent Robotics (CeDRI) and its M2PI entrepreneurship programme, with funding support from the “la Caixa” Foundation and BPI [65].

The Balvia ecosystem is architected as an event-driven, distributed system in which the physical pill dispenser, a Flutter-based mobile application, an optional smartwatch companion application, and a Spring Boot back-end hosted on AWS communicate through defined message events rather than continuous polling [66, 67]. The dispenser itself relies on motorised release across multiple storage sections, with onboard sensors confirming that a dose has been mechanically released, and the system withholds dispensing until the correct user has been identified [68]. Companion-device communication uses Bluetooth Low Energy with a custom Version-Command-Length-Parameters (V-C-L-P) message protocol, backed by a fail-safe RESTful API layer that keeps data flowing if the Bluetooth link is briefly interrupted [68]. Medication schedules are cached locally (via an on-device SQLite store on the companion applications) and re-synchronised with the cloud at fixed intervals, giving the ecosystem a degree of offline tolerance broadly consistent with the edge-first philosophy discussed in Section 2.2 [68, 69].

Measured against the analytical axes developed throughout this section, Balvia represents a meaningful advance over the smaller academic prototypes surveyed above, most notably in engineering maturity: commercial-grade cloud infrastructure, an existing companion-device ecosystem, and active national funding and incubation support. However, based on the publicly available documentation, the same two structural gaps identified elsewhere in this chapter appear to persist. First, dose validation is confirmed at the point of mechanical release and by the patient’s own acknowledgement on the companion device, rather than by an independent physical measurement of collection, such as gravimetric confirmation on a dedicated collection platform [68]; the associated thesis explicitly lists sensor-based collection confirmation as a feature for future iterations rather than one already validated end-to-end. It is worth noting that the same source elsewhere states, in less unambiguous terms, that the physical dispenser already incorporates a scale-based check on whether a dispensed dose has been consumed; the public record does not clarify whether this refers to an implemented and validated mechanism or a planned capability, and this assessment is offered as provisional pending clarification from Balvia’s own technical documentation rather than as a settled claim. Second, no autonomous physical fail-safe response to an uncollected dose

is documented in the available sources, which would place Balvia within the *Passive* tier of the active fail-safe classification introduced previously. Balvia therefore constitutes the closest national precedent to the present work, offering superior commercial polish and cloud openness, but it does not, on the basis of the available literature, unambiguously close the technology-clinical gap that this dissertation addresses.

2.3.7 Synthesis: The Technology-Clinical Gap

A vertical cross-reading of the multi-axis market landscape (summarised in Table 2.6) exposes the precise technology-clinical gap that leaves cognitively impaired patients under-protected.

Table 2.6: Visual syndthesis of the technology-clinical gap across representative smart adherence architectures. (✓ Fully Addressed | ~ Partially Addressed | ✗ Failed / Critical Risk)

Architecture	Accessibility & Cost	Storage Regime	True Validation	Active Fail-Safe	Offline Resilience
MedReady / LiveFine	~ Medium	f ~ Low (pre-sorted)	✗ Open-loop	✗ Passive	✓ Yes
Hero Health	✗ Subscription wall	✓ High (bulk)	~ Partial (internal count)	✗ Passive (exposed cup)	✗ Cloud-dependent
MedaCube	✗ High upfront cost	✓ High (bulk bins)	~ Partial	~ Human-gated	✗ Cloud-dependent
Evondos	✗ Institutional pricing	~ Pharmacy-reliant	~ Partial	~ Human-gated	✗ Cellular-reliant
Wong et al.	✓ Open/cost-effective	~ Pre-sorted	✗ Binary switch	✗ Passive	✗ Wi-Fi only
Nurhakim et al.	✓ Open/cost-effective	~ Pre-sorted	✗ Binary switch	✗ Passive	✗ Firebase-reliant
Balvia / SelfMed (PT)	~ Emerging startup pricing	~ Multi-compartment (motorised)	~ Partial (release sensor + user ack)*	✗ Passive	~ Local cache / BLE-API fail-safe
Target Architecture	✓ Open-source	✓ Bulk w/ singulation	✓ Gravimetric closed-loop	✓ Autonomous retraction	✓ Edge-first / RTC

Premium commercial systems possess advanced mechanics capable of bulk storage, but they lock safety features behind prohibitive paywalls and fail to autonomously retract exposed medication. Academic prototypes solve the financial barrier but retreat to manual, pre-sorted storage regimens while suffering from extreme cloud dependency, naive binary sensing, and completely passive responses to missed doses. A true closed-loop cyber-physical system is entirely absent from both sectors. Notably, this conclusion holds even when the closest national precedent is included in the comparison: Balvia/SelfMed (Section 2.3.6), the most mature Portuguese initiative in this space, improves on engineering polish and cloud infrastructure but, based on the available

literature, still confirms dose validation only at the point of mechanical release and via patient self-report, and does not implement an autonomous fail-safe response to an uncollected dose [68].

2.4 Summary and Positioning of the Present Work

The survey conducted in this chapter converges on a clear and specific architectural blueprint for an effective home medication management system. The required solution must guarantee: (1) **definitive dose validation** through gravimetric closed-loop sensing that confirms both mechanical dispense and patient collection; (2) **autonomous fail-safe retraction** of uncollected doses once the therapeutic window expires; (3) **asymmetric alert management** using a subscription-free, open cloud architecture that delivers push notifications only upon critical events, preserving caregiver attention; (4) **flexible bulk storage** with automated singulation; and (5) **edge-first offline resilience** through a hardware RTC and local schedule cache.

The MedBox system developed in this dissertation addresses this gap through a novel combination of: a rotating-cylinder singulation mechanism that provides deterministic, digitally controllable single-unit dispensing; a gravimetric drop-detection pipeline based on an HX711 load cell; a firmware architecture with local schedule cache and RTC fallback for offline resilience; a server-side Node.js notification pipeline that delivers push notifications independently of the mobile app's foreground status; and a dual-interface Flutter application designed in accordance with inclusive design principles for both caregiver and elderly user profiles. The system architecture, design decisions, and experimental validation of these components are described in the following chapters.

Chapter 3

Experimental Development

This chapter describes the development of the MedBox prototype, from the initial requirements-gathering process to the final mechanical, electronic, and software implementation. The chapter is organised to reflect the order in which these decisions were taken in practice: it begins with the field input that grounded the system requirements (Section 3.1), proceeds to the overall system architecture and the technological choices that follow from it (Section 3.2), and then details the three subsystems built on top of that architecture, namely the mechanical dispensing mechanism (Section 3.3), the electronics and firmware that control it (Section 3.4), and the mobile application and notification backend through which the caregiver interacts with the system (Section 3.5). Throughout the chapter, design decisions are linked explicitly back to the requirements and field observations that motivated them, so that the reasoning behind each choice, and not only the choice itself, is made explicit.

Figure 3.1 shows the final, three-compartment MedBox prototype resulting from this development process, before the individual design decisions behind it are discussed in detail below.



Figure 3.1: The final, three-compartment MedBox prototype: rotating-cylinder dispensing mechanism, shared collection platform with load cell, and control electronics.

3.1 Requirements Elicitation: Interviews with Pharmacists

Prior to consolidating the requirements presented in Section 3.2 and the design decisions discussed throughout this chapter, two semi-structured interviews were conducted with community pharmacists. The goal of these interviews was not to perform a formal qualitative study, but to ground the engineering decisions of the project in the day-to-day reality of medication management for elderly patients, as observed by healthcare professionals who routinely interact with this population.

The two interviewees were deliberately chosen to represent complementary perspectives. The first interview was conducted with Ema Paulino, community pharmacist and President of the Associação Nacional das Farmácias (ANF, the Portuguese National Pharmacies Association), who provided a perspective informed by national-level pharmaceutical policy and by the operation of community pharmacies in an urban setting. The second interview was conducted with Isabel Luz, Technical Director of Farmácia Rainha do Douro, a community pharmacy in Carrazeda de Ansiães, a rural municipality in the district of Bragança, who provided a ground-level perspective on

medication management among an elderly population in a low-density and rural region, a context in which issues of adherence, isolation, and dependence on caregivers tend to be particularly pronounced.

The topics discussed and the resulting insights are summarised below, and are referenced throughout the remainder of this chapter wherever they directly informed a design decision.

3.1.1 Compartmentalisation Strategy

One of the central open questions at the time of the interviews was whether the dispenser should organise its compartments *per medication type* or *per individual dose* (i.e. one pre-filled compartment for each scheduled intake, as in many commercial pill organisers). Both pharmacists noted that a per-dose organisation offers more flexibility when a patient is prescribed a large number of medications with different dosing schedules, but that it can also be considerably more laborious to set up and maintain: a weekly schedule with, for example, up to 5 intakes per day across 7 days would require at least 35 individual compartments to be filled, checked, and refilled every week. A per-type organisation, in contrast, was considered easier to operate, at the cost of requiring the dispensing mechanism to reliably isolate a single unit from a larger reservoir at the time of administration. This trade-off directly motivated the compartmentalisation strategy adopted for the MedBox prototype, discussed in Section 3.3.3..

3.1.2 Scale of Polypharmacy and Number of Compartments

A practical concern raised during the interview with Ema Paulino was the number of compartments that a final product would need to support. While extreme cases of polypharmacy exist (she described, as a relatively rare but real scenario, elderly patients taking up to 15 different medications across the course of a week) these cases would be difficult to accommodate in a compact, single-unit domestic device, and were considered an edge case better addressed in future iterations rather than a hard constraint on the present prototype. For the large majority of elderly patients under regular polypharmacy, however, Ema Paulino estimated that approximately 10 compartments in a final product would be sufficient to cover typical regimens. This figure provides a useful target for sizing a future production-ready version of the MedBox, and is referenced again in the discussion of future work in Section 5.1.

3.1.3 Dispensing Mechanism and Fail-Safes

When asked about the acceptability of an automated single-unit dispensing mechanism, both pharmacists confirmed that the central challenge — and the central value proposition — of such a device lies precisely in reliably dispensing *one and only one* medication unit per scheduled event, since both under- and over-dispensing carry clinical risk. The inclusion of a physical fail-safe mechanism capable of isolating the medication reservoir in case of a missed collection was explicitly endorsed as a desirable safety feature. This observation reinforces the design rationale for the hatch mechanism discussed in Section 5.1 as a priority item for future work.

3.1.4 Alarm Behaviour and Escalation

Regarding notifications and reminders, the interviews highlighted that elderly patients commonly forget to take their medication regardless of reminder mechanisms already in place (e.g. alarm clocks, notes left in visible locations), and that the effectiveness of a reminder is highly dependent on the individual and their daily routine. It was also noted that an alarm that the patient is unable to dismiss is particularly important: if no interaction is detected within a defined window (the pharmacists suggested approximately 15 minutes as a reasonable threshold), the system should escalate the alert in a manner comparable to a morning alarm clock. This observation aligns with, and provides field validation for, the progressive acoustic escalation policy implemented in the firmware and described in Section 3.4.4. The clinical relevance of alarm-equipped medication dispensers for this population is further supported by Vieira et al. [70], who reported a significant improvement in medication adherence among elderly hypertensive patients following the introduction of an electronic medication organiser with an integrated alarm, with the proportion of adherent patients increasing from 18.8% to 96.9% after intervention.

3.1.5 Storage Conditions

Both pharmacists emphasised that medications which cannot tolerate variations in temperature or humidity are, in current pharmacy practice, kept out of standard pre-organised pillboxes (*blisters* or weekly organisers) and are subject to specific storage lists. This reinforced the relevance of the environmental monitoring functionality of the MedBox (temperature and humidity sensing, described in Section 3.4.2.4), and of the corresponding alert thresholds, as a feature with direct clinical grounding rather than a purely technical add-on.

3.1.6 Limitations of Current Weekly Pillbox Solutions

Isabel Luz, drawing on her experience preparing weekly medication organisers for elderly patients in Carrazeda de Ansiães, identified several practical shortcomings of the solutions currently used in Portugal, of the type shown in Figure 3.2. She estimated that in approximately 30% of cases, medications placed into the daily compartments of these weekly organisers remain partially enclosed in their original packaging (e.g. small portions of blister packs), rather than being fully removed as loose units. This occurs because some blister packaging is difficult to open and is left partially intact by whoever prepares the organiser, which in turn makes it harder for the patient to remove the medication independently. She further noted that the printed labels on these organisers typically use small fonts, and that the lids of the daily compartments themselves can be difficult to open for patients with reduced manual dexterity (a population that overlaps significantly with the target users of the MedBox). In her own words, the current state of these solutions in Portugal is, in practice, far from adequate. These observations support the motivation for a dispensing mechanism that delivers a single, fully separated medication unit ready for immediate ingestion, without requiring the patient to manipulate packaging, open small compartments, or read small print, directly informing the mechanical design choices discussed in Section 3.3.



Figure 3.2: Example of a weekly pillbox organiser of the type currently used to prepare medication for elderly patients, with small individually-labelled daily compartments [71].

3.1.7 Choice of Test Medication

When discussing which medication would be most representative for bench testing, both pharmacists pointed to medications for chronic conditions such as diabetes and hypertension as the most commonly used among elderly patients, and therefore the most representative in terms of real-world relevance. However, these medications typically require a medical prescription, which made them impractical to obtain repeatedly for bench testing throughout the development of the prototype. Paracetamol was therefore adopted as the test medication, being widely available without a prescription, commonly used, and not subject to special storage constraints. Ben-u-ron, a widely available over-the-counter paracetamol brand in Portugal, was the specific product used throughout this dissertation, at both the 500 mg and 1000 mg dosages, precisely for this reason of easy, repeated access, in contrast to the prescription-only medications initially suggested by the pharmacists. This consideration directly informed the choice of test medication described in Section 3.3, where the subsequent decision to use 3D-printed pill substitutes, motivated by dimensional incompatibilities between the printed cylinder and the chosen tablets, is also discussed.

3.1.8 Personalisation of Forgetfulness-Mitigation Strategies

Ema Paulino emphasised that forgetfulness is one of the most common causes of poor adherence among elderly patients, and that there is no single strategy that works universally: effective approaches must be tailored to the individual's routine, living arrangements, and support network.

Examples mentioned included setting alarms on a mobile phone (when the patient is comfortable using one), involving a spouse or family member in the daily routine, and physically placing medication in the room where the patient spends most of their time (frequently the kitchen or bedroom). She stressed that solving the adherence problem goes well beyond technological solutions alone, although technology can provide meaningful support within a broader, personalised strategy. This view is consistent with the design philosophy of the MedBox, which combines automated dispensing with caregiver-facing notifications (Section 3.5.4) rather than positioning the device as a stand-alone solution.

3.1.9 Therapeutic Adherence as a Need Statement

Finally, both interviews independently converged on the observation that there is currently no mechanism that genuinely supports therapeutic adherence in elderly patients living at home, and that poor adherence represents both a significant clinical risk and a substantial avoidable cost to the healthcare system. Isabel Luz further reinforced this point, stating that any contribution toward improving therapeutic adherence is welcome, given how concerning the problem is and how much avoidable expenditure results from it. This observation reinforces the need statement underlying the present work, and underscores that any mechanism, even a partial one — that improves the reliability of medication intake at home represents a meaningful contribution.

The remainder of this chapter describes how these field observations were translated into concrete engineering decisions, starting with the overall system architecture (Section 3.2) and the mechanical design of the dispensing mechanism (Section 3.3).

3.2 System Architecture

This section presents the overall architecture of the MedBox system, describing how its different subsystems (the physical dispenser, the cloud infrastructure, the mobile application, and the notification service) relate to one another. It also introduces the user profiles considered, the main use cases, and the functional and non-functional requirements that guided the design decisions described in the following sections. The section concludes with a justification of the main technological choices, grounding the options taken against the alternatives that were considered.

3.2.1 System Overview

The MedBox system follows a distributed architecture organised into four layers, structured around a central cloud database that acts as the synchronisation point between all components (Figure 3.3):

1. **Hardware Layer (Edge):** the physical dispenser, controlled by an ESP32-C6 microcontroller, responsible for sensor readings, actuation of the dispensing motors, and Wi-Fi communication with the cloud;

2. **Persistence and Communication Layer (Cloud):** the Firebase Realtime Database (RTDB), which centralises the system state (schedule, inventory, sensors, history, alerts), and Firebase Cloud Firestore, which hosts the medication catalogue;
3. **Application Layer (Client):** the mobile application developed in Flutter, which offers the caregiver the full control of the MedBox;
4. **Service Layer (Backend):** a dedicated Node.js server that continuously monitors the system state in the cloud and triggers push notifications via Firebase Cloud Messaging (FCM) whenever an anomaly occurs.

This layered separation was deliberate and has direct implications for the robustness of the system: none of the layers depends on a direct, persistent connection to the others. The ESP32-C6 firmware maintains a local cache of the daily schedule, allowing it to continue operating, including dispensing medication at the scheduled times, even in the absence of an Internet connection (see Section 3.4.4 in Section 3.4). Similarly, the mobile application does not need to be in the foreground, or even open, for the user to be notified of an anomaly, since this responsibility is entirely delegated to the backend (Section 3.5.4 in Section 3.5).

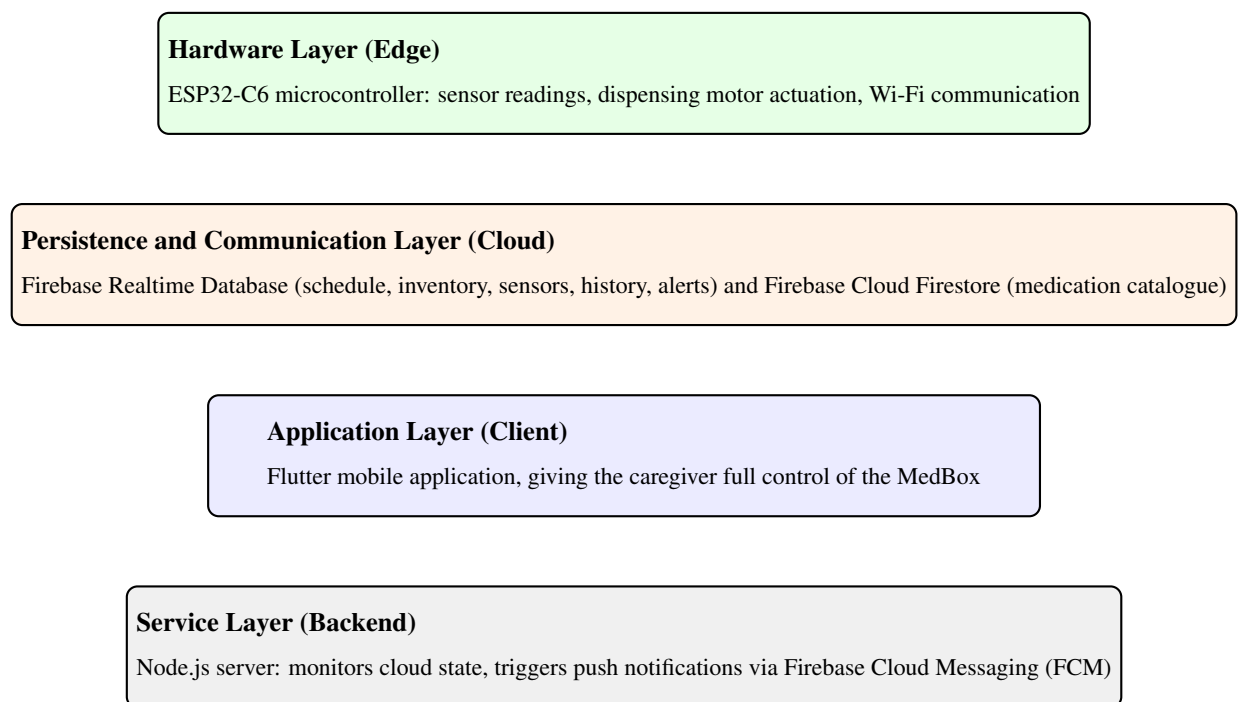


Figure 3.3: Overview of the layered architecture of the MedBox system: Hardware (Edge), Persistence and Communication (Cloud), Application (Client), and Service (Backend).

The typical data flow of the system is illustrated in Figure 3.4. The caregiver configures the weekly medication schedule and environmental thresholds through the app, which writes this information to the Firebase RTDB. The ESP32-C6 firmware synchronises the relevant nodes to a local cache and, at the scheduled time, performs the dispensing routine, confirming the pill drop

via the load cell and writing the outcome back to the cloud. In parallel, the firmware continuously publishes environmental readings and updates the system’s alert flags.

It is important to note that the resulting state changes reach the caregiver through two complementary paths. While the app is open, the caregiver observes these changes directly through the RTDB streams, typically within less than 5 seconds (NFR02), without relying on any push notification. The Node.js server and the FCM-based path become relevant precisely when the app is not in the foreground: upon detecting an anomalous condition through its persistent listeners, the server retrieves the caregiver’s notification token and sends a push notification, ensuring that critical alerts are not missed even when the app is closed.

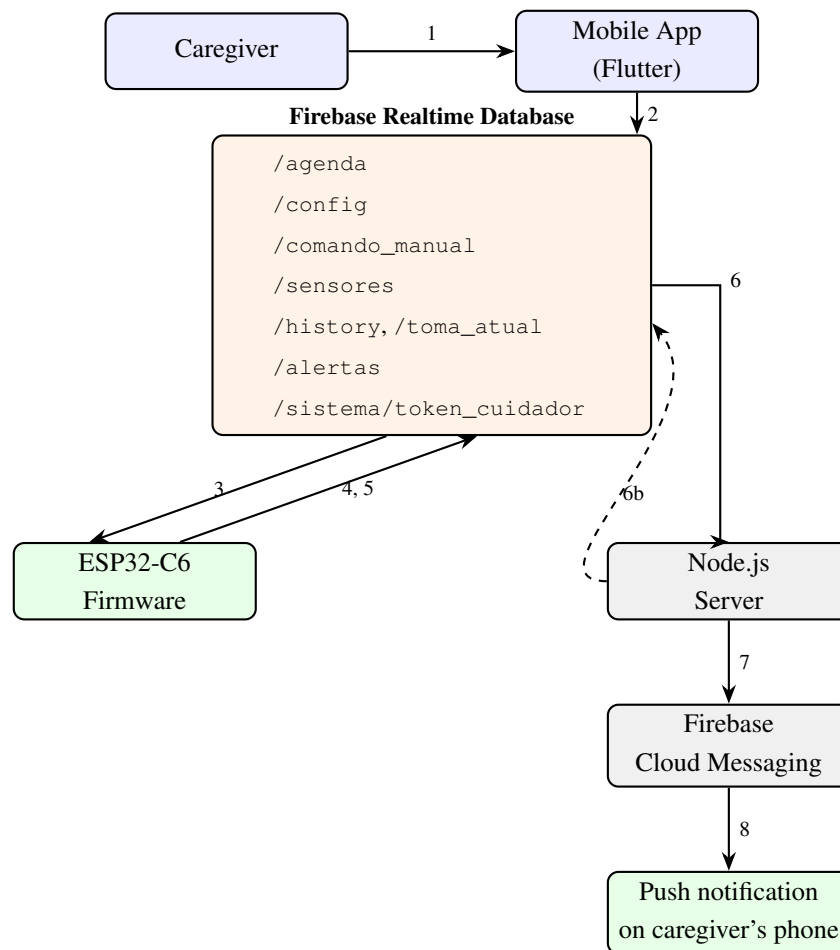


Figure 3.4: End-to-end data flow of the MedBox system, from schedule configuration by the caregiver to the dispensing event and the resulting push notification in case of an anomaly. Numbered steps are described in the accompanying text. Dashed arrows represent one-off reads rather than persistent listeners.

Figure 3.4 illustrates this flow step by step. The caregiver opens the app and defines a weekly schedule or environmental thresholds (step 1), and the app writes this information to `/agenda`, `/config`, and, in the case of an anticipated dispensing request, to `/comando_manual` (step

2). The ESP32-C6 firmware reads these same nodes (step 3) and, at the scheduled time, triggers the servo, confirms the drop via the load cell, waits for collection, and writes the outcome to `/history`, `/toma_atual`, and `/alertas` (step 4), while continuously publishing sensor readings to `/sensores` in parallel (step 5).

On the notification side, the Node.js server maintains two persistent listeners, one on `/alertas` for clinical alerts and another on `/sensores/dht11_interno/ultima_leitura` as a heartbeat signal, together with an internal watchdog timer that flags a connectivity loss after 25 seconds without a heartbeat (step 6). When a notification needs to be sent, the server performs a one-off read of the caregiver's token from `/sistema/token_cuidador` (step 6b, shown as a dashed arrow in the figure, since this is a single read rather than a persistent listener). On detecting an anomalous condition, the server sends a push request to Firebase Cloud Messaging (step 7), which in turn delivers the notification to the caregiver's device (step 8).

3.2.2 User Profiles and Use Cases

The system was designed to serve two user profiles with distinct needs, contexts of use, and levels of digital literacy.

3.2.2.1 Caregiver Profile

The caregiver, typically a family member or informal care provider, is the user responsible for configuring and remotely supervising the system. Access to this profile is protected by authentication (Firebase Authentication, e-mail/password), reflecting the sensitivity of the data managed (health and clinical information about a third party). The main features available to this profile are:

- **Schedule:** creation, editing, and removal of medication times, with support for daily repetition or repetition on specific days of the week, configuration of the number of units per dose, and definition of a tolerance window (*delay*) for collection;
- **Inventory:** association of a medication, searched from the integrated Infomed catalogue, with each physical compartment of the dispenser;
- **Monitor:** real-time visualisation of the temperature and humidity inside the dispenser, with an indication of the connectivity status of the sensor;
- **History:** chronological consultation of all dispensing events, with classification of the outcome of each dose (*taken, missed, failure, device_offline*);
- **Settings:** configuration of the temperature and humidity thresholds that trigger alerts, dynamically read by the firmware.

3.2.2.2 Elderly User Profile

An elderly user profile was designed, at the requirements stage, according to inclusive design principles, on the assumption that the end user may have visual impairments, reduced motor dexterity,

or limited familiarity with technology. This profile was not implemented or tested in the current prototype; it is documented here as the intended design and is revisited as a priority item for future work (Section 5.1). The intended design called for an interface that:

- **requires no authentication by default:** access is direct, removing the login barrier; an optional biometric authentication method (fingerprint or facial recognition) is considered as a future, caregiver-configurable alternative for households where an extra layer of access control is desired, without making it a requirement for the typical user;
- presents a minimalist interface, with enlarged typography, focused exclusively on immediately relevant information: the next scheduled dose (medication, time, and compartment) and any active alerts (overdue dose, temperature or humidity outside limits);
- does not expose configuration features, avoiding the risk of accidental changes to the medication schedule.

3.2.2.3 Use Cases

Figure 3.5 presents the use case diagram of the system. The main use cases include:

- **UC1: Configure medication schedule** (Caregiver): definition of time, medication, dosage, and repetition days;
- **UC2: Manage inventory** (Caregiver): association of medications with physical compartments;
- **UC3: Check system status** (Caregiver): visualisation of the next dose, environmental sensors, and active alerts;
- **UC4: Request anticipated dispensing** (Caregiver): sending a manual command for immediate dispensing, outside the scheduled time, without duplicating the dose programmed for that day;
- **UC5: Receive anomaly alert** (Caregiver): automatic push notification upon anomalous conditions, without requiring active interaction with the application;
- **UC6: Dispense medication** (System/Dispenser): autonomous execution of the dispensing routine at the scheduled time, including drop confirmation and collection confirmation.

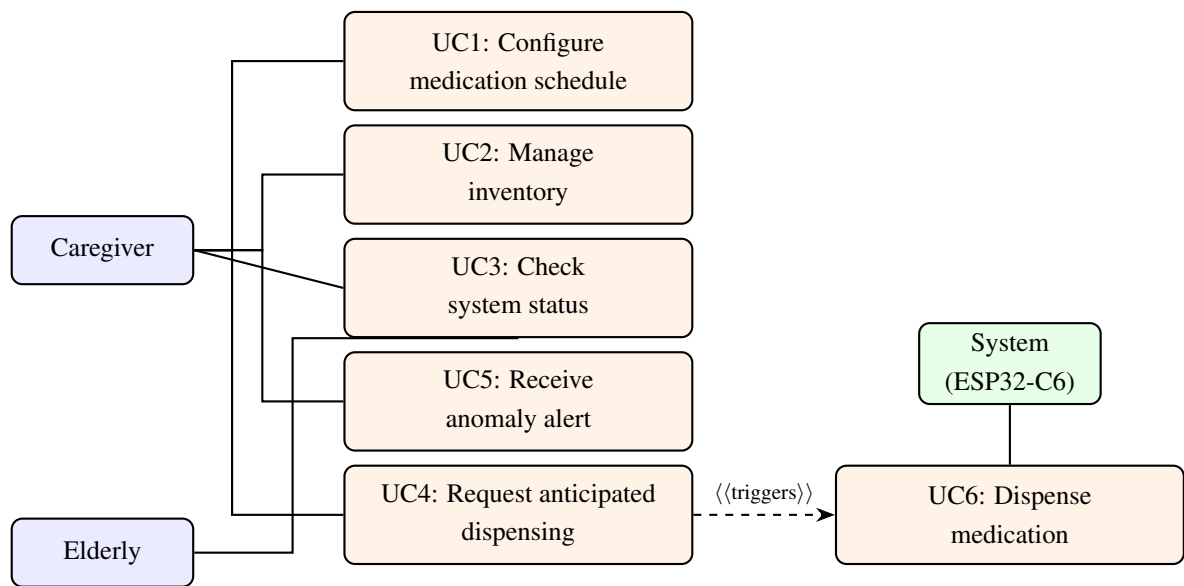


Figure 3.5: Use case diagram of the MedBox system. The dashed arrow indicates that an anticipated dispensing request (UC4) directly triggers the dispensing routine (UC6), bypassing the regular schedule.

A detailed description of the two use cases most directly relevant to the system’s core dispensing and safety logic, UC6 (Dispense Medication) and UC4 (Request Anticipated Dispensing), including preconditions, main flow, postconditions, and alternative flows, is provided in Appendix A (Tables A.1 and A.2).

3.2.3 System Requirements

The definition of the system requirements resulted from an iterative process, in which initially envisioned requirements were later revised in light of hardware, time, and scope constraints typical of a master’s dissertation prototype (this evolution is discussed in detail in Sections 3.3 and 3.4). The full functional and non-functional requirements considered for the final prototype are listed in Appendix B (Tables B.1 and B.2); they are summarised in Figure 3.6, grouped by theme, and referenced by ID throughout the remainder of this chapter.

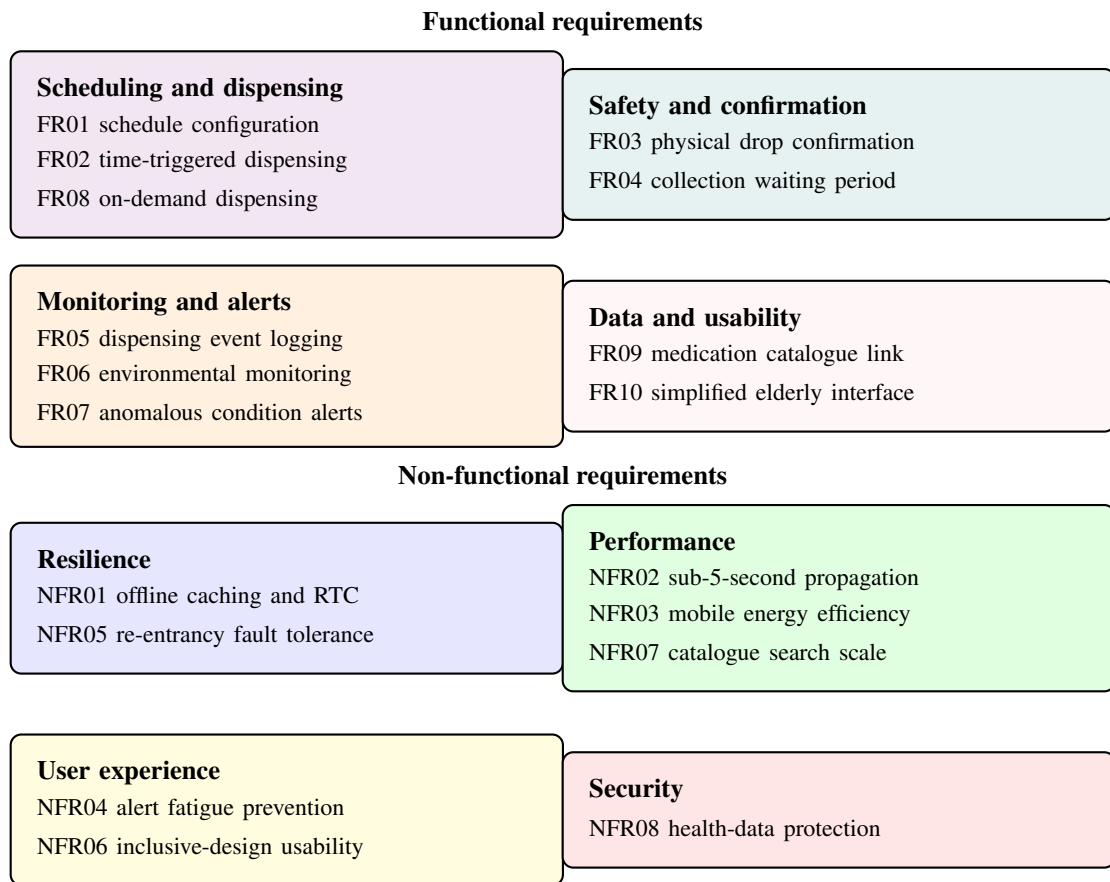


Figure 3.6: The eighteen functional and non-functional requirements of the MedBox system, grouped into eight thematic categories.

As noted in Appendix B, FR10 and NFR06, both scoped for the elderly-facing interface, were not implemented or tested within the current prototype and are deferred to future work (Section 5.1). NFR08 is likewise only partially met by the current prototype: as discussed in Section 4.9 (Limitations), the Firebase Realtime Database rules used during development grant unrestricted read and write access, a deliberate simplification adopted to accelerate prototyping that is identified as a priority item for future work.

3.2.3.1 Cloud Platform: Firebase Realtime Database

The choice of Firebase Realtime Database as the central persistence and communication layer was motivated by three factors. First, its nature as a streaming-oriented NoSQL database allows both the ESP32-C6 firmware and the Flutter application to react to state changes in real time, without the need to implement a polling mechanism, which would be computationally expensive on both the microcontroller and the smartphone. Second, the official SDK for Arduino/ESP32 (`Firebase_ESP_Client`) offers direct integration with the RTDB, significantly reducing implementation complexity compared to alternatives such as a self-hosted MQTT broker, which would require additional infrastructure and its own security and availability management. Third,

the RTDB integrates natively with Firebase Authentication and Firebase Cloud Messaging, allowing the authentication of the IoT device, the authentication of the caregiver, and the sending of push notifications to share the same ecosystem, without the need to manage multiple platforms and their respective credentials.

As an alternative, the use of a relational database (e.g. PostgreSQL) hosted on a dedicated server with an intermediate REST API was considered. This approach was discarded at the prototyping stage because it would imply a significantly higher development and infrastructure maintenance effort, without proportional benefits for the scale and requirements of a proof-of-concept system.

3.2.3.2 Mobile Development Framework: Flutter

The mobile application was developed in Flutter, a choice made over alternatives such as native development (Kotlin/Swift) or React Native. The main reasons were:

- **Single codebase:** Flutter allows generating applications for Android and iOS from a single codebase, which is particularly relevant for a dissertation project with limited resources and time;
- **Native Firebase integration:** the `firebase_database` package offers reactive streams (`onValue`) that map directly onto the RTDB data model, simplifying the construction of interfaces that update automatically in response to changes in the cloud;
- **Maturity of the widget ecosystem:** the availability of components such as `CupertinoPicker`, `showModalBottomSheet`, and `PopupMenuButton` made it possible to quickly build consistent interfaces for managing the schedule and inventory, without the need to develop components from scratch.

3.2.3.3 Microcontroller: ESP32-C6

The selection of the ESP32-C6 as the central control unit of the dispenser resulted from a comparison with two alternatives widely used in prototyping projects: the Arduino (no native Wi-Fi variants, requiring additional communication modules) and the Raspberry Pi (whose processing capacity and full operating system constitute overengineering for the task of periodically reading sensors and actuating servo motors). The ESP32-C6 offers native 802.11 b/g/n Wi-Fi connectivity, a low-power RISC-V architecture, and a sufficient number of GPIO pins to simultaneously support up to four servo motors, a load cell (HX711 interface), a temperature and humidity sensor (DHT11), a real-time clock (DS3231 RTC via I2C), and an acoustic actuator (buzzer), as detailed in Section 3.4.

3.2.4 Summary

The architecture presented in this section establishes the foundations on which the developments described in the following sections are built: the mechanical design of the dispensing mechanism

(Section 3.3), the implementation of the firmware and associated electronics (Section 3.4), and the mobile application and notification service (Section 3.5). The separation into decoupled layers, communicating through a central cloud database, proved essential to the resilience of the system in the face of connectivity loss, mechanical failure, and mobile application shutdown scenarios. These scenarios are discussed in detail in the experimental evaluation presented in Section 4.

3.3 Mechanical Development

This section describes the mechanical design and construction of the MedBox dispensing mechanism. It covers the physical requirements that guided the design, the three approaches considered over the course of the project, the rationale behind the compartmentalisation strategy from a mechanical standpoint, the final SolidWorks design in detail, the 3D printing process, the decision to use 3D-printed pill substitutes for the initial bench testing, and the extension of the validated single-compartment mechanism into the final three-compartment assembly.

3.3.1 Physical Requirements of the Dispensing Mechanism

Building on the functional requirements introduced in Section 3.2.3 (in particular FR02 and FR03), the dispensing mechanism was required to satisfy the following physical constraints:

- **Single-unit dispensing:** the mechanism must release exactly one medication unit per dispensing event, since both under- and over-dispensing carry clinical risk, as confirmed in the interviews described in Section 3.1;
- **Compactness:** the mechanism must fit within a domestic, countertop-sized enclosure, in contrast to the large-scale dispensing systems typically found in hospital settings;
- **Multi-compartment support:** the overall design must accommodate multiple independent dispensing units, one per medication type, as discussed in Section 3.3.3;
- **Manufacturability via FDM:** the mechanism must be producible using fused deposition modelling (FDM) 3D printing, the manufacturing process available for this project, which imposes constraints on tolerances, overhangs, and part geometry.

3.3.2 Approaches Considered

Three distinct mechanical approaches were considered over the course of the project. They are presented here in the order in which they were explored, together with the reasoning that led from one to the next.

3.3.2.1 Flap-Hatch Mechanism (*Palheta-Alçapão*)

The first approach, and the one followed for the longest period, until close to the point at which 3D printing became feasible, consisted of a flap or hatch (*alçapão*) positioned beneath the medication

reservoir (Figure 3.7). A servo motor would rotate to open this hatch, allowing gravity to drop a medication unit into a collection tray below.

At this stage, bench testing focused on validating the electronics and the communication pipeline (servo, ESP32-C6, Firebase, and the mobile app) rather than the singulation behaviour of the mechanism itself. Coins were used on the bench to simulate the falling of a medication unit, and the servo rotation simulated the opening of the hatch, but the actual ability of the flap geometry to reliably release a single unit was not yet tested physically. During this phase, all sensors intended for the final system, including the HX711 load cell, the DHT11 temperature and humidity sensor, and the DS3231 RTC, were individually tested and integrated with the ESP32-C6 and the Firebase pipeline.

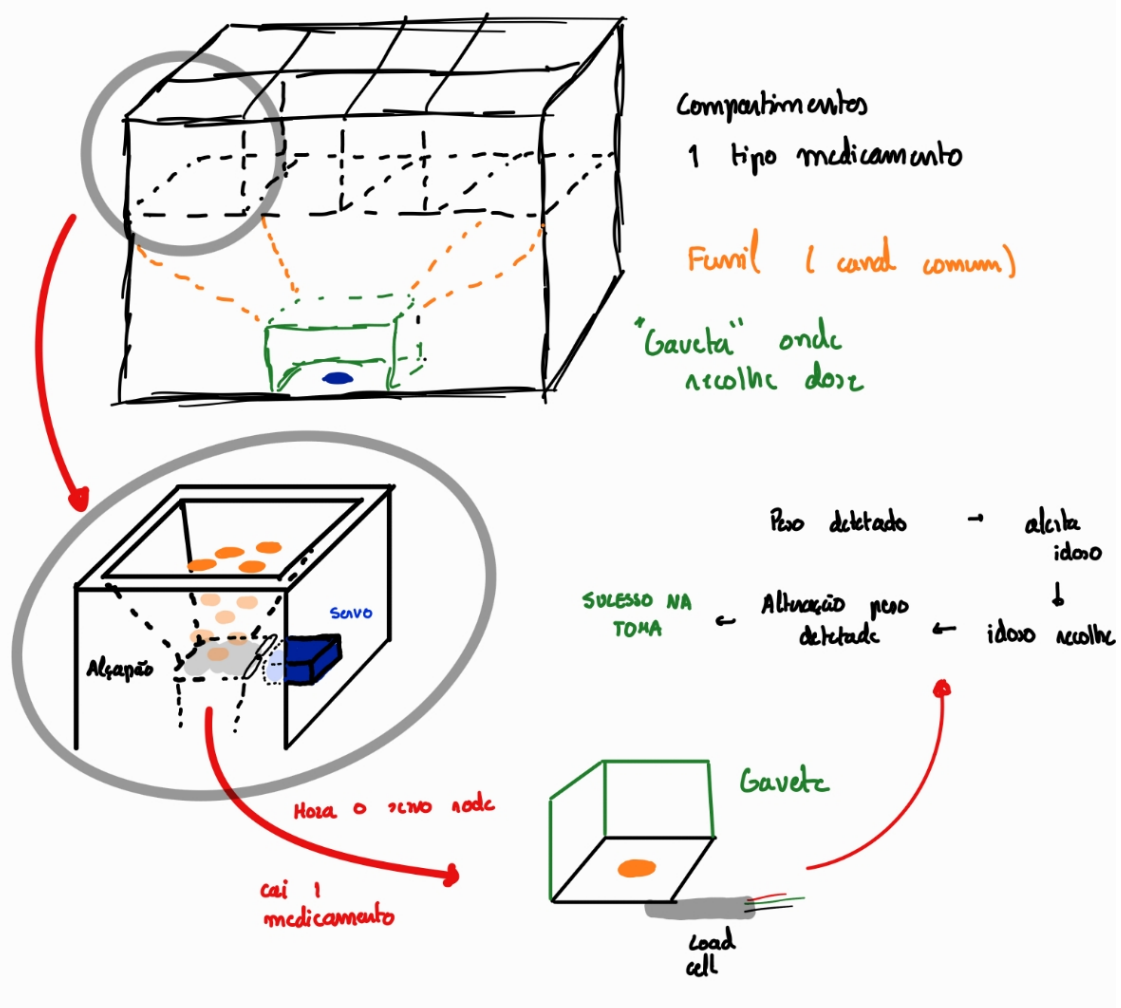


Figure 3.7: Early concept sketch of the flap-hatch dispensing mechanism, made during the initial ideation phase.

A variation of this approach was also discussed with the project supervisor (Prof. Vítor Hugo): instead of a hatch that opens once and remains open, a lower gate would open and close very rapidly, with the intention of letting only a single medication unit fall through before closing

again. This idea was never developed beyond the conceptual stage, as it did not appear likely to be 100% reliable, since the timing required to guarantee that exactly one unit (and not zero or two) passes through a fast-opening gate would depend heavily on the size, shape, and yielding packing of the units above it, factors that are difficult to control or predict.

This approach was ultimately abandoned after a discussion with a PhD student in mechanical engineering (Diogo Gomes), who pointed out that reliably dispensing exactly one medication unit through a flap-hatch geometry is considerably more difficult than it appears, since the behaviour of a stack or pile of units above an opening is hard to control without additional mechanical features (e.g. agitation, indexing, or a singulation channel). This observation directly motivated the move to the rotating cylinder approach described in Section 3.3.2.3.

3.3.2.2 PICK-SHAKE-DROP

A second approach, identified during a review of existing dispensing mechanisms in the literature, was the PICK-SHAKE-DROP concept, in which a vibrating platform agitates a small reservoir of medication units until a single unit falls through a calibrated opening. This approach was not prototyped: it was discarded at an early stage on the grounds of mechanical complexity (the need for a vibration actuator and a tuned platform) and poor compatibility with FDM-printed parts, whose surface friction and tolerances make vibration-based singulation unpredictable without extensive calibration.

3.3.2.3 Rotating Cylinder with Servo Actuation

The third and final approach, adopted for the prototype, consists of a cylinder with a single slot sized to receive one medication unit, mounted inside a housing and rotated by a servo motor (Figure 3.8). In its resting position, the slot is aligned with a reservoir above the cylinder; when the servo rotates the cylinder by approximately 180°, the slot carries the medication unit to the bottom of the housing, where it is released through a funnel onto a collection platform; the servo then returns the cylinder to its resting position, ready to receive the next unit.

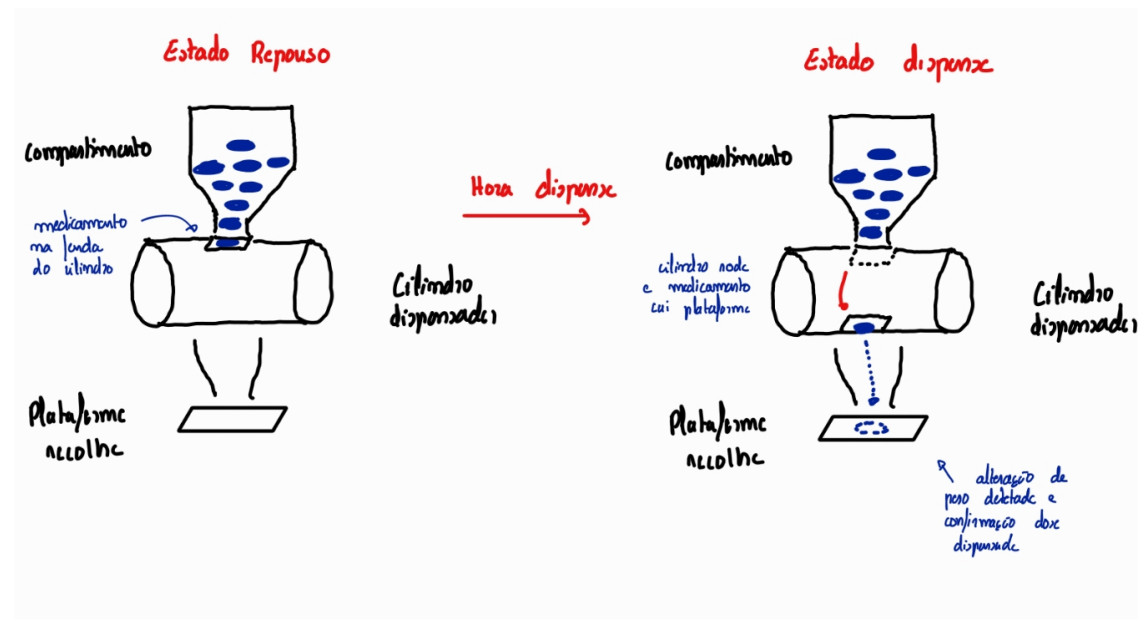


Figure 3.8: Informal design sketch of the operating principle of the rotating cylinder mechanism: in its resting position, the cylinder’s slot is aligned with the funnel and receives a medication unit from the reservoir; the servo then rotates the cylinder by approximately 180°, carrying the unit to the bottom of the housing where it is released onto the collection platform, before the cylinder returns to its resting position.

This principle is not new: a closely related mechanism is described in patent CA2417103C, *Variable Size Pill Dispenser* [72], filed in 2003, which describes a wheel with slots dimensioned to receive a single tablet, capsule, or caplet, connected to a motor through a gear mechanism. That patent was developed for hospital-scale, high-throughput dispensing systems. The relevance of this prior art to the present work is twofold: first, it shows that the underlying principle, isolating a single unit using a rotating slotted element, works and has been used at industrial scale for over two decades; second, it points to a gap that the present work addresses. Existing consumer-oriented automatic pill dispensers also organise their reservoirs by medication type, similarly to the approach adopted here (Section 3.3.3), but rely on considerably more complex singulation mechanisms: the Hero dispenser uses a vacuum-tube mechanism to retrieve a unit from the cartridge assigned to each medication [5], while the MedaCube relies on an internal pick-and-place mechanism to retrieve individual pills from bulk containers [6]. The gap addressed by the present work is therefore not the per-type organisation itself, but the adaptation of the simple, mechanically robust rotating-slot principle described in CA2417103C, originally intended for industrial-scale equipment, into a cost-effective, open, FDM-printable implementation built around commodity electronic components, in contrast to the proprietary, mechanically complex (vacuum- or robotics-based) singulation mechanisms found in current consumer devices. This supports the need statement introduced in Section 3.1.

The rotating cylinder approach was selected for three reasons. First, it provides deterministic, digitally controllable singulation: the cylinder either is or is not in the resting position, with no

intermediate states to calibrate. Second, it has no moving parts beyond the cylinder itself and the servo, which is favourable for FDM manufacturing. Third, it scales naturally to the multi-compartment, per-medication-type strategy described in Section 3.3.3, since each compartment can house an identical cylinder unit driven by its own servo, a property that was subsequently confirmed in practice when the design was extended from one to three compartments (Section 3.3.9).

3.3.3 Compartmentalisation Strategy: Mechanical Implications

The decision to organise the dispenser per medication type rather than per individual dose, motivated from a usability standpoint in Section 3.1, also has direct mechanical consequences that reinforced the choice of the rotating cylinder mechanism.

Under a per-dose organisation, as discussed in Section 3.1, a weekly schedule with up to 5 intakes per day would require at least 35 individual compartments. Even though the mechanism required for each individual cell could in principle be simple (e.g. a trapdoor releasing the cell's contents), replicating any mechanism, however simple, 35 times represents a substantial increase in total part count and assembly effort for a domestic device, compared to the handful of compartments required under a per-type organisation.

Under the adopted per-type organisation, the number of compartments corresponds to the number of distinct medications the patient takes regularly, typically in the order of a handful for a polymedicated elderly patient (and, per the estimate discussed in Section 3.1, around 10 for a production-ready device covering most cases). Each compartment is mechanically self-contained: a reservoir, a rotating cylinder, a servo, and a funnel, replicated identically across compartments. This modularity means that the design effort invested in a single compartment, described in the remainder of this section, applies directly to every compartment of the device, without requiring per-compartment mechanical redesign, and is precisely the property exploited when the prototype was extended to three compartments in Section 3.3.9.

Table 3.1 summarises the trade-offs between the two organisational strategies from a mechanical standpoint, complementing the usability-oriented discussion in Section 3.1.

Table 3.1: Mechanical trade-offs between per-type and per-dose compartmentalisation strategies.

	Per-type (adopted)	Per-dose
Number of compartments	Equal to the number of distinct medications: a handful for a typical elderly patient, ~ 10 for a production device (main advantage)	At least 35 for a weekly schedule with up to 5 intakes/day
Mechanism per compartment	Requires an active singulation mechanism (rotating cylinder) to isolate one unit from a bulk reservoir	Mechanically simpler in principle: each cell holds a single pre-measured dose, and a basic trapdoor opening the bottom of the cell is sufficient to release its contents
Refilling effort	Refilled in bulk per medication, less frequently	Refilled individually per cell, typically weekly, by the caregiver or pharmacist
Tablet-size flexibility	Each compartment's slot must be sized for the medication it holds; this was resolved for the three compartments implemented in this prototype (Section 3.3.9) by dimensioning each cylinder individually for its target tablet	Not a constraint at the mechanism level: a trapdoor cell can release units of any size or shape regardless of dimensions

Table 3.1 shows that the per-dose strategy has advantages at the level of an individual compartment: a simpler mechanism, no singulation requirement, and no size constraint. These advantages, however, do not scale: a per-dose device would need to replicate its (simple) mechanism roughly 35 times, an increase in part count and assembly effort with no equivalent engineering solution. The challenges of the per-type strategy, by contrast, are concentrated in a single, bounded engineering problem: designing one reliable single-unit singulation mechanism, replicated once per medication. This is precisely the problem addressed by the rotating cylinder mechanism developed in Section 3.3.2.3, and, once solved for one compartment, the same solution was shown to apply, with per-medication resizing of the cavity dimensions, to the two additional compartments described in Section 3.3.9.

3.3.4 Scaling to Additional Compartments: Fixed versus Medication-Dependent Parameters

The mechanical design described in this section was first developed and validated for a single compartment, sized for the test medication used throughout the initial phase of this dissertation, before being extended to the three-compartment assembly described in Section 3.3.9. Extending the design to additional compartments did not require redesigning the dispensing mechanism from scratch. Instead, it required distinguishing which design parameters are independent of the specific medication, and therefore directly reusable, from which parameters depend on the dimensions of the tablet being dispensed and had to be recalculated for each medication.

Table 3.2 summarises this distinction, specifying for each parameter which physical component of the mechanism it belongs to. The fixed parameters, the funnel wall angle of the embedded body funnel, the cylinder-to-cavity clearance, and the servo coupling and mounting, are governed either by the operating principle of the mechanism itself (Section 3.3.2.3) and by general material and manufacturing constraints (Section 3.3.6), and, in the specific case of the secondary collector funnel's base radius, by a single dimension chosen to accommodate the largest tablet among the three medications handled by the device, rather than by the medication assigned to any individual compartment. The variable parameters, namely the depth, width, and axial length of the cylinder's slot (its cavity), were originally sized directly from the dimensions of the Ben-u-ron 500 mg tablet used for the single-compartment prototype, and were subsequently recalculated, following the same sizing criteria, for the two additional compartments (*Ben-u-ron 500 mg* circular tablet and *Ben-u-ron 1000 mg*) described in Section 3.3.9: each cavity dimension was kept close to, but slightly larger than, the corresponding dimension of the tablet it holds, wide enough to admit the tablet comfortably but narrow enough to avoid a second unit entering the same cavity, which would risk future jamming (Section 4.6). In particular, cavity width was kept below twice the tablet width, to preserve the geometric guarantee that only one unit can enter the cavity at a time.

The secondary collector funnel's base radius, which feeds Compartments 2 and 3 into the main tunnel (Section 3.3.5), was not resized per compartment. Instead, a single radius was used across all three compartments, dimensioned to accommodate the largest tablet measurements among the three medications: the length of the Ben-u-ron 1000 mg tablet and the width of the Ben-u-ron 500 mg tablet, whichever combination of dimensions proved most demanding.

Table 3.2: Mechanical design parameters: fixed (medication-independent) versus medication-dependent dimensions.

Parameter (component)	Type	Notes
Funnel wall angle, 50° (embedded body funnel, Section 3.3.5)	Fixed	Chosen empirically, through iterative testing, to reliably direct medication units toward the cylinder slot without an excessively tall funnel profile.
Cylinder-to-cavity clearance (cylinder and body)	Fixed per print	Selected empirically per printed batch to balance free rotation against powder ingress; independent of tablet geometry, but sensitive to printer and material tolerances.
Servo coupling, shaft alignment, mounting screws (body and cap)	Fixed	Determined by the servo model and the printed body geometry, not by the medication.
Slot (cavity) depth, cylinder	Medication-dependent	Kept close to, but slightly larger than, the tablet's corresponding dimension, to fit the tablet comfortably while limiting the risk of future jamming. Specific values for each medication are reported in Table 3.3 (Section 3.3.9).
Slot (cavity) width, cylinder	Medication-dependent	Kept close to, but slightly larger than, the tablet width, and always below twice the tablet width, to geometrically prevent two units entering side by side. Specific values reported in Table 3.3.
Slot (cavity) axial length, cylinder	Medication-dependent	Kept close to, but slightly larger than, the tablet length, to seat one unit in the longitudinal orientation. Specific values reported in Table 3.3.
Secondary collector funnel base radius (Section 3.3.5)	Fixed, common to all compartments	Matched to the cylinder diameter; a single radius was used for all three compartments, sized to the largest tablet dimensions among the three medications (Ben-u-ron 1000 mg length, Ben-u-ron 500 mg width).

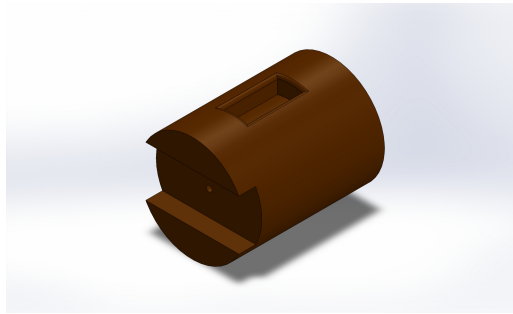
The medication-dependent cavity dimensions, namely the depth, width, and axial length of each cylinder's slot, are reported separately in Table 3.3 in Section 3.3.9, together with the caliper-measured tablet dimensions each of them was sized from.

In practice, this meant that the two additional compartments built for real medication reused the funnel angle, the clearance approach, and the servo coupling design validated for the original compartment, without modification. Only the cylinder's cavity dimensions were recalculated for each of the two new tablet geometries, following the sizing criteria already established for the original compartment, while the secondary funnel's base radius was set once for all compartments, as described above. This localised the redesign effort to a small, well-defined set of parameters, rather than requiring the mechanism's operating principle to be revalidated from scratch for each new compartment. The full SolidWorks technical drawings for the individual parts described in this section are provided in the appendices referenced alongside each part below.

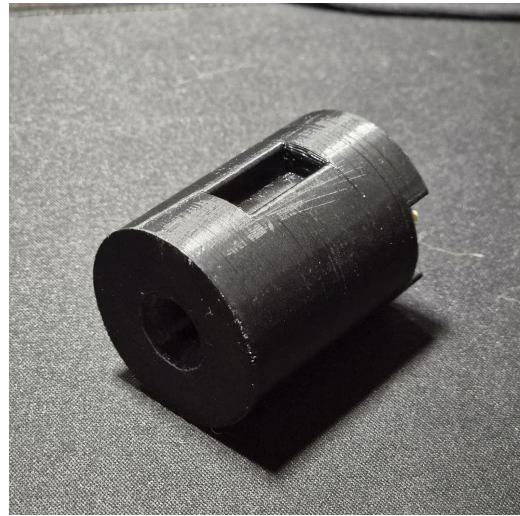
3.3.5 Final Mechanical Design

The mechanical design went through several iterations in SolidWorks before converging on the geometry that was ultimately 3D printed and tested. These earlier iterations are not detailed here, as they were superseded early in the process and add little to the discussion; the remainder of this section describes, on a part-by-part basis, the final design that was printed for the first, single-compartment stage of the prototype, together with the reasoning behind each design decision. This design was subsequently replicated, with medication-specific resizing, into the two additional compartments described in Section 3.3.9.

Cylinder. The rotating element consists of a cylinder with a single slot, or cavity (*fenda*), cut into its outer surface and sized to receive exactly one medication unit (Figure 3.9). In its resting position, this slot aligns with the funnel above it, allowing gravity to settle a unit into the cavity; once the servo rotates the cylinder by approximately 180°, the same slot carries that unit to the release point at the bottom of the housing. On one of the cylinder's flat end faces, a rectangular cut is machined into the cylinder, matching the profile of the servo horn. This design allows the cylinder to connect directly to the servo, which sits at the rear of the assembly, outside the body: the servo horn is inserted straight into the cylinder's lateral cut, so that the servo's rotation is transmitted to the cylinder without any intermediate coupling part, reducing both part count and the number of potential sources of play (*backlash*) in the rotation. The cylinder shown in Figure 3.9 corresponds to the original compartment, sized for the 3D-printed pill substitute (Section 3.3.7); the two additional cylinders built for Ben-u-ron 500 mg and 1000 mg follow the same design, with their slot dimensions resized individually per medication, as detailed in Section 3.3.9. The full technical drawing for this part is provided in Appendix C.1.



(a) SolidWorks design



(b) 3D-printed part

Figure 3.9: The rotating cylinder of the original compartment, shown here sized for the 3D-printed pill substitute: the slot (*fenda*) sized for one medication unit, and the lateral cut matching the servo horn profile, are both visible. (a) SolidWorks design and (b) the corresponding 3D-printed part.

Body, embedded asymmetric funnel, and cap. The body houses the cylinder in a cylindrical cavity and was designed with the servo mounted at the rear, facing into this cavity through a hole in the rear wall (Figure 3.11). Two aspects of this hole were critical. First, its position had to be precisely centred on the cylinder's rotation axis: any offset would cause the cylinder to bind against the cavity walls during rotation, which was, as described in Section 3.3.6, exactly the failure mode observed in the first printed body. Second, the thickness of the rear wall had to account not only for the structural thickness of the wall itself, but also for the depth of the servo body up to the point where the horn protrudes (Figure 3.10), so that the horn would reach the cylinder's lateral cut at the correct depth without the servo housing colliding with the body or leaving an unsupported gap. This dimensioning had to be done jointly: the wall thickness, the servo's body-to-horn distance, and the depth of the lateral cut on the cylinder all had to be consistent with one another for the cylinder to sit correctly inside the cavity while remaining fully engaged with the servo horn.

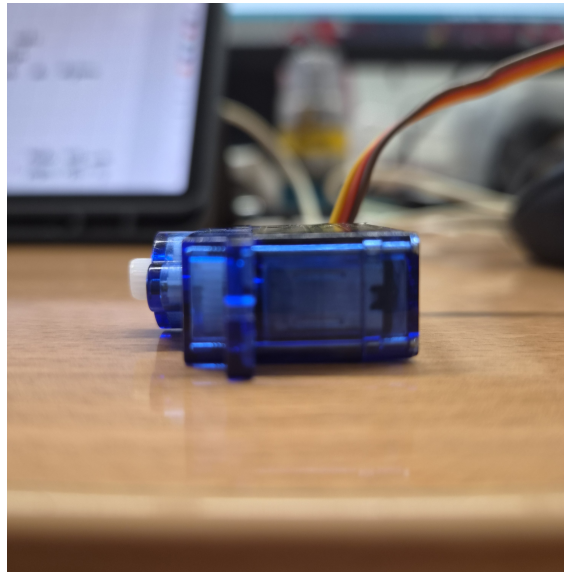


Figure 3.10: Side view of the SG90 servo motor used to actuate the rotating cylinder. The distance from the body of the servo to the tip of the horn, visible here, is the dimension that had to be accounted for when sizing the thickness of the body's rear wall.

An asymmetric funnel is embedded directly in the top of this same body, guiding medication units from the reservoir into the cylinder's slot when the cylinder is in its resting position. The decision to use an asymmetric, rather than a symmetric (conical), funnel geometry is supported by the literature on hopper design for granular and tablet-like materials. Ketterhagen and Hancock [73] used the discrete element method to compare eccentric (asymmetric) feed hoppers with concentric conical hoppers in the context of pharmaceutical tablet presses, and found that eccentric hoppers achieve discharge rates up to 35% higher than concentric conical hoppers of equivalent size, while exhibiting a similar mass-flow/funnel-flow behaviour. More generally, symmetric conical hoppers are prone to the formation of stagnant zones near the walls, which can lead to bridging, the formation of a stable arch of material that halts flow entirely. Eccentric, asymmetric geometries, with one steeper wall directing material toward the outlet, reduce the extent of these stagnant zones and the associated risk of bridging [73]. In the context of the MedBox, each compartment's asymmetric funnel was oriented so that its steeper wall directs medication units toward that compartment's cylinder slot, reducing the likelihood that a unit becomes lodged against a shallow wall above the slot rather than falling into it. The full technical drawing for the body, including the funnel angle and the rear wall dimensioning discussed above, is provided in Appendix C.3.

Finally, a cap closes the body on the side opposite the servo. This cap integrates a bearing and a short support shaft that partially penetrates the cylinder along its rotation axis. This addresses a stability concern with single-point actuation: if the cylinder were supported only at the servo end, its opposite end would be unconstrained, allowing it to wobble radially within the cavity, particularly once a medication unit has shifted its centre of mass off-axis. The shaft and bearing arrangement on the cap helps the cylinder rotate concentrically about its axis at both ends, reducing play during rotation. The technical drawing for the cap is provided in Appendix C.4; the shaft,

modelled and printed as a separate part, is detailed in Appendix C.2.

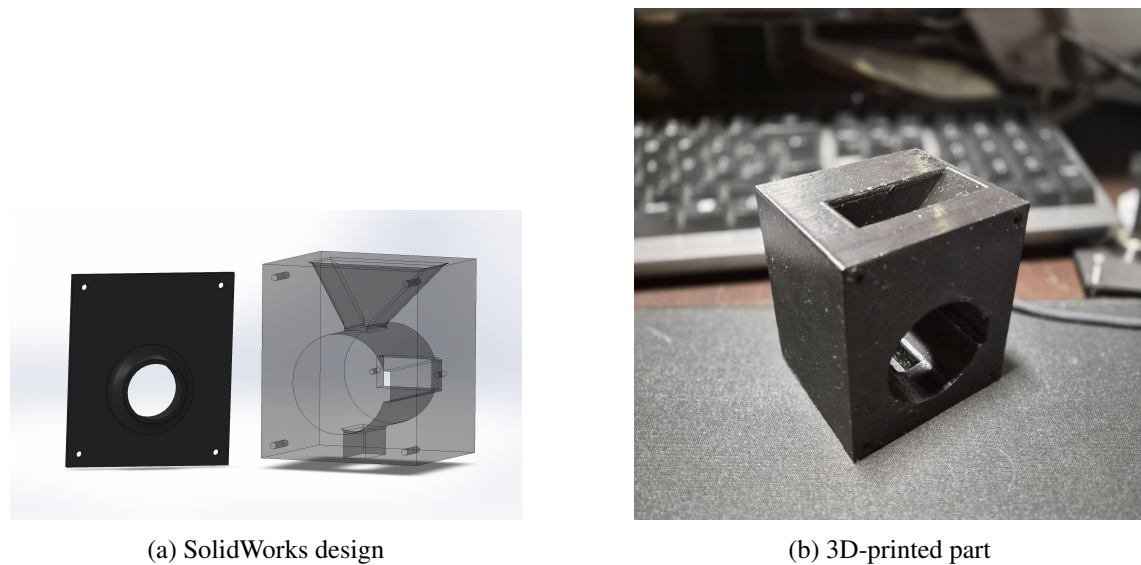


Figure 3.11: The body of the dispensing mechanism, with the asymmetric funnel embedded in its top face: (a) SolidWorks design and (b) the corresponding 3D-printed part.

Support structure: main tunnel and secondary collector funnel. The three compartment bodies described above sit on top of a shared support structure built around a main tunnel that runs directly from Compartment 1 down to the collection platform (Figure 3.12). Rather than giving each compartment its own independent load cell, the design routes all three compartments to a single sensing point: Compartment 1 is aligned directly over the main tunnel, so a unit dispensed from it falls straight down the tunnel to the platform below, while Compartments 2 and 3 dispense laterally into a secondary collector funnel that merges their two outputs and feeds them into the main tunnel through a lateral opening in its wall. The secondary collector funnel was deliberately designed with a curved lower section, rather than a straight one, so that a unit dispensed from Compartment 2 or 3 gains additional velocity as it slides down toward this merge point, rather than losing momentum through a sharper directional change. This curved section joins the main tunnel close to the tunnel's own lower opening, immediately above the collection platform, so that regardless of which of the two paths a unit takes to reach that point, it has only a short remaining distance to fall onto the same platform. From that point on, a unit originating from any of the three compartments completes its fall through the same main tunnel and lands on the same collection platform, allowing one load cell to confirm the drop and collection of medication regardless of which compartment dispensed it. The technical drawings for the main tunnel and for the secondary collector funnel are provided, respectively, in Appendix C.6 and Appendix C.7.

Each compartment also rests on its own upper support piece, connecting it to the funnels described above. Since this piece is identical across all three compartments, only a single technical drawing is provided, in Appendix C.5.

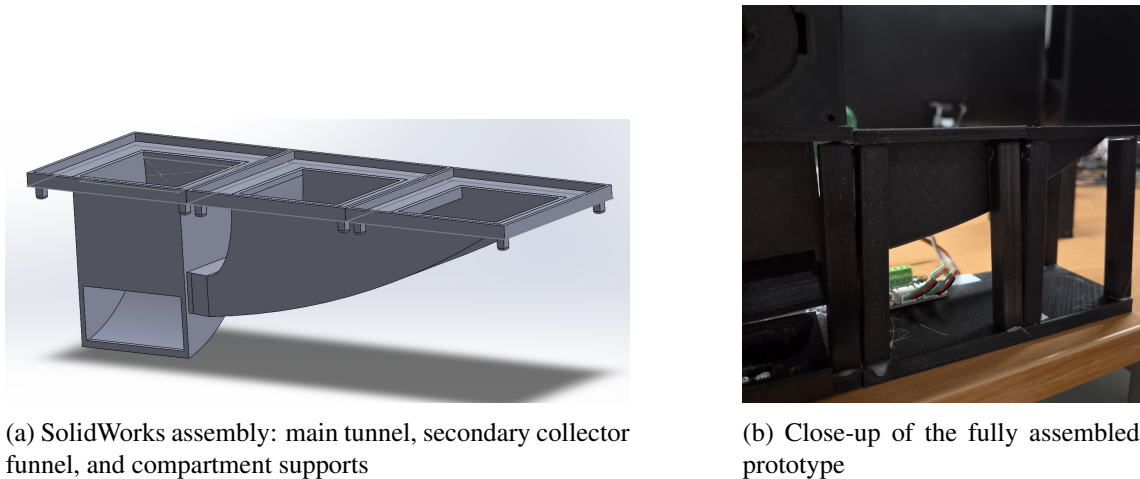


Figure 3.12: The main tunnel and secondary collector funnel: (a) SolidWorks assembly showing only the main tunnel, the secondary collector funnel, and the upper compartment supports, without the lower supports or pillars described below; (b) a close-up photograph of the fully assembled prototype, showing the same funnels in their final position.

Lower supports and the load cell mounting point. The compartment supports and funnels described above do not rest directly on the ground; they are held up by a set of pillars, printed separately from the rest of the support structure to make the printing process more efficient, and terminating in one of two types of lower support (Figure 3.13). Below Compartment 1, the lower support incorporates a groove for the load cell, since this is the point at which the main tunnel reaches the collection platform; below Compartments 2 and 3, the lower support is purely structural, as these two compartments do not require their own sensing point and only need to hold the funnel and compartment supports at the correct height.

The lower support with the load cell groove is where the final mounting strategy for the sensor is actually implemented, and is deliberately treated here as its own topic, separate from the funnel geometry above: the load cell is in contact with this fixed structure only at the end opposite to the one screwed to the platform onto which medication units fall. One end of the load cell is screwed into the groove of this lower support, a fixed point, while the other end is screwed to the platform itself (Section 3.3.5), leaving the sensor free to flex along its length as intended, rather than being constrained anywhere else along its body. The technical drawing for this piece is provided in Appendix C.8.

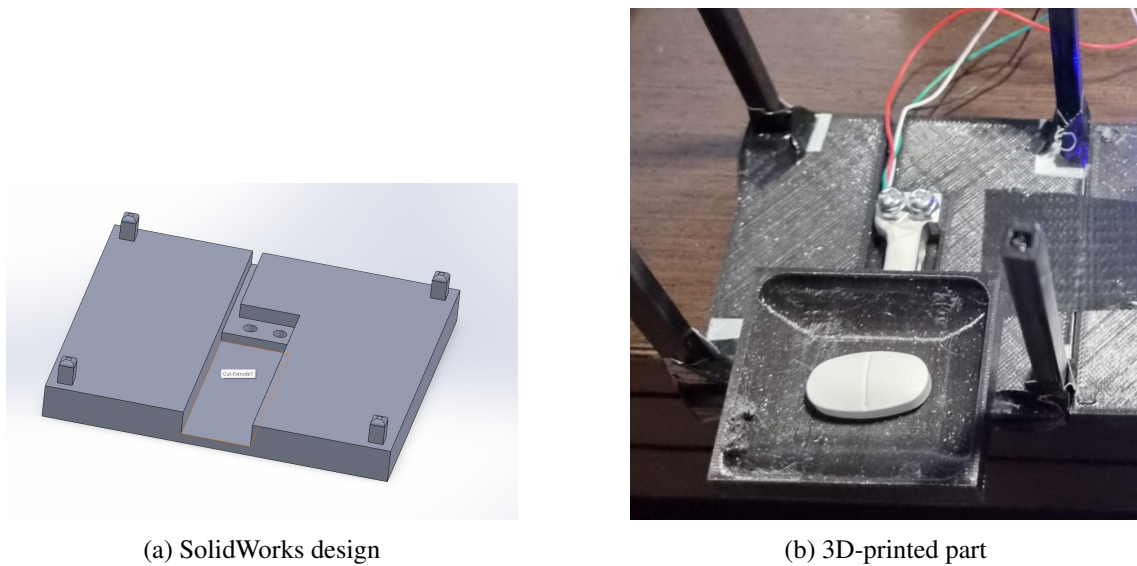


Figure 3.13: The lower support piece below Compartment 1, showing the groove into which one end of the load cell is screwed, the only point of contact between the load cell and this fixed part of the structure: (a) SolidWorks design and (b) the corresponding 3D-printed part.

Load cell platform. The platform supporting the load cell (Figure 3.14) is itself a separate printed part, designed and iterated independently of the support structure it sits within. Two earlier platform iterations were built and tested before the final geometry was reached; their results were compromised by an inadequate load cell fixation and are summarised only briefly, without detailed figures, in Section 4.2.1 of the testing chapter. For the final, three-compartment configuration, the platform was enlarged relative to these earlier iterations, to accommodate medication arriving from three possible drop points via the tunnel and collector funnel described above and to make manual collection easier for the caregiver or user.

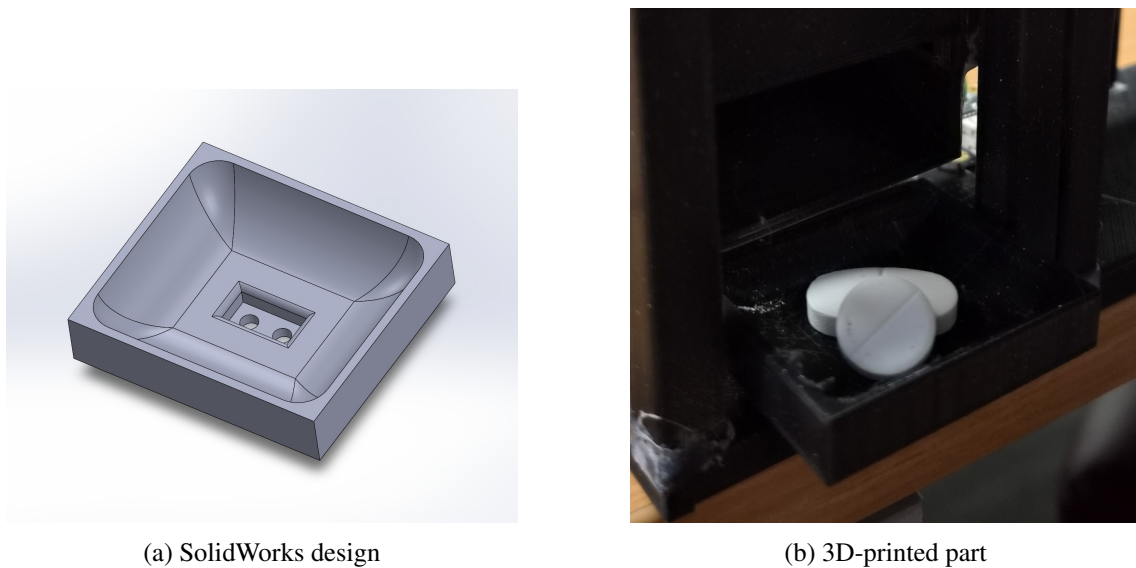


Figure 3.14: The final load cell platform, onto which medication units fall regardless of which compartment dispensed them: (a) SolidWorks design and (b) the corresponding 3D-printed part.

More significantly, the load cell mounting itself was redesigned in this final configuration to directly address the fixation-related mechanical sensitivity identified during the earlier platform iterations (Section 4.2.1), where a provisional adhesive-tape fixation was found to be the dominant source of measurement instability, even after reinforcement. In this new approach, described above for the lower support piece that holds it (Section 3.3.5), the load cell is constrained only at its two screwed ends: one end fixed to the lower support structure, the other end fixed to the platform onto which medication falls, with the remainder of its length left suspended (unsupported) along the rest of its span. This mounting strategy follows standard practice for strain-gauge load cells, which are designed to flex freely along their length in response to an applied load and to be constrained only at their designated mounting points; any additional, uncontrolled contact along the sensor's body, such as the groove-based, adhesive-reinforced fixation used previously, introduces parasitic stiffness and unpredictable mechanical coupling with the surrounding structure, consistent with the instability diagnosed during the earlier iterations (Section 4.2.1). The validation of this final mounting design is reported in Section 4.5. The technical drawing for the final platform geometry is provided in Appendix C.9.

3.3.6 3D Printing Process and Setbacks

The final design was manufactured using FDM 3D printing in PLA. In the first print of the original, single-compartment body, the hole intended to receive the servo was not correctly centred relative to the cylindrical cavity housing the cylinder. As a result, when the servo was mounted, its rotation axis was offset from the cylinder's axis, causing the cylinder to bind and preventing smooth rotation. The body was redesigned with the rear hole repositioned to be concentric with the cylinder cavity, and reprinted.

From the outset, the cylinder's slot for this first, single-compartment prototype was dimensioned for a 3D-printed pill substitute rather than a specific commercial tablet, for the reasons detailed in Section 3.3.7.

3.3.7 3D-Printed Pills as a Test Proxy

For the initial, single-compartment validation phase, it was decided from the start to use a 3D-printed pill substitute, dimensioned to fit the cylinder slot printed for this stage of the prototype (Figure 3.15), rather than a specific commercial tablet. The printed substitute measures 17 mm in length, 7 mm in width, and 4.5 mm in thickness.



(a) Real paracetamol tablet (Pharmakern) in the cylinder slot



(b) 3D-printed pill substitute in the cylinder slot

Figure 3.15: Comparison, for reference, between a real paracetamol tablet (Pharmakern) and the 3D-printed pill substitute used for the initial single-compartment tests, shown in the same cylinder slot: (a) the real tablet, and (b) the printed substitute.

This decision was made for two reasons. First, it isolates the mechanical variables under study, the behaviour of the asymmetric funnel, the rotation of the cylinder, and the weight reading by the load cell, from the dimensional variability of real tablets, which differ widely in size and shape across medications. Using a controlled, known geometry for the test units allowed these mechanical behaviours to be evaluated independently of tablet-specific characteristics, at a stage of the project where time was limited, while still allowing the full mechanical pipeline, from reservoir to funnel to cylinder to collection platform, to be exercised end to end. Second, it avoided the need to redesign and reprint the cylinder for a specific commercial tablet's dimensions before the singulation principle itself had been validated.

This original compartment, together with its printed pill substitute, was retained unchanged in the final, three-compartment configuration (Section 3.3.9), continuing to serve as the mechanically-validated reference compartment, while the two new compartments were dimensioned directly for

real medication (Ben-u-ron 500 mg and 1000 mg), removing the need for a printed proxy in those two positions.

3.3.8 Surface Treatment: Rationale for Omitting Resin Coating

The 3D-printed parts described in this section were tested in their raw FDM-printed state, without any epoxy resin coating. This decision was made for reasons of variable isolation and geometric precision during the rapid-prototyping phase: applying a resin coating would alter the dimensions and surface friction of the printed parts in ways that are difficult to predict and control, making it harder to attribute the observed mechanical behaviour specifically to the calibrated clearances of the FDM-printed geometry.

This is, however, a deliberate simplification rather than a final design decision. Before any such device could be introduced into a clinical or domestic environment in contact with medication intended for ingestion, the internal surfaces would need to be treated with a biocompatible coating compliant with the relevant food-contact material regulations, namely FDA 21 CFR 175.300 (indirect food additives: resinous and polymeric coatings) and EU Regulation 10/2011 (plastic materials and articles intended to come into contact with food). This requirement is recorded as a priority item for future work in Section 5.1.

3.3.9 Three-Compartment Implementation

Building on the fixed-versus-variable parameter analysis of Section 3.3.4 and following the suggestion of the dissertation supervisor, the single-compartment prototype was extended into a final, three-compartment assembly. The original compartment, validated mechanically with the 3D-printed pill substitute (Section 3.3.7), was kept unchanged, and two new compartments were added, each dimensioned for a specific real medication:

- **Compartment 2, Ben-u-ron 500 mg:** a new rotating cylinder with a slot sized for the circular tablet geometry of this medication, following the same operating principle described in Section 3.3.2.3;
- **Compartment 3, Ben-u-ron 1000 mg:** a new rotating cylinder with a slot sized for the larger tablet geometry of this dosage.

Each new cylinder reused, without modification, the fixed parameters identified in Table 3.2 (funnel wall angle, cylinder-to-cavity clearance, servo coupling and mounting), while the cavity dimensions were recalculated individually for each tablet, following the sizing criteria established for the original compartment; the secondary collector funnel's base radius, by contrast, was kept common to all compartments, as described in Section 3.3.4.

Table 3.3 reports, for each of the two real medications, the tablet dimensions measured with a caliper, and the corresponding cavity dimensions chosen for that compartment's cylinder, following the sizing criterion introduced in Section 3.3.4: each cavity dimension kept close to, but

slightly larger than, the tablet's own dimension, and cavity width kept below twice the tablet width, to geometrically prevent two units entering the slot side by side.

Table 3.3: Tablet dimensions (caliper-measured) and corresponding cylinder cavity dimensions chosen for each compartment.

Medication	Tablet length	Tablet width	Tablet thickness	Cavity (depth × width × length)
Ben-u-ron 500 mg	13.5 mm	13.5 mm	3.8 mm	14.5 × 14.5 × 4.2 mm
Ben-u-ron 1000 mg	21 mm	11 mm	7 mm	23 × 13 × 8 mm

Each compartment retains its own individual, embedded asymmetric funnel (Section 3.3.5), and the three compartments are routed into a single sensing point via the main tunnel and secondary collector funnel described in Section 3.3.5: Compartment 1 drops directly into the main tunnel, while Compartments 2 and 3 dispense into the secondary funnel that merges into it. This architecture allows the three compartments to share a single load cell for drop and collection confirmation, avoiding the cost and complexity of instrumenting each compartment independently.

A point of attention specific to this shared drop path, assessed in the testing described in Section 4.6, is whether medication units dispensed from different compartments in close succession can interact within the main tunnel or the secondary collector funnel, a failure mode that did not exist in a design where each unit had an independent, unshared drop path. This is examined alongside the multi-pill jamming behaviour of each individual compartment in the same section.

3.4 Electronics and Firmware

This section describes the electronic components selected for the MedBox prototype, the circuit connecting them to the ESP32-C6 microcontroller, the firmware that controls the dispensing mechanism described in Section 3.3, and its integration with the Firebase Realtime Database introduced in Section 3.2. The section also documents the main challenges encountered during firmware development and the solutions adopted, as well as the components that were considered during the design phase but were ultimately not implemented in the current prototype.

3.4.1 Selection and Justification of Components

This subsection presents the six hardware components selected for the MedBox prototype and the reasoning behind each choice: the ESP32-C6 microcontroller that coordinates the system, the SG90 servo motors that actuate the dispensing mechanisms described in Section 3.3.2.3, the HX711 load cell amplifier used for drop and collection confirmation, the DHT11 environmental sensor, the DS3231 real-time clock, and the passive buzzer used for acoustic alerting. Where a

viable alternative existed, it is discussed alongside the component actually selected, together with the reasoning that favoured one over the other (Figure 3.16).

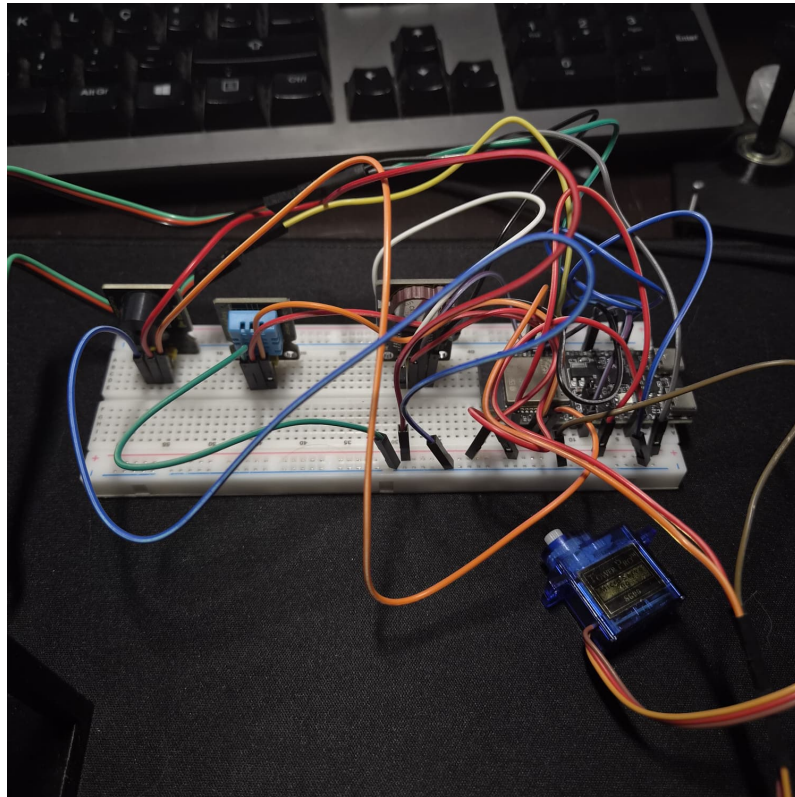


Figure 3.16: Breadboard circuit connecting the ESP32-C6 to the HX711 load cell amplifier, DHT11 sensor, DS3231 RTC, buzzer, and servo motors.

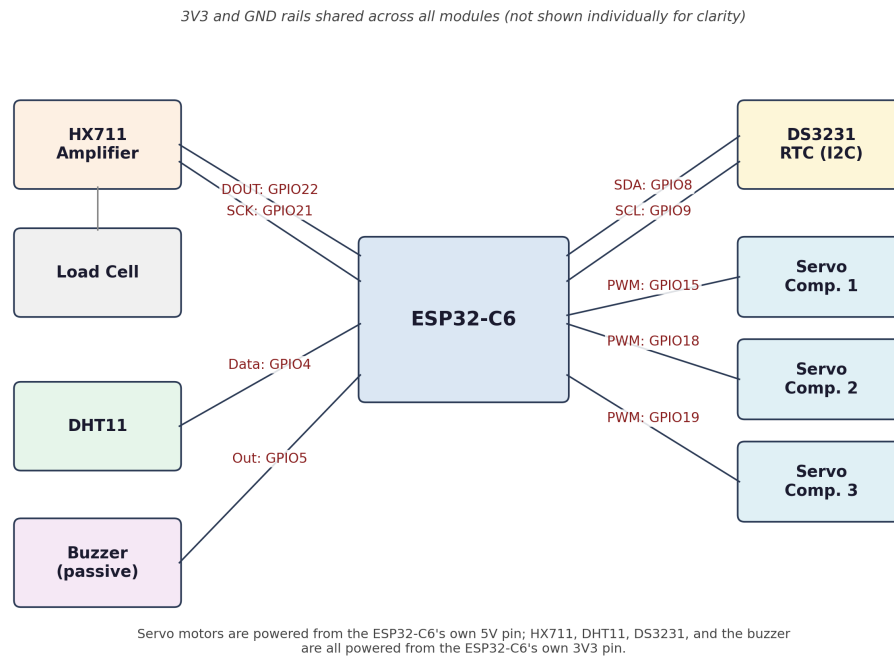


Figure 3.17: Wiring diagram of the ESP32-C6 and the peripherals listed in Table 3.4, complementing the breadboard photograph of Figure 3.16 with an explicit view of the GPIO connections.

3.4.2 Firmware-Level Component Description

This subsection describes each electronic component from the perspective of how the firmware drives and reads it, complementing the component selection rationale given in Section 3.4.1.

3.4.2.1 Microcontroller: ESP32-C6

The ESP32-C6 was selected as the central control unit on the basis of its native Wi-Fi connectivity and its sufficient number of GPIO pins. Table 3.4 summarises the pin assignment used in the final, three-compartment prototype.

Table 3.4: Pin assignment for the ESP32-C6 in the MedBox prototype.

Component	Signal	GPIO
DHT11	Data	4
HX711	SCK	21
HX711	DOUT	22
Buzzer	Output	5
Servo (compartment 1, original/test)	PWM	15
Servo (compartment 2, Ben-u-ron 500 mg)	PWM	18
Servo (compartment 3, Ben-u-ron 1000 mg)	PWM	19
DS3231 RTC	SDA / SCL (I2C)	8 / 9
Hatch (<i>alçapão</i>) servo, not active	PWM	14

3.4.2.2 Actuator: SG90 Servo Motors

Each dispensing mechanism described in Section 3.3.2.3 is actuated by its own SG90 micro servo, a cost-effective, hobby-grade servo capable of rotating approximately 180° under PWM control. The SG90 was chosen for three reasons: its torque is expected to be sufficient to rotate the 3D-printed cylinder against the friction of the printed bearings; its physical dimensions, including the distance from the servo body to the horn, could be accommodated within the wall thickness of the printed body, as discussed in Section 3.3.5; and it is widely available and inexpensive, which is appropriate for a domestic-scale proof-of-concept prototype.

The firmware declares three servo channels, one per compartment of the final assembly (Table 3.4), corresponding to the original compartment and the two new compartments described in Section 3.3.9. In a production device, the number of channels would need to scale further to match the target number of medication compartments, estimated at approximately 10 for typical polypharmacy regimens (Section 3.1), requiring a different hardware configuration.

During initialisation, each active servo channel is attached to its GPIO pin first (`attach()`), and only then commanded to its zero-degree resting position (`write(0)`), followed by a 600 ms settling delay. This ordering is required because the PWM signal is only active on the pin after `attach()` is called; issuing `write()` before `attach()` has no electrical effect and leaves the servo free to move unpredictably when the signal becomes active. This initialisation sequence is applied identically across all three active servo channels.

3.4.2.3 Drop Confirmation: HX711 Load Cell Amplifier

A single load cell, read through an HX711 24-bit analogue-to-digital amplifier, is shared by all three compartments to confirm that a medication unit has physically dropped onto the collection platform after any of the three cylinders rotates (FR03), via the merged collector funnel described in Section 3.3.5. The HX711 was selected because it provides sufficient resolution to detect the small weight of a single tablet or pill substitute, while requiring only two digital pins (DOUT and

SCK) for communication with the ESP32-C6. The calibration of this sensor, including the re-designed rigid mounting described in Section 3.3.5 and the determination of the detection threshold used in the firmware, is described in detail in Section 4.5.

Drop detection is performed by comparing the current platform weight against a baseline reading taken immediately before each dispensing rotation, and declaring a successful drop when the incremental weight delta exceeds a configurable threshold. Because the three compartments share one load cell, the firmware associates the resulting drop confirmation with whichever compartment initiated the dispensing rotation for that cycle.

When two or more medications are dispensed in the same grouped event (Section 3.4.6), the load cell is tared only once, at the very start of that event, rather than being re-tared before each individual medication. Instead, a fresh baseline weight is measured immediately before every single unit's own rotation, regardless of which medication it belongs to. Because this baseline is actively re-measured rather than assumed to be zero, it correctly accounts for any weight already resting on the platform from an earlier medication in the same group that has not yet been collected, so the delta attributed to each subsequent unit still isolates just that unit's own weight. A direct consequence of this design is that collection is confirmed only once, at the end of the whole group: the user is expected to retrieve every medication dispensed in that time slot together, rather than one at a time between doses.

Table 3.5 summarises the constants that govern this detection logic, to serve as a single point of reference for the parameters discussed throughout this section and exercised in the testing chapter.

Table 3.5: Firmware constants governing drop and collection detection.

Constant	Value	Role
PILL_THRESHOLD_G	0.3 g	Minimum weight delta, above the pre-rotation baseline, required to register a candidate drop reading
LEITURAS_QUEDA_ESTAVEL	5 readings	Number of consecutive readings that must stay above PILL_THRESHOLD_G before a drop is confirmed; a single reading that later falls back below threshold resets this counter to zero
TIMEOUT_QUEDA_MS	6000 ms	Maximum time allowed for a drop to be confirmed before the unit is treated as not dispensed
INTERVALO_POLLING_MS	300 ms	Sampling interval used while polling for both drop and collection
COLLECTION_THRESHOLD_G	0.15 g	Maximum absolute weight allowed for the platform to be considered empty
LEITURAS_ESTABILIZACAO	10 readings	Number of consecutive readings that must stay within COLLECTION_THRESHOLD_G of zero before collection is confirmed; equivalent to 3 s of continuous stability at the sampling interval above
LEITURAS_BASE_ESTAVEL	4 readings	Number of readings averaged to obtain the pre-rotation baseline weight, with their spread reported for diagnostic purposes
DELAY_ASSENTAMENTO_TARE_MS	1500 ms	Settling delay after tare(), before the first baseline reading of a dispensing event
DELAY_ASSENTAMENTO_MED_MS	2000 ms	Settling delay between different medications within the same grouped dispensing event
DELAY_ASSENTAMENTO_UNIDADE_MS	2000 ms	Settling delay between consecutive units of the same medication, when the prescribed quantity exceeds one

The distinction between PILL_THRESHOLD_G and LEITURAS_QUEDA_ESTAVEL is the key safeguard discussed in Section 4.5.3: a single reading above threshold is not, on its own, sufficient to confirm a drop, since a momentary touch or vibration can produce a comparable spike. Only a weight that remains above threshold for five consecutive readings, corresponding to roughly 1.5 s of sustained weight, is accepted as a genuine drop.

Figure 3.18 summarises the full sequence of steps executed for each unit dispensed, and how it repeats across units and medications within a single grouped dispensing event, before collection

is finally assessed once for the whole group. The annotated example values are taken from a recorded session dispensing three medications scheduled at the same time, discussed further in Section 4.7.2.

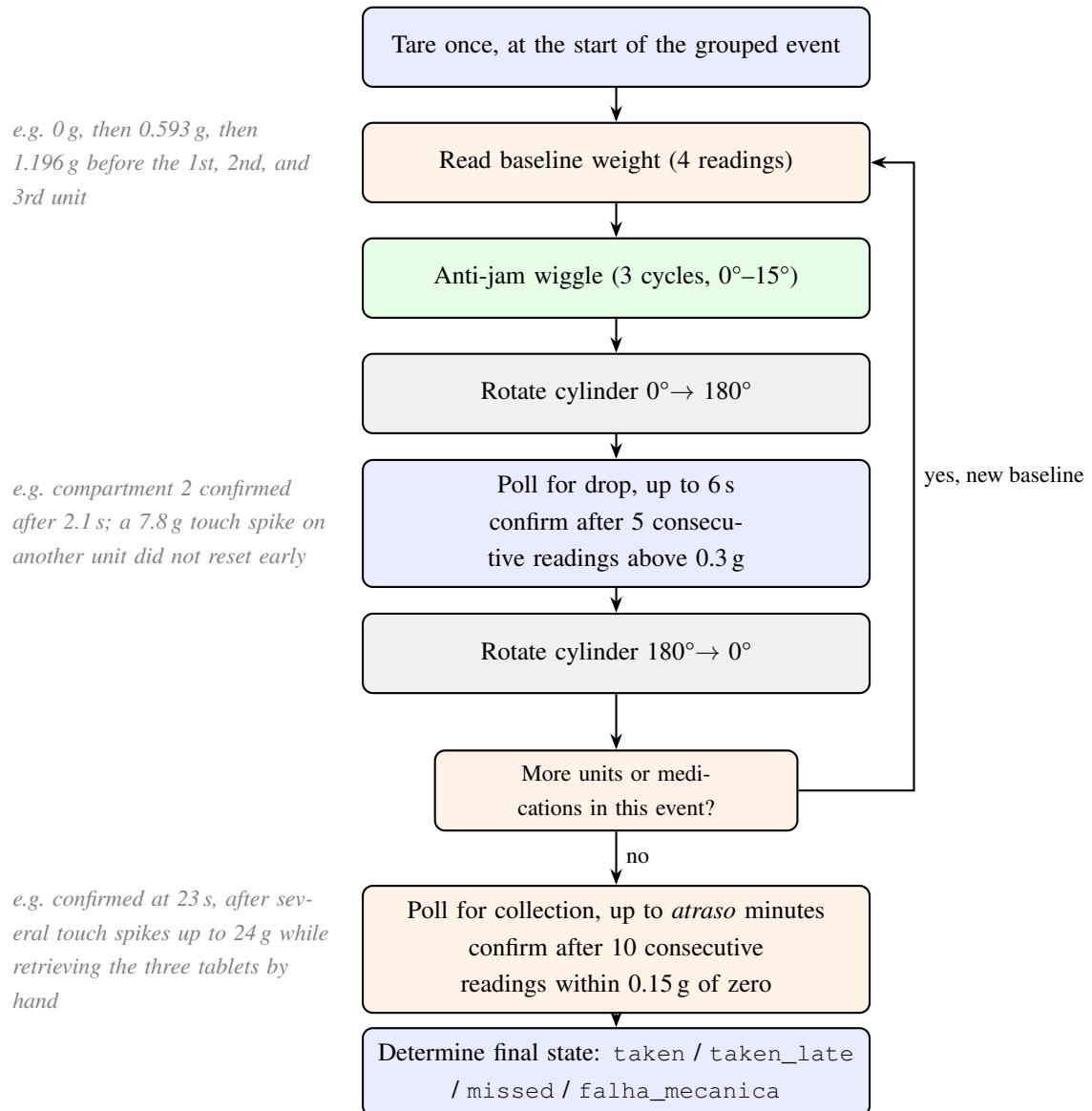


Figure 3.18: Sequence of firmware steps executed during a grouped dispensing event, from the initial tare to the final collection outcome. The decision point loops back to a fresh baseline reading for each additional unit or medication in the same event; collection is assessed only once, after every unit has been dispensed. Annotated examples are drawn from the three-medication session discussed in Section 4.7.2.

3.4.2.4 Environmental Monitoring: DHT11 Temperature and Humidity Sensor

A DHT11 sensor provides the temperature and humidity readings used for the environmental monitoring functionality (FR06), motivated by the storage condition concerns raised in the interviews

with pharmacists (Section 3.1). Table 3.6 summarises the key specifications of the DHT11 against its direct successor, the DHT22, which was considered as an alternative.

Table 3.6: Specification comparison between the DHT11 and DHT22 sensors [74, 75].

Parameter	DHT11	DHT22
Temperature range	0 to 50 °C	-40 to 80 °C
Temperature accuracy	±2 °C	±0.5 °C
Humidity range	5 to 95 %RH	0 to 100 %RH
Humidity accuracy	±5 %RH	±2 %RH
Resolution	1 °C / 1 %RH	0.1 °C / 0.1 %RH
Sampling rate	1 Hz	0.5 Hz

The DHT11 was chosen for the prototype on the grounds of cost and availability, and its accuracy is sufficient for demonstrating the alert functionality in a proof-of-concept context, where the objective is to validate the end-to-end pipeline (sensor reading, threshold comparison, RTDB write, push notification) rather than to monitor storage conditions with clinical precision. However, the DHT11's accuracy of ±2 °C and ±5 %RH would not meet the requirements of a production medical device, where tighter tolerances are necessary to reliably detect and flag conditions that could compromise medication stability. In that context, the DHT22, with its ±0.5 °C temperature accuracy and ±2 %RH humidity accuracy, would be a significantly more appropriate choice, and its integration requires no firmware changes beyond updating the compile-time flag (`TIPO_DHT`), which the firmware already supports.

3.4.2.5 Time Reference: DS3231 Real-Time Clock

A DS3231 real-time clock module, connected via I2C, provides a fallback time source for the firmware when an NTP (network time protocol) connection is unavailable, supporting the offline resilience requirement (NFR01). The DS3231 was chosen over relying solely on the ESP32-C6's internal clock because it includes a temperature-compensated crystal oscillator and a battery backup, maintaining accurate time across power cycles and periods without Internet connectivity, both of which are essential for a device whose core function, dispensing medication at scheduled times, depends on accurate timekeeping. As described in Section 3.4.4, the RTC also assumes full timekeeping responsibility when the device boots in offline mode following a failed Wi-Fi connection attempt.

3.4.2.6 Alert Actuator: Passive Buzzer

A passive buzzer provides the acoustic alarm escalation described in the scientific alert policy documentation referenced in Section 3.1 (the 15-minute escalation threshold suggested by the interviewed pharmacists). A passive buzzer was chosen over an active buzzer because it allows the firmware to control the tone and the number of beeps directly through GPIO toggling, enabling the

progressive escalation (1, 2, or 3 beeps depending on elapsed time, as described in Section 3.4.4) without requiring additional driving circuitry. This alarm escalation applies uniformly regardless of which of the three compartments the overdue dose originated from.

3.4.3 Firmware Development Phases

The firmware was developed incrementally, in four phases, each building on the previous one and validated independently before integration (Figure 3.19).

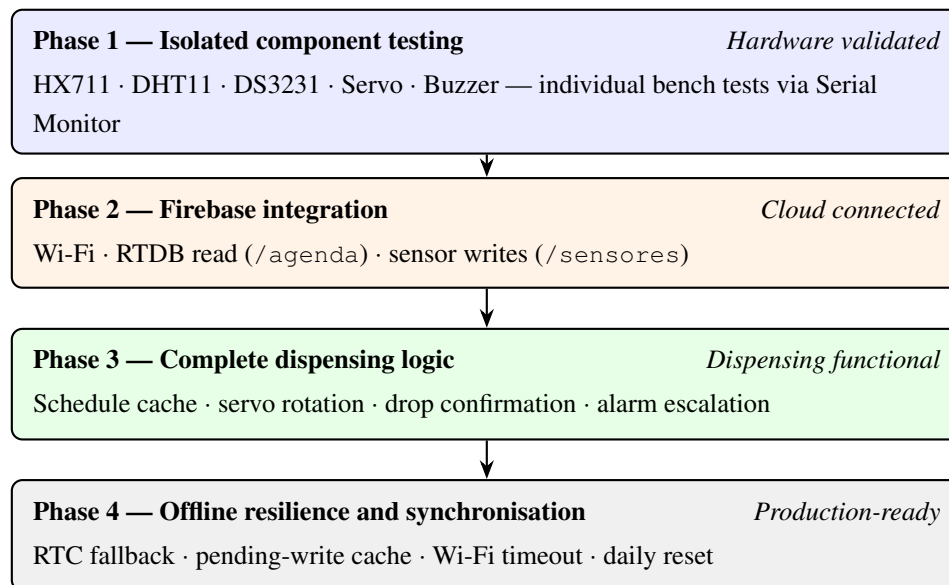


Figure 3.19: Incremental firmware development phases, each validated independently before integration of the next.

Phase 1: Isolated component testing. Each sensor and actuator (HX711, DHT11, DS3231, servo, buzzer) was individually connected to the ESP32-C6 and tested in isolation, with readings and outputs verified through the Arduino IDE Serial Monitor, without any connection to the mobile app or to Firebase. This phase, conducted during the exploratory mechanical approaches described in Section 3.3.2.1, established that each component functioned correctly and established the baseline values (e.g. raw HX711 readings, DHT11 ranges) needed for later calibration.

Phase 2: Firebase integration. The firmware was extended to connect to Wi-Fi and to the Firebase Realtime Database, initially focused on reading the /agenda node and writing sensor readings to /sensores, establishing the basic cloud connectivity later relied upon by the mobile app.

Phase 3: Complete dispensing logic. The full dispensing routine (UC6, Table A.1) was implemented, including the local schedule cache, the servo rotation sequence, load-cell-based drop confirmation, the collection-waiting period, and the progressive acoustic alarm. This logic was implemented generically over the compartment index, allowing it to be reused unchanged when the second and third compartments were added (Section 3.3.9).

Phase 4: Offline resilience and synchronisation. The local cache and the RTC fallback were added to support operation without an Internet connection (NFR01), together with the logic to synchronise pending writes to the RTDB once connectivity is restored, and the daily reset of all doses to `pending` at midnight. In this phase, the Wi-Fi connection attempt was also bounded by a 10-second timeout, after which the device proceeds in offline mode rather than blocking indefinitely.

3.4.4 Firmware Architecture

The firmware uses a timer-based polling architecture within the Arduino `loop()` function. Four independent software timers, represented in Figure 3.20, each implemented using the `millis()` subtraction idiom, govern the main periodic behaviours:

- **Dispensing check (every 5 seconds):** the firmware compares the current time, obtained from NTP or, in its absence, from the DS3231 RTC, against the local schedule cache, and triggers the dispensing routine (UC6) for the relevant compartment when a match is found;
- **Sensor publication (every 4 seconds):** the firmware reads the DHT11 and publishes the temperature and humidity values to `/sensores/dht11_interno`, which also serves as the heartbeat signal monitored by the Node.js server's watchdog (Section 3.5.4);
- **Manual command polling (every 2 seconds):** the firmware checks `/comando_manual/executar` to detect anticipated dispensing requests from the caregiver (UC4, Table A.2);
- **Daily reset check (every hour):** the firmware compares the current date against the last recorded reset date in the RTDB, and triggers the daily dose reset if a new day has begun.

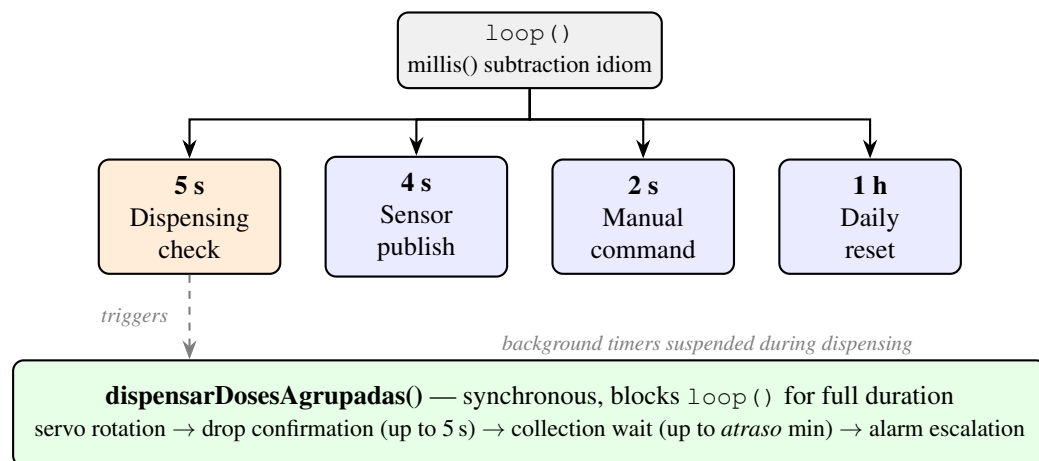


Figure 3.20: Firmware architecture: the Arduino `loop()` function drives four independent software timers. When the dispensing check (5 s timer, highlighted) detects a scheduled dose, it triggers `dispensarDosesAgrupadas()`, which executes synchronously and suspends the remaining background timers for its full duration.

It should be noted that this timer-based polling governs only the background periodic tasks. The dispensing routine itself (`dispensarDosesAgrupadas()`) executes synchronously and occupies the `loop()` thread for the full duration of the dispensing sequence, including the physical rotation of the relevant cylinder, the drop-confirmation polling window (up to 5 seconds per unit), and the collection-waiting period (up to `atraso` minutes, configurable per dose). During this time, the background timers are suspended. For the current three-compartment prototype, this is an accepted design trade-off: the device is actively engaged in its primary function during the dispensing window and no concurrent background task is time-critical in that context. A conversion to a fully non-blocking, state-machine-based architecture is identified as a priority item for future work (Section 5.1).

Beyond this timer-based polling architecture, the firmware implements six further mechanisms addressing specific engineering requirements identified during development. Figure 3.21 summarises them, and each is described individually below.

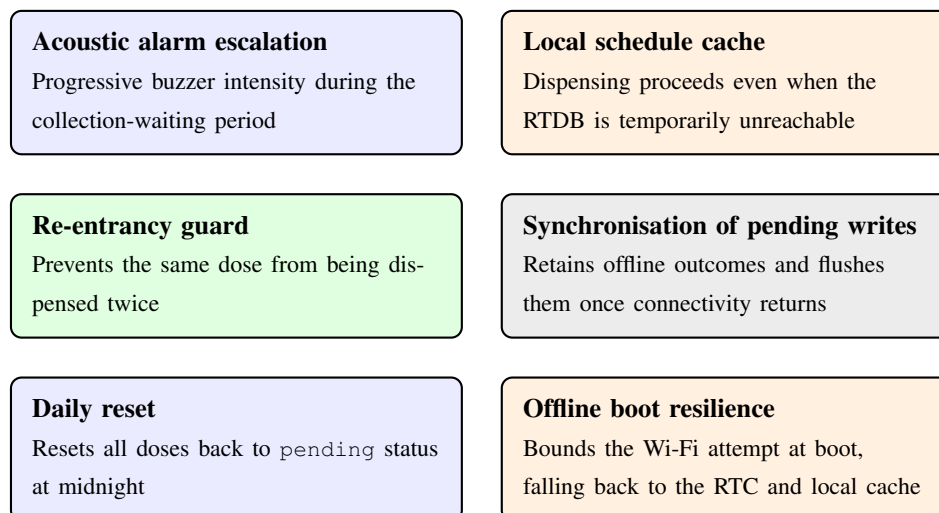


Figure 3.21: Additional firmware mechanisms, beyond the timer-based polling architecture, described individually below.

Acoustic alarm escalation. During the collection-waiting period that follows a confirmed drop, the firmware implements a progressive alarm policy using two independent software timers. Within each five-minute alarm cycle, the buzzer is active for the first 30 seconds and silent for the remaining 270 seconds. The intensity of the alarm increases with elapsed time: during the first five minutes, the buzzer emits one beep every 5 seconds; between five and ten minutes, two beeps every 3 seconds; and beyond ten minutes, three beeps every 1.5 seconds. The first beep of each cycle is deliberately deferred by the full inter-beep interval from the moment the collection window opens, giving the user a brief grace period before the alarm activates.

Local schedule cache. The daily agenda is synchronised from `/agenda` into a local array of structures (`Dose cacheDoses[10]`), each holding the compartment number, scheduled time, medication name, quantity, tolerance window (`atraso`), dispensing status, and creation date and

time of a scheduled dose. The creation date and time fields are used to avoid incorrectly flagging as overdue any dose that was added to the schedule on the same day at a time later than its scheduled dispensing hour. This cache allows the dispensing check to proceed even when the RTDB is temporarily unreachable, directly supporting NFR01, and now spans doses across all three active compartments.

Re-entrancy guard. To prevent the same dose from being dispensed twice, for example if the firmware briefly loses and regains connectivity within the same minute, the variable `ultimaHoraDispensada` records the HH:MM string at which a dispensing event was last triggered. Before dispensing, the firmware checks both this variable and the cached `status` field of the dose (only dispensing if `status == "pending"`), providing two independent layers of protection against duplicate dispensing (NFR05). The guard is reset on power-on; robustness against reboot-induced duplicates relies on the `status == "pending"` check, which is persisted in the RTDB.

Synchronisation of pending writes. When the firmware cannot reach the RTDB at the moment a dispensing event completes, the outcome is retained in a local pending-log buffer and written to `/history` and `/agenda` once connectivity is restored. If the device was offline at the scheduled dispensing time and the dose was therefore not dispensed, the outcome is recorded as `device_offline` and an alert flag is set in `/alertas/falha_dispositivo`.

Daily reset. Once per hour, the firmware compares the current date against `/sistema/ultima_data_reset`. If a new day is detected, all entries in `/agenda` for the current weekday are reset to `status = "pending"` and `dispensada = false`, and the alert flag `/alertas/falha_dispositivo` is cleared, preparing the schedule for the next dispensing cycle.

Offline boot resilience. The Wi-Fi connection attempt at boot is bounded by a 10-second timeout (40 attempts at 250 ms intervals). If the access point is unreachable within this window, the device logs a diagnostic message and proceeds without network connectivity. NTP synchronisation and Firebase authentication are skipped entirely in this case; the DS3231 RTC provides the time reference and the local cache, if already populated from a prior connected session, provides the schedule. Pending writes accumulated during offline operation are flushed automatically once `Firebase.reconnectWiFi(true)` restores the connection.

3.4.5 Firebase Realtime Database Integration

The firmware authenticates with the Firebase Realtime Database using a dedicated device account (`idoso@dispensador.pt`), separate from the caregiver accounts managed through Firebase Authentication in the mobile app. Table 3.7 summarises the RTDB nodes read and written by the firmware, consistent with the end-to-end data flow described in Section 3.2 (Figure 3.4).

Table 3.7: Firebase Realtime Database nodes accessed by the firmware.

Node	Access	Purpose
/agenda	Read/Write	Daily medication schedule, synchronised to the local cache; status and dispensing flag updated after each event, across all three compartments
/config	Read	Temperature and humidity alert thresholds, set by the caregiver
/comando_manual	Read/Write	Anticipated dispensing commands (UC4)
/sensores	Write	Temperature and humidity readings, published every 4 seconds
/history	Write	Outcome of each dispensing event
/toma_atual	Write	State of the current/most recent dispensing event, including drop and collection confirmation timestamps
/alertas	Write	Clinical alert flags (temperatura_excedida, humidade_excedida, falha_dispositivo, dose_em_atraso)
/sistema	Read/Write	Date of last daily reset, used to detect mid-night transitions

Two fields in `/toma_atual` and `/alertas` serve related but distinct purposes. The field `/toma_atual/alerta_ativo` reflects whether the most recent dispensing event ended without confirmed collection, and is used by the mobile app to display a real-time alert banner. The field `/alertas/dose_em_atraso` serves the same semantic purpose but is written to a separate path that the Node.js notification server monitors to trigger a push notification through the Firebase Cloud Messaging pipeline (Section 3.5.4). Both fields are written simultaneously at the end of each dispensing event, ensuring that neither the in-app display nor the push notification channel is missed.

3.4.6 Challenges and Solutions in Firmware Development

Several practical challenges were encountered during firmware development, in addition to the architectural considerations already described in Section 3.4.4.

Servo initialisation sequence. During early testing, the servo was observed to produce a brief mechanical jolt at device startup. The root cause was identified as a sequencing error in the initialisation code: `write(0)` was called before `attach()`, which has no electrical effect because the PWM signal is not yet active on the GPIO pin. When `attach()` subsequently activated the

PWM output, the servo received its first pulse from an indeterminate hardware state and moved abruptly to an unexpected position before settling at zero degrees. The fix was to call `attach()` first, activating the PWM signal on the pin at its default pulse width, and then issue `write(0)` to command the zero-degree position, followed by a 600 ms settling delay. This fix was applied identically to all three active servo channels.

Servo angular calibration. Beyond the initialisation sequence, the angular range and speed of each servo's rotation had to be tuned empirically so that the corresponding cylinder's slot aligns correctly with its funnel opening in its resting position, and completes a full dispensing rotation without overshooting, which would misalign the slot on return. The firmware moves each servo in steps of 1° with a 5 ms inter-step delay, producing a full 180° sweep over approximately 900 ms, the speed validated in Section 4.5. This produces a smoother rotation and reduces mechanical stress on the printed cylinder and bearings, for all three compartments.

Multiple medications scheduled at the same time. When two or more medications are scheduled for the same time slot, in different compartments, the firmware groups these doses and dispenses them sequentially, with a 2-second settling delay between consecutive dispensing operations (`DELAY_ASSENTAMENTO_MED_MS`). This delay was introduced after observing that dispensing multiple doses back-to-back, without a pause, could result in the load cell not settling before the next reading, affecting drop confirmation for the second dose; with three compartments now sharing a single load cell (Section 3.4.2.3), this settling delay is particularly relevant, and its adequacy for the merged collector-funnel geometry is examined in Section 4.6. A separate 2-second inter-unit delay (`DELAY_ASSENTAMENTO_UNIDADE_MS`) is applied between consecutive units of the same medication when the prescribed quantity exceeds one.

Anticipated dispensing and same-day duplication. As described in detail in Table A.2, the resolution of the anticipated-dispensing duplication issue (FR08) was ultimately implemented on the mobile app side, by marking the corresponding `/agenda` entry as `taken` at the moment the anticipated dispensing is requested, rather than through firmware-side logic.

Dose-overdue alert synchronisation. At the end of each dispensing event, the firmware writes `/toma_atual/alerta_ativo` to indicate whether the dose was collected within its tolerance window. However, the Node.js server's clinical alert listener (Section 3.5.4) monitors `/alertas/dose_em_atraso`, a separate node in a different subtree. The firmware now writes both fields simultaneously, ensuring that an uncollected dose triggers a push notification through the existing server-side pipeline in addition to updating the in-app display.

Spurious alarm at collection window opening. In an earlier version of the firmware, the acoustic alarm was triggered immediately when the collection window opened, before the user had any opportunity to collect the medication. The cause was that the timer variable `tUltimoBipe` was initialised to zero, so the first comparison against the current time (`millis()`) always evaluated as exceeding the inter-beep interval, causing an immediate beep. This was corrected by initialising `tUltimoBipe` to `millis()` at the start of the collection phase, so the first beep only fires after the configured inter-beep interval has genuinely elapsed.

3.4.7 Components Considered but Not Implemented

Three components were designed into the system, at least at the software or mechanical level, but are not active in the current prototype. Each is discussed here for completeness, with the corresponding future-work item referenced in Section 5.1.

Magnetic tamper sensors (MC38). The mechanical design originally anticipated a magnetic reed switch (MC38) on each compartment's lid, to detect unauthorised opening of a medication compartment. These sensors were set aside for the current three-compartment prototype, since their operating principle was validated independently on the bench and is not considered technically difficult to integrate into a future multi-compartment device.

Visual alert LEDs. The scientific alert policy underlying the acoustic escalation also defines a five-stage visual escalation (green to red LEDs). These were not implemented in the current prototype because the prototype is powered via USB from a PC rather than from batteries, and the additional current draw of multiple LEDs was not considered justified for a bench setup. This is revisited in the context of the battery-powered design discussed in Section 5.1.

Hatch servo. A servo channel (GPIO 14) is reserved in the firmware for a fail-safe hatch mechanism, which would physically isolate the medication reservoir if a dispensed dose is not collected within its tolerance window, as discussed in Section 3.3.2.1. This channel is declared and configured in software but not actuated; the firmware logs a placeholder diagnostic message at the point where the hatch logic would execute, making the integration point explicit for future implementation. The mechanical design of this fail-safe component is identified as a priority item for future work.

3.5 Mobile Application and Backend

This section describes the development of the mobile application and the Node.js notification backend. It covers the structure of the two user-facing interfaces, the medication catalogue database and the pipeline used to populate it, the approach adopted for recurring dose management, and the architecture of the server-side notification service, including the rationale for choosing a server-side approach over a client-side one.

3.5.1 Mobile Application Structure

The mobile application was developed in Flutter and is centred on the caregiver profile, with a dedicated interface described below (Figures 3.22, 3.23 and 3.24). A second, elderly-facing interface was originally envisioned, as discussed in Section 3.2.2, but was not implemented or tested within the scope of this prototype, and is discussed instead as a priority item for future work (Section 5.1). The application uses Firebase Authentication for the caregiver profile and connects to both the Firebase Realtime Database (RTDB) and Firebase Cloud Firestore, depending on the type of data being accessed.

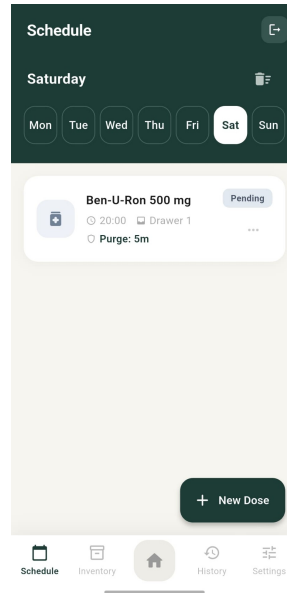
The caregiver interface is structured around five tabs, each corresponding to a functional area of the system:

- **Agenda** — the central tab for medication schedule management, allowing the caregiver to create, edit, and delete scheduled doses. Doses are organised by day of the week, with each day displaying the list of doses scheduled for that day and their current status. To create a new dose, the caregiver selects: (1) the medication, chosen by name from the Infomed catalogue integrated in the Inventory tab; (2) the compartment of the dispenser in which that medication is loaded (one of the three compartments described in Section 3.3.9); (3) the dispensing quantity (number of units per dose); (4) the scheduled dispensing time; (5) the tolerance window (*atraso*), i.e. the time available for collection before the fail-safe hatch mechanism would activate. This parameter was deliberately left configurable rather than fixed, based on the recommendation from the interviewed pharmacists that different medications have different handling requirements, a value that can be determined for each specific medication from its patient information leaflet; and (6) the frequency of the dose, which can be restricted to a single specific day or set to repeat on any combination of days of the week (to accommodate, for example, medications taken daily or on alternate days). Recurring doses share a common `groupId`, allowing the entire series to be edited or deleted in a single operation;
- **Inventory** — allows the caregiver to associate a medication from the Infomed catalogue with each physical compartment of the dispenser. The association is stored in a dedicated `/inventory` node of the RTDB and is used by the Agenda tab to pre-fill the medication field when the caregiver selects a compartment;
- **Monitor** — the operational overview tab, designed to give the caregiver an immediate snapshot of the system state. From top to bottom, the tab displays: (1) the dispenser's connectivity status (online or offline), derived from the most recent heartbeat timestamp in `/sensores/dht11_interno/ultima_leitura`; (2) the current temperature and humidity inside the dispenser, updated every 4 seconds by the firmware; (3) the next scheduled dose (medication, compartment, and time); (4) the most recent dispensing event and its outcome, read from `/toma_atual`, including whether the dose was collected and the collection time; and (5) any active alert flags from `/alertas`, displayed as prominent banners;
- **History** — presents a chronological log of all dispensing events, read from the `/history` node, with colour-coded status labels for each outcome (*taken*, *missed*, *failure*, *device_offline*). Each entry records both the originally scheduled dispensing time (`scheduled_time`) and the timestamp at which the dose was actually collected (`hora_recolha`), providing a complete record of any delays between the two;
- **Settings** — allows the caregiver to manually configure the three environmental alert thresholds: maximum temperature, minimum temperature, and maximum humidity. These values

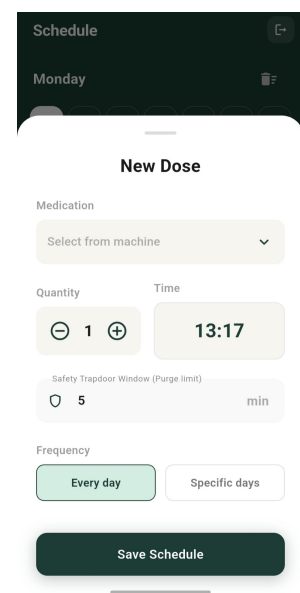
are written to the `/config` node of the RTDB and are read by the firmware at every 4-second sensor cycle ($\text{INTERVALO_SENSORES} = 4000\text{ms}$), meaning that any threshold change made in the app takes effect at the next sensor reading without requiring a device restart.



(a) Login screen



(b) Agenda tab



(c) Dose creation modal

Figure 3.22: Caregiver interface: (a) Firebase Authentication login screen; (b) Agenda tab showing scheduled doses by day with status indicators; (c) dose creation modal with fields for medication, compartment, quantity, time, tolerance window, and day-of-week frequency selection.

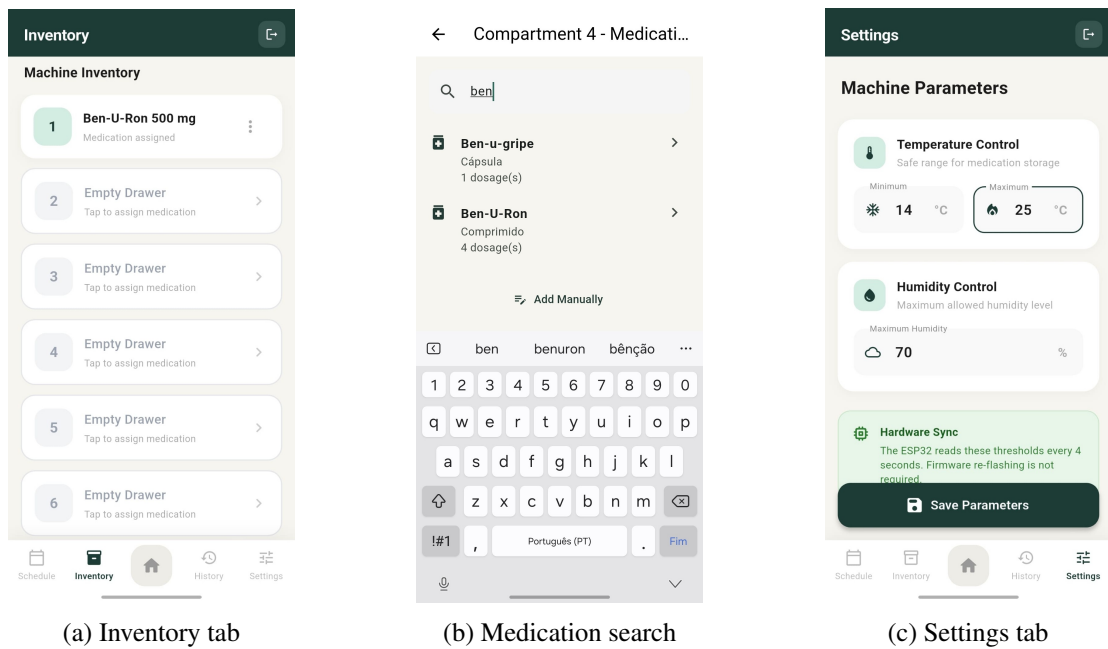


Figure 3.23: Inventory and settings: (a) compartment-to-medication association; (b) Infomed catalogue search using a `search_slug` prefix query over 6000+ entries; (c) manual configuration of environmental alert thresholds.

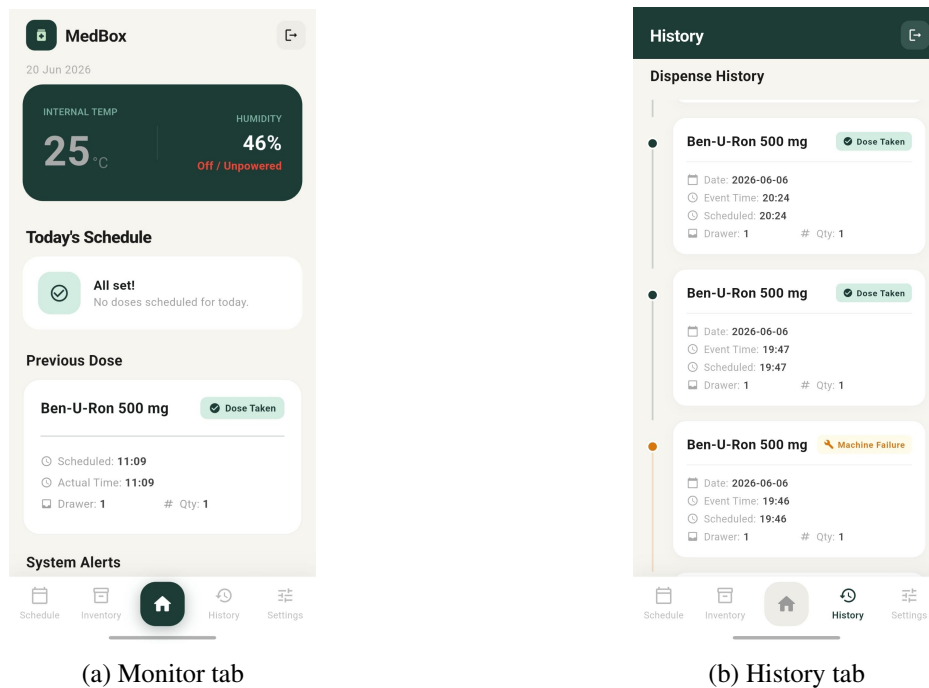


Figure 3.24: Monitoring and history: (a) Monitor tab showing connectivity status, live temperature and humidity readings, next scheduled dose, most recent dispensing outcome, and active alert banners — all updated via `onValue` RTDB streams; (b) History tab with chronological dispensing log, colour-coded outcome labels, and both the scheduled and actual collection times.

All data-driven views in the caregiver interface use Flutter's `StreamBuilder` widget, subscribed to `onValue` streams from the RTDB. This means that any change made to the dispenser state (e.g. a dose taken, a sensor reading updated, an alert raised) is reflected in the interface within the latency bound established by NFR02, without requiring the user to manually refresh the view.

3.5.2 Medication Catalogue Database

To provide the caregiver with a realistic medication search experience, the application integrates a catalogue of medications commercially available in Portugal. Rather than using fictitious data, the catalogue was derived from the publicly available *Listas de Preços Aprovadas* published monthly by the Autoridade Nacional do Medicamento e Produtos de Saúde (Infarmed) [76]. This document lists all medications with an active *Autorização de Introdução no Mercado* (AIM), including their commercial names, active substances, dosages, and pharmaceutical forms, and is accessible without requiring access to any private API.

3.5.2.1 Data Cleaning Pipeline

The raw dataset was processed through a data-cleaning pipeline before being loaded into the application's database. The pipeline consisted of three steps:

1. **Attribute filtering:** attributes irrelevant to the dispensing system (registry numbers, CN-PEM codes, packaging dimensions, pricing, and reimbursement data) were removed;
2. **Pharmaceutical form filtering:** only medications in tablet or capsule form were retained, since these are the only forms compatible with the dispensing mechanism;
3. **Deduplication:** the original dataset contains multiple entries for the same medication due to different commercial pack sizes. A deduplication step, based on the combination of medication name and dosage, was applied to produce a list of unique entries for the user interface.

After cleaning, the dataset comprised over 6000 validated entries. A Python automation script, executed in Google Colab using the `pandas` library for data manipulation and the `firebase-admin` library for authenticated cloud writes, migrated the entire catalogue to a Firebase Cloud Firestore collection named `medicamentos`, creating one document per unique medication entry. This approach allowed the 6000-entry catalogue to be loaded in a single automated pass, avoiding the risk of manual entry errors.

3.5.2.2 Efficient Text Search with `search_slug`

To support fast and efficient text search over a catalogue of this size from a mobile client, each Firestore document includes a `search_slug` field containing a normalised (lowercase, diacritics removed) concatenation of the medication name and active substance. Medication search queries in the app use a range query on this field:

```

FirebaseFirestore.instance.collection('medicamentos')
  .where('search_slug', isGreaterThanOrEqualTo: query)
  .where('search_slug', isLessThanOrEqualTo: '$query\uf8ff')

```

This approach avoids downloading the entire collection to the client and enables prefix-based search with low latency, directly supporting NFR07.

3.5.3 Recurring Dose Management and Database Indexing

When a caregiver creates a dose that repeats on multiple days of the week, the application creates one `/agenda` entry per day, all sharing the same `groupId` timestamp. This allows the caregiver to edit or delete the entire recurring series in a single operation, without having to identify and modify each daily entry individually.

The deletion of a recurring series requires a server-side query of the form `orderByChild('groupId').equalTo(groupId)` over each weekday's agenda node. Without a server-side index, Firebase would download the entire collection to the client to perform this filtering, which is both slow and wasteful of mobile data. To prevent this, a secondary index was configured on the Firebase Realtime Database rules:

```

{
  "rules": {
    "agenda": {
      "$day": { ".indexOn": ["groupId"] }
    }
  }
}

```

This `.indexOn` rule instructs the Firebase server to maintain an internal index on the `groupId` field within each day's agenda node, reducing the time complexity of the deletion query from a full collection scan to a direct index lookup.

3.5.4 Notification Architecture

Push notifications to the caregiver's device are one of the key functional requirements of the system (FR07, NFR04). Two architectural approaches were considered for implementing this functionality.

Approach 1 — Client-side listeners. The Flutter app maintains persistent `onValue` listeners on the `/alertas` node and triggers local notifications when an alert flag changes. This approach is simple to implement but has a critical limitation: it requires the app to be running in the foreground or background at all times. On Android, background processes are subject to the operating system's battery management policies (Doze Mode), which can kill or suspend background listeners, making alert delivery unreliable precisely when the app is not in active use. This would violate NFR03 (mobile client energy efficiency).

Approach 2 — Server-side watchdog (adopted). A dedicated Node.js server, running independently of the mobile app, maintains persistent Firebase listeners and dispatches push notifications through Firebase Cloud Messaging (FCM) whenever an anomalous condition is detected. Because the push notification is delivered by FCM directly to the device’s notification system, the app does not need to be running: the notification arrives regardless of whether the app is open, suspended, or not installed on the foreground. This approach directly satisfies NFR03 and NFR04.

3.5.4.1 Node.js Notification Server

The Node.js server implements three monitoring mechanisms, illustrated in Figure 3.4:

- **Clinical alerts listener** (`db.ref("alertas").on("value")`): monitors the four alert flags in `/alertas` (`dose_em_atraso`, `temperatura_excedida`, `humidade_excedida`, `falha_dispositivo`). When any flag transitions from `false` to `true`, a push notification is sent to the caregiver’s FCM token, retrieved via a one-off `.get()` on `/sistema/token_cuidador`. Internal state locks (`notifiedOverdue`, `notifiedTemp`, `notifiedHum`) ensure that each alert condition triggers at most one notification, rather than firing repeatedly on every RTDB event;
- **Heartbeat listener** (`db.ref("sensores/dht11_interno/ultima_leitura").on("value")`): records the server-side timestamp of the most recent sensor reading from the ESP32-C6. Each time a new reading arrives, the server updates an internal anchor (`lastServerAckTime`) and, if the device had previously been flagged as offline, sends a connectivity-restored notification;
- **Watchdog timer** (`setInterval`, every 3 seconds): compares the current server time against `lastServerAckTime`. If no heartbeat has been received for more than 25 seconds, the device is considered offline and a connectivity-loss notification is dispatched. This threshold accounts for the 4-second sensor publication interval of the firmware plus a tolerance margin for network jitter.

The server is authenticated with Firebase using a service account key, which grants it administrative privileges over the project’s Realtime Database and Cloud Messaging service. The FCM token of the caregiver’s device is registered in `/sistema/token_cuidador` by the mobile app at startup, via the `FirebaseMessaging.getToken()` call, and is read by the server on demand when a notification needs to be sent.

Chapter 4

Testing, Results and Discussion

This chapter describes the experimental validation of the MedBox prototype, in its final, three compartment configuration (Section 3.3.9): the original compartment, a rotating cylinder dedicated to Ben-u-ron 500 mg, and a rotating cylinder dedicated to Ben-u-ron 1000 mg, all three routed through the main tunnel and secondary collector funnel (Section 3.3.5) onto a single, rigidly mounted load cell (Section 3.3.5). The testing process followed an iterative, engineering driven methodology: each phase revealed a specific limitation that informed a concrete design or parameter change before the next phase was conducted. The chapter is organised to reflect this progression honestly, documenting both the outcomes and the reasoning behind each design decision.

The core mechanical idea, a rotating cylinder that singulates and dispenses individual medication units, actuated by a servo and confirmed by a load cell, was successfully implemented and demonstrated in physical hardware for all three compartments. All testing reported from Section 4.4 onward was carried out using Ben-u-ron 1000 mg and 500 mg tablets, the 3D-printed pill substitute, and a 1 euro coin, as reference objects. The only exception is a brief, preliminary bench scale trial (Section 4.2), conducted at an earlier stage of the project to validate the application and firmware logic before the final load cell fixation was capable of reliably resolving weights as small as a single tablet.

A second structural change concerns the firmware's detection logic itself. Earlier in the project, detection was based on a RAW ADC delta relative to a post-tare baseline. This was replaced by a gram calibrated approach: a `CALIBRATION_FACTOR` is obtained once from a known reference weight (a 1 euro coin, 7.5 g) via `scale.set_scale()`, after which all readings are obtained directly in grams via `scale.get_units()`. Detection thresholds are now expressed in grams, `PILL_THRESHOLD_G = 0.3` and `COLLECTION_THRESHOLD_G = 0.15`, rather than as RAW deltas. This change, and its validation against real medication, is the subject of Phase 2 (Section 4.4).

4.1 Test Plan

Table 4.1 summarises the testing phases, their objectives, and the principal metrics recorded.

Table 4.1: Summary of testing phases and objectives.

#	Phase	Objective	Key metric
1	Servo rotation (empty cylinder), all three compartments	Confirm 100% mechanical rotation before introducing any other variable	Rotation success rate
2	Load cell calibration and validation, final platform	Obtain <code>CALIBRATION_FACTOR</code> from a known reference weight (1 euro coin) and validate the resulting gram reading against real medication units	Weight mean, standard deviation, CV
3	Dispensing test, landing rate and detection stability	Confirm that a real dispensing cycle, at the adopted servo speed, reliably lands the unit in the collection tray and that the calibrated gram threshold correctly detects it, for each compartment	Landing rate (out of 50), per compartment
4	Jamming characterisation and mitigation attempt	Identify the cause of jamming observed during loading, and evaluate a proposed mitigation routine, per compartment	Jam occurrence and position, per compartment
5	End-to-end system test, real medications	Validate the full pipeline (app, Firebase, firmware logic, notifications) using real medications across all three compartments	Pass/fail per feature

4.2 Test Environment

From Phase 2 onward, testing was conducted using the load cell and platform installed in the same physical location and configuration as in the final prototype (Section 3.3.5): the enlarged platform and the 0.2 g resolution HX711 load cell, rigidly screw mounted. Not every phase exercises this environment in the same way, and not every phase involves all three compartments together. In particular, the jamming characterisation of Phase 4 is carried out on a single compartment at a time, and does not require the load cell at all, since the metric of interest there is purely mechanical (a successful rotation versus a jam).

Before this configuration was reached, an earlier, preliminary bench trial was carried out with the sole purpose of validating the application and firmware logic (Firebase synchronisation, scheduling, notification pipeline) independently of fine grained weight detection, at a stage of the project when the load cell's fixation was not yet rigid enough to reliably resolve weights as small as a single tablet. This trial used a 1 euro coin (7.5 g) as a stand in reference object, chosen simply

because it was large enough to be read consistently by the load cell at that stage, not because it is representative of a medication unit. Its role and results are summarised briefly in Section 4.7, and are superseded by the tests reported in Sections 4.4 through 4.7.

4.2.1 Historical Note: Early Platform Iterations

Before converging on the final platform and load cell mounting described in Section 3.3.5, two earlier platform geometries were built and tested: an initial flat platform with edge mounted retaining screws, and a subsequent concave platform with centred screws, intended to reduce pill bouncing and guide landing position toward the sensing element. Both iterations, however, shared the same provisional load cell fixation, a groove sized to the sensor's dimensions and reinforced with adhesive tape. Diagnostic testing at the time showed that this fixation, rather than the platform surface geometry, was the dominant source of measurement instability: light mechanical contact anywhere on the surrounding structure produced signal excursions large enough to prevent a reliable drop detection threshold from being established. This finding is not reported here in further detail, since it concerns hardware that was subsequently replaced; it is recorded because it directly motivated the rigid, screw mounted load cell fixation adopted for the final assembly (Section 3.3.5), in which the load cell is constrained only at its mounting screws and left suspended along the rest of its length, rather than resting in a groove. Figure 4.1 shows the two earlier platform geometries and the provisional load cell placement used with each.

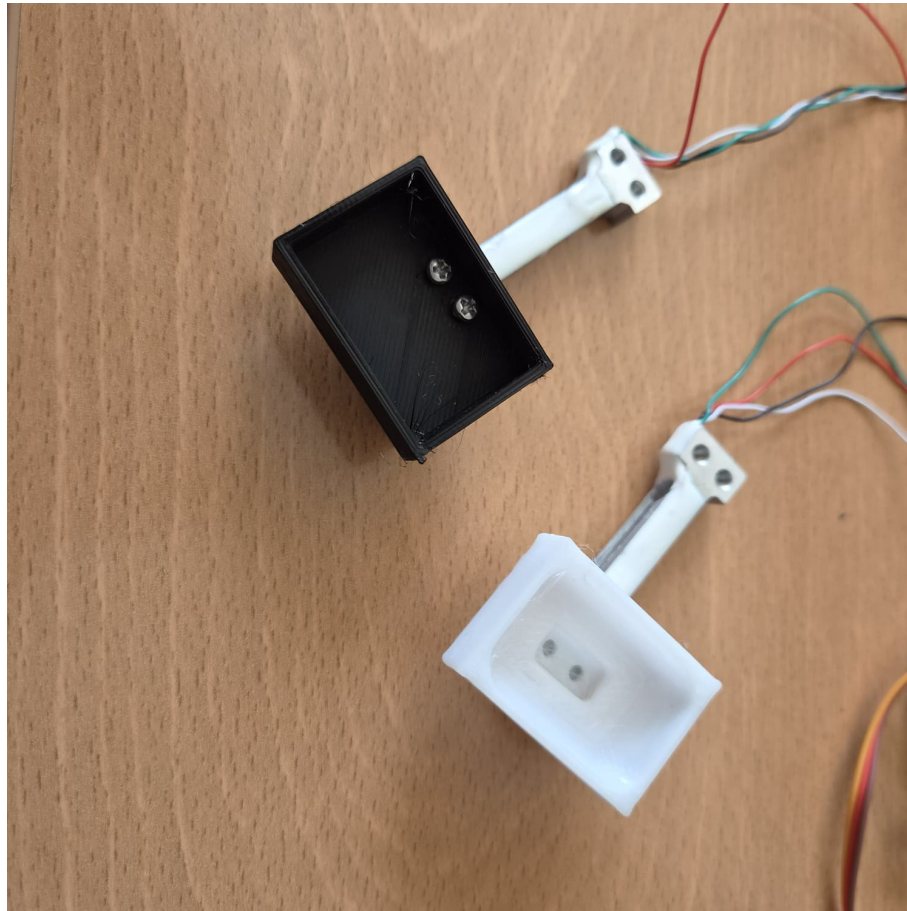


Figure 4.1: Early platform iterations and their provisional load cell placement: (a) initial flat platform with edge mounted screws; (b) concave platform with centred screws. Both used the same groove and adhesive tape fixation for the load cell.

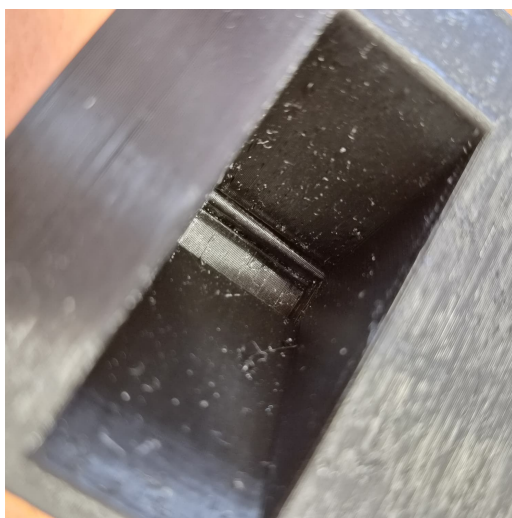
4.3 Phase 1: Servo Rotation Test — Empty Cylinder

Before any pill was introduced into the system, the first test validated the purely mechanical behaviour of each servo cylinder assembly: whether the servo could reliably complete a full $0^\circ \rightarrow 180^\circ \rightarrow 0^\circ$ rotation cycle without stalling or binding. This test was conducted with each cylinder empty in turn, isolating the mechanical variable from any load or friction caused by a pill inside the slot (Figure 4.2).

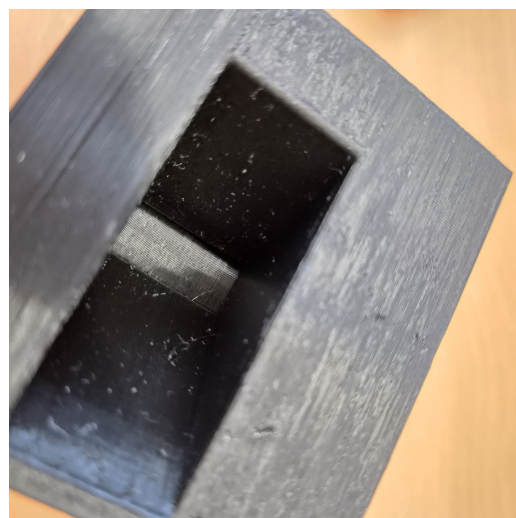
During initial assembly of the original compartment, its cylinder exhibited binding at certain angular positions. This was attributed not to eccentricity or poor concentricity between the printed cylinder and the body cavity, but rather to the dimensional clearance between the cylinder's outer radius and the cavity's inner radius being deliberately kept very small, a requirement to prevent medication from becoming lodged or jammed in the gap during rotation. Given the inherent dimensional tolerances of the FDM printing process, this minimal clearance resulted in localised friction that exceeded the torque output of the servo motor, causing it to stall rather than rotate freely. The issue was resolved by mechanically trimming (*limar*) the inner surface of the body

at the contact points until smooth, resistance free rotation was achieved throughout the full 180° sweep, while preserving a clearance small enough to avoid medication entrapment. The same clearance criterion, validated empirically on the original compartment, was applied directly when printing the two new cylinders (Ben-u-ron 500 mg and 1000 mg), avoiding a repeat of this trial and error adjustment.

After this adjustment, each of the three servos completed the full rotation cycle successfully on every trial, across all tested speeds. This 100% mechanical rotation rate confirms that all three cylinder body assemblies function correctly as isolated mechanisms and justifies proceeding to the calibration and dispensing tests of Phases 2 and 3.



(a) Resting position (0°): slot visible



(b) Rotated position (180°): slot hidden

Figure 4.2: Empty cylinder rotation test, close-up on the slot area: the servo cylinder assembly performing the full $0^\circ \rightarrow 180^\circ \rightarrow 0^\circ$ cycle after the inner body surface was trimmed to eliminate friction induced binding. (a) Resting position, with the slot visible and aligned with the funnel opening; (b) rotated position, with the slot turned away and hidden inside the housing.

4.4 Phase 2: Load Cell Calibration and Validation

4.4.1 Rationale and Protocol

With the final, rigidly mounted load cell in place (Section 3.3.5), the firmware's detection logic was migrated from a RAW delta comparison to a gram calibrated one. This required obtaining a `CALIBRATION_FACTOR` from a known reference weight, and then validating that the resulting gram reading was accurate and repeatable enough to reliably distinguish a real medication unit from the platform's baseline noise.

The procedure, implemented in firmware and run interactively over serial, consists of two stages:

1. **Calibration with a 1 euro coin (7.5 g).** The platform is tared (`scale.tare(20)`), the coin is placed by hand, and 10 RAW readings are collected at fixed intervals and averaged. This is repeated for 3 independent repetitions (each with its own tare), and the final `CALIBRATION_FACTOR` is computed as the overall mean RAW value divided by the known reference weight (7.5 g), then applied via `scale.set_scale()`.
2. **Validation with real medications.** Using the `CALIBRATION_FACTOR` obtained above, the same tare and measure procedure is repeated with a single medication unit as the reference object, this time reading directly in grams (`scale.get_units(1)`), across 5 independent repetitions of 10 readings each.

The firmware sketch used to run this procedure is included in Appendix D.1. Figure 4.3 shows the setup used to weigh the medication units on the final platform.

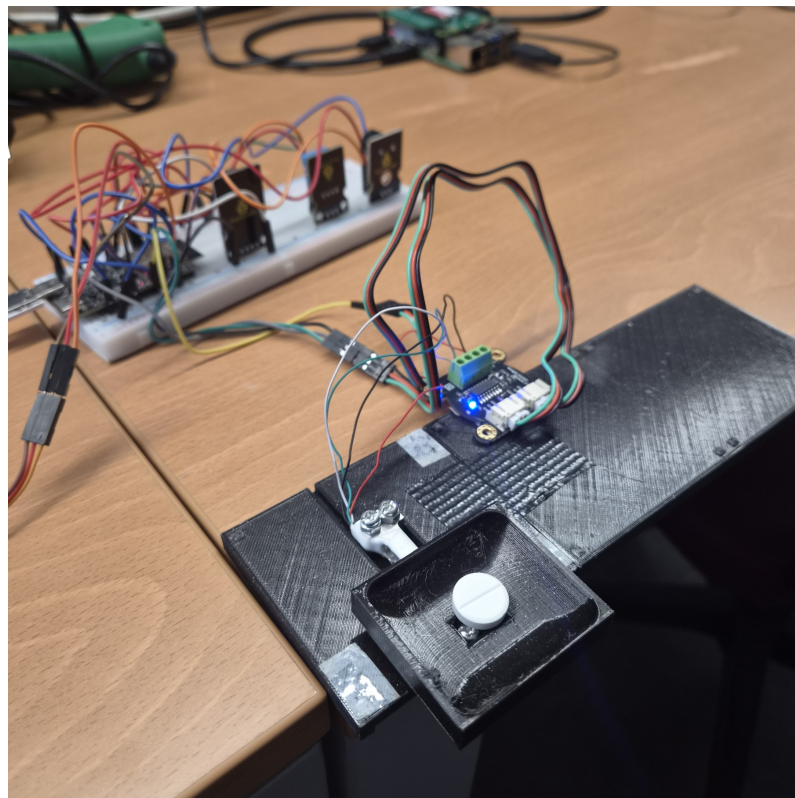


Figure 4.3: Setup used to weigh individual medication units on the final platform, with the calibration sketch running over serial.

This validates, on the actual sensor and mounting used in the final assembly, both the calibration constant itself and the precision with which it resolves a real, single medication unit, before that unit is ever dispensed through the cylinder and funnel path (characterised separately in Phase 3, Section 4.5).

4.4.2 Results

The calibration factor was computed as

$$\text{CALIBRATION_FACTOR} = \frac{\overline{RAW}_{\text{coin}}}{m_{\text{coin}}} \quad (4.1)$$

where $\overline{RAW}_{\text{coin}}$ is the overall mean RAW value obtained across the three calibration repetitions, and $m_{\text{coin}} = 7.5 \text{ g}$ is the known reference weight of the coin.

Table 4.2 summarises the coin calibration. The three repetitions were consistent, with a min-max variation of 0.41% around the overall mean RAW value of -79875 , yielding, from Equation 4.1, $\text{CALIBRATION_FACTOR} = -10650.0$.

Table 4.2: Load cell calibration, 1 euro coin (7.5 g), final platform, rigid fixation.

Repetition	Mean RAW value
1	-79878
2	-80039
3	-79709
Overall mean	-79875

Table 4.3 summarises the five validation repetitions with the Ben-u-ron 1000 mg tablet, using the calibration factor above.

Table 4.3: Validation against a real medication unit (Ben-u-ron 1000 mg), final platform.

Repetition	Mean RAW value	Weight (g)
1	-12978	1.219
2	-12975	1.218
3	-12919	1.213
4	-13022	1.223
5	-12896	1.211
Mean	-12958	1.217

The mean measured weight, 1.217 g, is consistent with the nominal composition of a Ben-u-ron 1000 mg tablet [**benuron_rcm**], and the standard deviation across the five repetitions was 0.004 g, corresponding to a coefficient of variation of approximately 0.3%. This low variance, obtained on the final rigid fixation and directly on a real medication rather than a heavier substitute, confirms that the instability previously observed with the earlier, adhesive tape mounted iterations (Section 4.2.1) was indeed eliminated by the rigid screw mounted fixation: no residual sensitivity of comparable magnitude was observed.

Following the same protocol, a Ben-u-ron 500 mg tablet, the lightest medication unit used in the system, was also weighed on the final platform. Table 4.4 summarises the five repetitions obtained.

Table 4.4: Validation against a real medication unit (Ben-u-ron 500 mg), final platform.

Repetition	Weight (g)
1	0.602
2	0.592
3	0.596
4	0.605
5	0.597
Mean	0.598

The mean measured weight, 0.598 g, with a standard deviation of 0.005 g across the five repetitions (coefficient of variation of approximately 0.8%), confirms that the final platform also resolves this lighter tablet reliably, with variability of the same order of magnitude observed for the heavier Ben-u-ron 1000 mg tablet.

Based on these two results, the firmware’s detection thresholds were defined in grams, with a comfortable margin below the lightest medication units handled by the system: `PILL_THRESHOLD_G` = 0.3 for drop detection, and `COLLECTION_THRESHOLD_G` = 0.15 for confirming a stable, empty platform state after collection. Both the 3D-printed substitute and the Ben-u-ron 500 mg tablet weigh approximately 0.6 g, so these thresholds were set low enough to remain safely below that weight, while staying clearly above the platform’s baseline noise. Both values are used, unmodified, in the dispensing tests of Section 4.5.

4.5 Phase 3: Dispensing Test, Landing Rate and Detection Stability

4.5.1 Protocol

With rotation confirmed at 100% for all three compartments (Section 4.3) and the gram threshold validated by direct placement (Section 4.4), this phase evaluates whether an actual dispensing cycle, servo rotation followed by the pill falling through the funnel and tunnel onto the platform, reliably lands the medication inside the collection tray, and whether the calibrated threshold correctly registers the resulting drop.

A servo speed of 5 ms per degree of rotation (approximately 900 ms per full 180° sweep) was adopted for all dispensing tests reported in this section. This speed was chosen as an acceptable one, allowing the medication unit to be dispensed quickly while avoiding the unpredictable behaviour that excessively fast rotations were observed to cause during earlier testing.

This test was performed independently for each of the three compartments, dispensing the appropriate reference object in each case (the 3D-printed substitute for the original compartment, and the Ben-u-ron 500 mg and 1000 mg tablets for the two new compartments). For each compartment, 50 dispensing cycles were performed at the adopted speed, recording for each cycle whether the ejected unit landed inside the collection tray. Figure 4.4 shows the test scenario.



Figure 4.4: Dispensing test scenario: servo cylinder assembly, funnel, and final platform with rigidly mounted load cell, used across the 50 dispensing cycles of Phase 3.

4.5.2 Results

For this test, a dedicated firmware sketch was used: a single key press over serial triggers one full $0^\circ \rightarrow 180^\circ \rightarrow 0^\circ$ rotation of the servo, after which the load cell is read directly in grams for a short period, so that the resulting weight profile can be observed. This sketch is included in Appendix D.2.

Two example runs of 30 consecutive readings, obtained with this sketch after dispensing a Ben-u-ron 1000 mg tablet, are provided in full as raw serial log output in Appendix D.5, rather than as a plotted figure. Drop confirmation requires five consecutive stable readings above threshold, not a single reading, so the exact shape of any momentary spike at the moment of impact is not itself load-bearing for the result, only the stable value the reading settles at afterwards. Excluding two isolated readings, 9.323 g in the first run and 1.649 g in the second, attributable to touch rather than the drop itself (Section 4.5.3), the remaining readings in both runs stayed within 1.148 g and 1.251 g, closely matching the 1.217 g mean weight established for this tablet in the calibration validation (Table 4.3). This confirms that, once landed, the platform reads the dispensed unit's weight consistently and correctly, and that the calibrated `PILL_THRESHOLD_G` of 0.3 g is comfortably exceeded by an actual drop.

An important difference between compartments was observed regarding the timing of this spike. Compartment 1 sits directly above the main tunnel and the platform (Section 3.3.5), so a unit

dispensed from it falls almost immediately and the resulting spike appears shortly after the rotation completes. Compartments 2 and 3, by contrast, dispense laterally into the secondary collector funnel before the unit reaches the main tunnel, a longer path that takes noticeably more time to traverse. Rather than taking a single weight reading immediately after the rotation, the firmware's drop detection is implemented as a polling loop that keeps checking the weight repeatedly for up to `TIMEOUT_QUEDA_MS` (5000 ms), sampling every `INTERVALO_POLLING_MS` (300 ms), starting as soon as the dispensing rotation completes. This generous polling window, rather than a single read taken too early, is what accommodates the longer travel time from Compartments 2 and 3: as long as the unit reaches the platform before the 5 second window elapses, the drop is still detected correctly, regardless of which compartment it was dispensed from. This is the mechanism currently implemented in the production firmware, and no separate, compartment-specific startup delay was found to be necessary in addition to it.

Table 4.5 summarises the landing results obtained for the three compartments.

Table 4.5: Landing rate per compartment, 50 dispensing cycles each.

Compartment	Cycles	Landed in tray	Success rate
Original (substitute)	50	33	66%
Ben-u-ron 500 mg	50	35	70%
Ben-u-ron 1000 mg	50	48	96%

Ben-u-ron 1000 mg achieved by far the highest landing rate of the three compartments, while the two lighter units, Ben-u-ron 500 mg and the 3D-printed substitute, performed noticeably worse. This is consistent with basic projectile behaviour: a lighter unit is more easily deflected from a straight drop by small variations in its release point or orientation, and is more likely to bounce or roll away from the collection tray on impact, rather than settling immediately where it lands. A heavier tablet such as Ben-u-ron 1000 mg is comparatively less affected by these perturbations. A possible direction for future work is to redesign the funnel geometry so as to reduce the drop velocity of lighter units before they reach the platform, or to increase the height of the platform's front wall, reducing the chance that a unit that lands correctly still bounces or rolls out of the tray afterwards.

4.5.3 Detection Stability Under Real Dispensing Conditions

During the dispensing trials, momentary weight spikes were occasionally observed on the platform. Diagnostic testing showed that these spikes are not caused by mechanical coupling between the servo and cylinder assembly and the load cell platform: with the platform at rest and no contact from the user, the baseline reading proved essentially stable, with no meaningful variation attributable to the mechanism itself. The spikes that were observed occurred specifically when a finger touched the platform, most notably while retrieving the dispensed unit during the collection window.

This is particularly relevant for the intended user population. Elderly users, who may have reduced motor dexterity, are more likely to make repeated or prolonged contact with the platform while picking up a small tablet, and this contact can be registered by the load cell as a spurious weight change, given how sensitive the sensor is at this scale. This behaviour is expected to be assessed more thoroughly during the usability testing planned with elderly participants.

To ensure that the platform is read correctly, both before a dispensing cycle and while confirming that the medication has actually landed, two safeguards were introduced in the firmware (the final version used for this test is included in Appendix D.4): a settling delay after each tare operation, and a dedicated pre-dispense stability check, `lerPesoBaseEstavel()`. This function collects a short series of baseline readings, together with their mean and their minimum to maximum variation, and only allows the dispensing cycle to proceed once the baseline is confirmed to be stable. Together with the post-rotation delay described in the previous section, which accounts for the extra travel time from Compartments 2 and 3, these mechanisms ensure that the load cell has settled before a medication unit is dropped, and that the detection window is aligned with the unit's actual arrival, so that the resulting weight delta can be attributed reliably to the dispensed unit rather than to residual instability or to user contact.

4.6 Phase 4: Jamming Characterisation and Mitigation Attempt

4.6.1 Observed Cause of Jamming

While loading medication units into each compartment for the dispensing tests of the previous sections, it was observed that a unit does not always enter the cylinder's slot in the same orientation. Figure 4.5 illustrates this for a Ben-u-ron 1000 mg tablet: in the desired orientation, the tablet lies flat within the slot without obstructing the surrounding cavity; in the undesired orientation, the tablet becomes wedged against the edge of the slot, preventing the cylinder from completing its rotation and causing a jam.

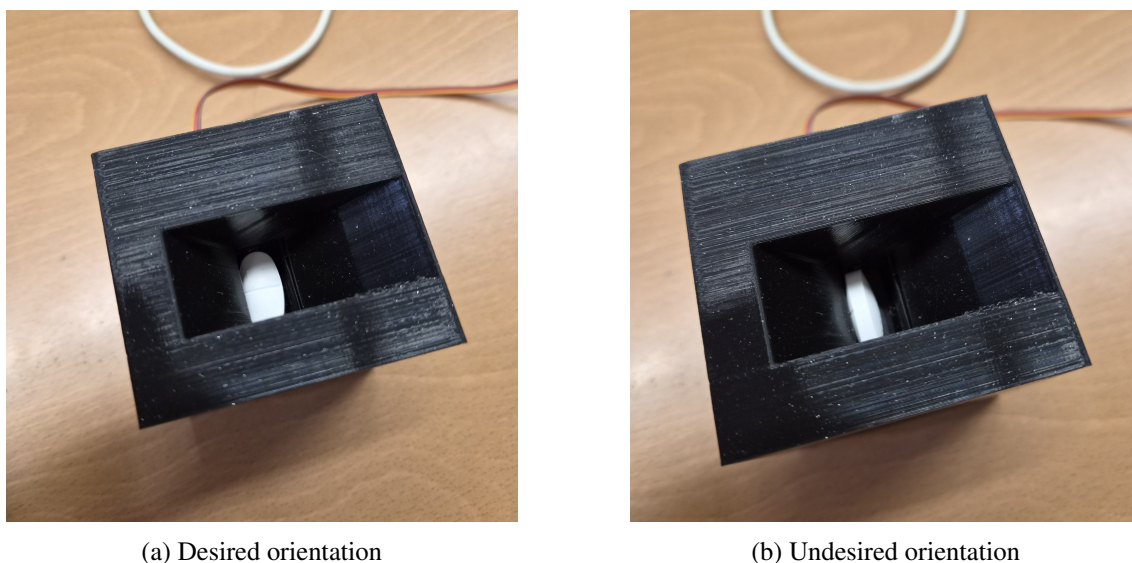


Figure 4.5: Desired and undesired orientation of a Ben-u-ron 1000 mg tablet inside the cylinder slot: (a) the tablet lies flat within the slot without obstructing the surrounding cavity; (b) the tablet is wedged against the edge of the slot, preventing the cylinder from completing its rotation and causing a jam.

4.6.2 Mitigation Attempt

To reduce the incidence of jams caused by this orientation issue, a mitigation routine was implemented in firmware. Before each dispensing rotation, the servo performs a small amplitude oscillation, intended to reposition a unit that may be wedged at the slot entrance before the full $0^\circ \rightarrow 180^\circ$ rotation is attempted. The oscillation parameters used in the device's main firmware (3 repetitions, 0° to 15° , 150 ms dwell at each extreme) were tried first, and were found insufficient to meaningfully reduce jamming. A considerably more aggressive configuration was then tested: 10 repetitions, a wider 0° to 30° oscillation, a shorter 25 ms dwell at each extreme, and a faster 3 ms per degree step (compared to the 5 ms per degree used for the main dispensing rotation). Jamming still occurred with this configuration, indicating that the oscillation, however aggressive, does not by itself solve the problem, since it only disturbs the cylinder itself rather than the pile of units resting above it in the reservoir. The corresponding firmware sketch, with these final parameters, is included in Appendix D.3.

4.6.3 Protocol

For each of the three compartments, the funnel was loaded with 5 units (the 3D-printed substitute for the original compartment, and Ben-u-ron 500 mg / 1000 mg tablets for the two new compartments), with the first unit always placed by hand in the correct orientation inside the slot before starting, so that any jam observed could be attributed to units above it in the funnel rather than to an incorrect initial setup. Dispensing cycles were then triggered one at a time, using the mitigation enabled firmware described above, and each cycle was classified as a successful single unit

dispense or a jam. This was repeated across 5 independent loading trials per compartment, each starting from a full reload of 5 units. Figure 4.6 shows the funnel loaded with the 5 units used in each trial.



Figure 4.6: Funnel loaded with 5 medication units at the start of a jamming trial.

4.6.4 Results

Table 4.6 reports, for each compartment and each of the 5 loading trials, the dispense attempt (out of 5) at which the cylinder jammed.

Table 4.6: Position of the first jam (dispense attempt, out of 5 loaded units) across 5 loading trials, per compartment, with the mitigation routine active.

Compartment	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Mean position
Original (substitute)	4	3	3	2	4	3.2
Ben-u-ron 500 mg	2	2	3	2	3	2.4
Ben-u-ron 1000 mg	3	4	3	4	3	3.4

All three compartments jammed in every one of the 5 trials, so the mitigation routine did not eliminate jamming for any of the three medications. What differs between compartments is how many single unit dispenses succeeded, on average, before the jam occurred: Ben-u-ron 1000 mg reached the highest mean position (3.4), the 3D-printed substitute an intermediate one (3.2), and Ben-u-ron 500 mg the lowest (2.4), jamming consistently earlier than the other two.

For Ben-u-ron 1000 mg, jams were most often observed when a tablet entered the slot on its side, in the undesired orientation described above, while a second tablet partially occupied the space left free beside it, blocking rotation (Figure 4.7). A related failure mode was also observed in which two tablets competed simultaneously for the same slot opening and wedged against each other, so that neither one actually entered the slot.



Figure 4.7: Example of a jam in the Ben-u-ron 1000 mg compartment: a tablet in the undesired orientation, with a second tablet partially occupying the remaining space, blocking rotation.

The Ben-u-ron 500 mg compartment jammed earlier and more consistently than the other two. This compartment reuses the same cylinder body as the rest of the assembly, rather than one scaled down specifically for the smaller 500 mg tablet, so a larger amount of free space remains around the slot inside the housing. Because the 500 mg tablet is circular, its slot is also cut with a rounded profile to match, rather than the more constrained rectangular opening used for the two elongated units; this rounded opening does not naturally separate adjacent tablets the way a narrower rectangular slot does, so more tablets can sit close to the funnel's outlet at once (Figure 4.8), increasing the chance that more than one unit is competing for the slot at any given moment. In addition, because of its circular geometry, the 500 mg tablet was observed to settle in a comparatively more random arrangement within the compartment, without a preferred long axis to align itself along,

unlike the other two, more elongated, units, which further compounds the crowding described above.

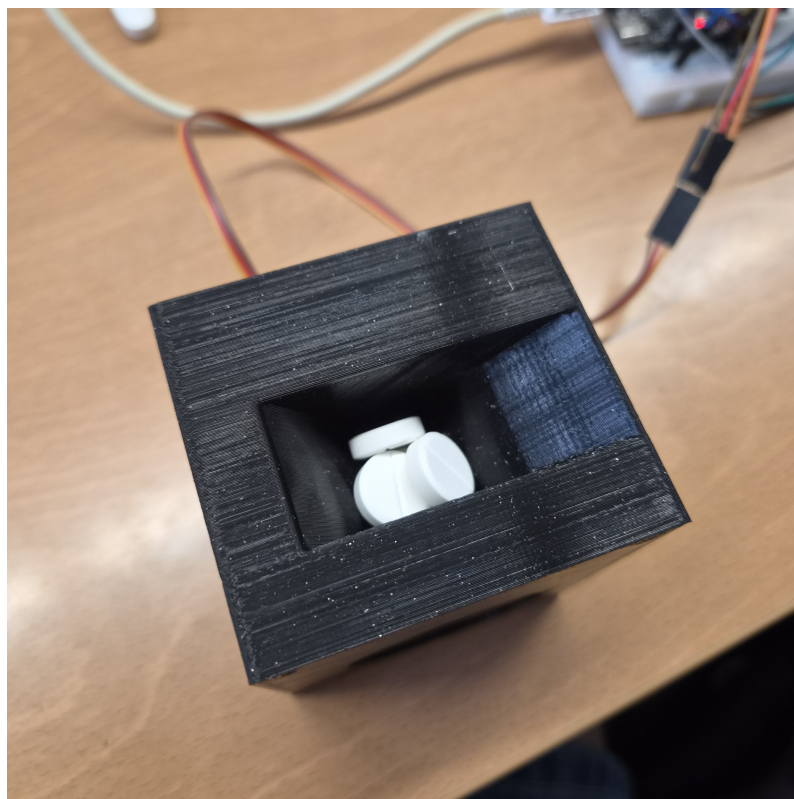


Figure 4.8: Ben-u-ron 500 mg compartment: tablets crowding close to the funnel outlet, due to the free space left around the slot by the shared cylinder body and the rounded slot profile matching the tablet's circular geometry.

This result runs counter to the initial expectation that a circular tablet, having no single preferred long axis, might be less prone to orientation dependent jamming than an elongated one. In practice, the Ben-u-ron 500 mg compartment jammed earlier than both the printed substitute and Ben-u-ron 1000 mg, suggesting that, for this geometry, the dominant failure mode is not orientation at the slot itself but the more unpredictable way circular tablets settle and crowd together above it. Redesigning the compartment or cavity specifically around the settling behaviour of circular tablets, rather than reusing the same cylinder body across all three compartments, is identified here as an item for future work.

During this testing, it was also observed that manually agitating the entire compartment, rather than only the cylinder, reliably cleared a jam. This suggests that a mitigation strategy acting on the reservoir as a whole, rather than on the cylinder alone, could be considerably more effective than the pre-dispense oscillation attempted in this section. This observation is closely related to the PICK-SHAKE-DROP concept discussed and discarded in Section 3.3.2.2 on the grounds of mechanical complexity; a hybrid approach, combining the current rotating cylinder with a small amount of reservoir level agitation rather than a full vibrating platform, is recorded here as a

promising direction for future work.

Cross-compartment interaction at the shared collector funnel, that is, whether a unit still in transit from one compartment can interfere with a unit released shortly afterwards from another, was not part of this round of testing and remains an item for future work.

4.7 Phase 5: End-to-End System Test

This phase validates the application, Firebase synchronisation, and firmware logic end to end, rather than the mechanical dispensing behaviour characterised in the previous phases. It is reported in two stages: a preliminary bench trial, conducted early in the project with a coin as a stand in reference object, before the final load cell fixation was ready; and a final validation, conducted on the complete assembly with real medications dispensed across all three compartments.

4.7.1 Preliminary Bench Validation (Coin, App and Firmware Logic Only)

At an earlier stage of the project, before the final load cell fixation was capable of reliably resolving weights as small as a single tablet, a preliminary bench test was conducted using a 1 euro coin as a stand in reference object, chosen for its consistent readability at that stage rather than as a representative medication weight. Figure 4.9 shows this bench setup. This test validated the application, Firebase synchronisation, and the firmware's dispensing/detection and notification logic end to end, independently of fine grained weight detection on the final sensor. All core features exercised at this stage, scheduled dispensing, drop and collection detection, history logging, real time monitor updates, and the push notification pipeline, passed. This trial is reported here only for context; it is superseded by the final platform test below, which uses real medications throughout.

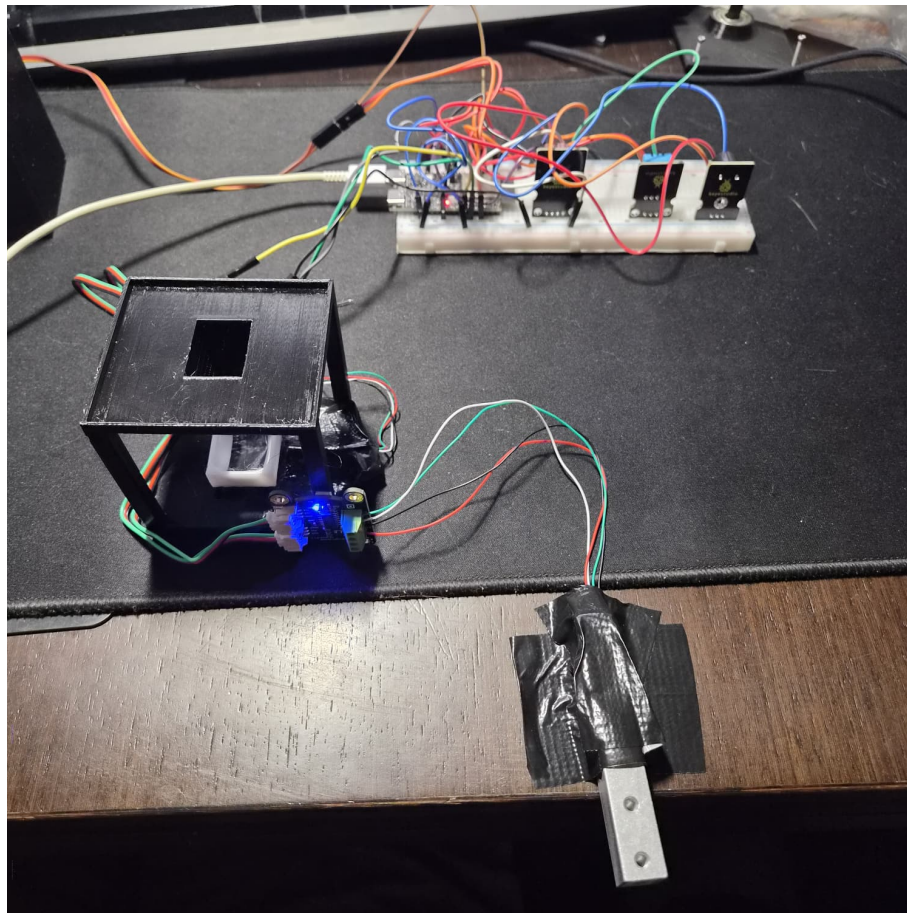


Figure 4.9: Preliminary bench setup used to validate the application and firmware logic, with a 1 euro coin used as a stand in reference object.

4.7.2 Final Validation (Real Medications, Three Compartments)

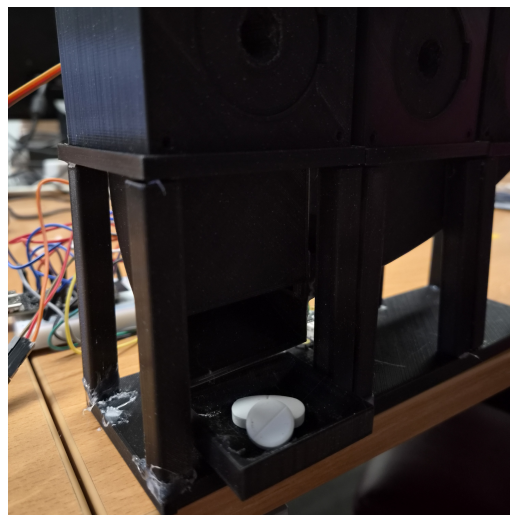
The end-to-end test was repeated on the final, three compartment assembly, dispensing real medications (the substitute unit, Ben-u-ron 500 mg, and Ben-u-ron 1000 mg) rather than the coin, and using the gram calibrated detection logic validated in Phase 2 (Section 4.4). Figure 4.10 shows the complete final system used for this test. All other firmware logic, including the dispensing routine, alarm escalation, offline caching, and notification triggers, was used unmodified, and exercised across all three compartments.



(a) Front view



(b) Side view



(c) Detail view

Figure 4.10: Final end-to-end test setup, shown from three angles: ESP32-C6 development board with full sensor circuit, final 0.2 g resolution load cell, three compartment dispensing mechanism, and mobile app running on the caregiver's device.

4.7.2.1 Functional Validation

Table 4.7 summarises the outcome of each functional check performed on the final assembly with real medications.

Table 4.7: Functional validation results, pass/fail per system feature, final assembly, real medications.

Feature	Verification method	Result
Dose dispensed at scheduled time, correct compartment	Serial Monitor + visual	Pass
Drop confirmed (weight above PILL_THRESHOLD_G)	Serial Monitor	Pass
Collection confirmed (platform stable, below COLLECTION_THRESHOLD_G)	Serial Monitor	Pass
Dispensing event written to /history	Firebase console	Pass
/toma_atual updated correctly	Firebase console	Pass
/agenda updated to taken	Firebase console	Pass
History tab shows correct entry	App screenshot	Pass
Monitor tab updates in real time (<5 s)	App observation	Pass
DHT11 temperature reading correct	Visual plausibility check ¹	Pass
DHT11 humidity reading correct	Visual plausibility check ¹	Pass
Buzzer activates after tolerance window	Audio observation	Pass
Buzzer escalation (1→2→3 beeps)	Audio at accelerated test timing	Pass
Late collection (taken_late) after 3rd alarm level	Serial Monitor + Firebase + push	Pass
Push notification: overdue dose	Notification tray (app closed)	Pass
Push notification: temperature exceeded	Lower threshold in Settings	Pass
Push notification: device offline (watchdog)	Power off ESP32	Pass (26 s observed)
Anticipated dispensing (UC4), no duplication	App On-Demand → /agenda	Pass
Two doses from different compartments at the same time, sequential dispensing, correct attribution	Serial Monitor + Firebase	Pass

All eighteen functional checks passed on the final assembly, dispensing real medications across all three compartments, using the gram calibrated detection logic rather than the RAW based

logic and the coin used in the preliminary bench trial. No behavioural differences were observed relative to the preliminary trial: the same firmware logic, exercising drop detection, collection confirmation, alarm escalation, offline caching, and the notification pipeline, produced consistent outcomes with real tablets as it did with the coin, confirming that the earlier bench trial was a valid proxy for the software and data layers of the system, independently of the mechanical dispensing behaviour characterised in Phases 1 through 4.

Two recorded sessions illustrate this end to end behaviour concretely and are summarised here as supporting evidence.

Three medications scheduled at the same time. A group of three doses was scheduled for the same time slot: one entry labelled in the schedule as Amoxicillin with clavulanic acid (Bluepharma, 875 mg + 125 mg), dispensed from the original compartment, together with Ben-u-ron 1000 mg and Ben-u-ron 500 mg. The medication name used for the first entry is illustrative rather than literal: the original compartment dispensed the same 3D-printed pill substitute used throughout its testing (Section 3.3.7), not an actual amoxicillin tablet; a medication name was nonetheless required to populate that schedule entry for this end-to-end session, and Amoxicillin with clavulanic acid was the name chosen for that purpose. The firmware dispensed all three sequentially, as described in Section 3.4.6. The baseline weight read before each unit rose consistently with the number of units already resting on the platform, from approximately 0 g before the first unit to 0.593 g before the second and 1.196 g before the third, confirming that a single tare at the start of the group, followed by a freshly measured baseline before each unit, correctly isolates the weight contributed by each new unit without requiring a hardware tare between medications, exactly as described in Section 3.4.6. A momentary spike of 7.8 g was recorded 1268 ms into the first unit's drop detection window, attributable to a touch or vibration rather than the tablet landing; the stability requirement introduced in Section 4.5.3 correctly prevented this spike from being counted, and the drop was only confirmed 1.5 s later, once the weight had settled at a stable value consistent with the expected tablet weight. During the subsequent collection window, several further spikes of similar or larger magnitude were recorded while the three tablets were being retrieved by hand; the ten-reading stability requirement kept the collection counter at zero throughout this period and only confirmed collection once the platform had genuinely returned to, and remained at, a near zero weight. The dispensing event concluded with `estado = tomado`, drop and collection both confirmed.

Two units of the same medication in one event. A single dose of Ben-u-ron 1000 mg with a prescribed quantity of two was dispensed from the original compartment. The first unit's drop was confirmed cleanly, with the delta reaching a stable value close to 1.2 g within five consecutive readings. Before the second unit, the freshly measured baseline correctly reflected the first tablet still resting on the platform, at approximately 1.2 g, rather than assuming an empty platform. During the second unit's drop detection window, several readings showed a partial delta of only 0.03 g to 0.12 g above the baseline, consistent with the tablet still settling after release rather

than having fully arrived; because these partial deltas did not exceed `PILL_THRESHOLD_G` for five consecutive readings, they were correctly not counted toward confirmation, and the drop was only confirmed once the weight reached a stable value around 1.2 g above the new baseline, corresponding to the second tablet having fully landed. This is a direct illustration of the value of the stability based confirmation introduced in Section 4.5.3: a single reading based threshold, as used earlier in this project, could plausibly have confirmed the drop prematurely on one of these partial readings.

The complete, unedited Serial Monitor output for both sessions described above is provided in Appendix D.6.

4.7.2.2 Notification Latency

Push notification latency was not formally timed across repeated trials, and remains identified as future work (Section 5.1). One data point is available from the device offline edge case (Section 4.7.2.3): the connectivity loss notification was observed 26 s after the ESP32-C6 was powered off, consistent with the architecture’s design, a 25 s watchdog threshold in the Node.js server (Section 3.4.4) plus a small margin attributable to the server’s own polling interval. For the remaining notification types, overdue dose and temperature exceeded, latency was observed qualitatively to be on the order of a few seconds from the corresponding Firebase write, consistent with the server’s persistent listeners dispatching immediately on value change rather than polling, but was not repeatedly timed.

Figure 4.11 shows two examples of push notifications received on the caregiver’s device during testing, delivered via Firebase Cloud Messaging while the application was closed: a connectivity-loss alert, triggered by the watchdog timer described above, and a temperature-exceeded alert, triggered by the environmental threshold check (Section 3.4.2.4).

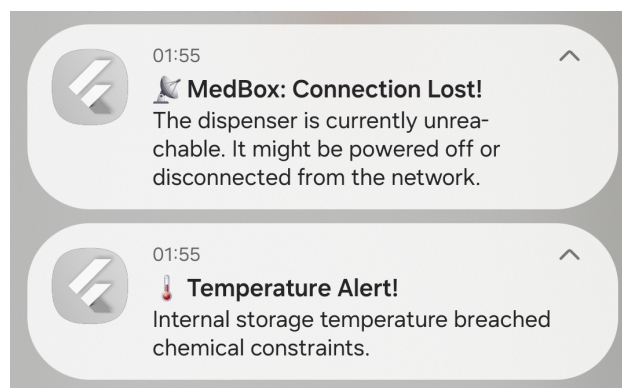


Figure 4.11: Examples of push notifications received on the caregiver’s device via Firebase Cloud Messaging, with the application closed: a device connectivity-loss alert (top) and a temperature-exceeded alert (bottom).

4.7.2.3 Edge Case Tests

The edge case scenarios in Table 4.8 are evaluated on the final, three compartment hardware, where network and power interruptions, command re-entrancy, and collection stability behaviour under physical disturbance can be assessed under conditions representative of the deployed system.

Table 4.8: Edge case test scenarios, three compartment prototype.

Scenario	Expected behaviour	Result
Wi-Fi disconnected during dispensing	Dispensing completes via cache + RTC; <code>/history</code> written on re-connection	Pass
Device powered off at scheduled time	On reboot: <code>device_offline</code> logged; <code>falha_dispositivo</code> alert sent	Pass
App closed when notification arrives	Notification visible in tray via FCM	Pass
UC4 followed by scheduled dose	Dose not re-dispensed (<code>/agenda</code> already taken)	Pass
Same dose triggered twice in same minute	Re-entrancy guard blocks duplicate	Pass
Bouncing object during collection window	Stability counter resets on weight spike; confirms only after 10 consecutive near zero readings	Pass
Two doses from different compartments at the same scheduled time	Sequential dispensing with settling delay; correct compartment and history attribution for each	Pass

4.8 Discussion

4.8.1 Mechanical Validation

The Phase 1 result, 100% rotation success across all three cylinders after the friction fix applied to the original compartment, confirms that the rotating-cylinder singulation mechanism is intrinsically reliable as an actuator, and that the fixed design parameters identified in Section 3.3.4 (clearance, funnel angle, servo coupling) transfer directly to newly dimensioned cylinders without requiring the trial-and-error adjustment needed for the original compartment. The friction issue encountered during initial assembly is representative of the dimensional tolerances inherent to FDM 3D printing, and its resolution by mechanical trimming illustrates the iterative nature of rapid-prototyping development.

The rigid, screw-mounted load cell fixation adopted for the final platform (Section 3.3.5) directly addressed the instability observed informally with the earlier, adhesive-tape-mounted iterations (Section 4.2.1). This was confirmed quantitatively in Phase 2: both Ben-u-ron tablets tested, 1000 mg and 500 mg, were weighed with a coefficient of variation below 1% on the final platform, low enough to support a reliable gram-based detection threshold for both medications.

The servo speed of 5 ms per degree, adopted directly rather than through a further multi-speed comparison, proved viable for all three compartments in Phase 3: rotation completed quickly and predictably in every trial, with no cases of the pill exiting the slot early or being carried past the release position. What differed substantially between compartments was not the rotation itself but the landing rate in the collection tray: Ben-u-ron 1000 mg landed successfully in 96% of 50 dispensing cycles, while the two lighter units, Ben-u-ron 500 mg and the 3D-printed substitute, landed in only 70% and 66% of cycles respectively. This points to the weight of the unit itself, rather than the servo speed, as the dominant factor: lighter units are more easily deflected by small variations in release point or orientation and are more prone to bouncing or rolling out of the tray on impact. Addressing this discrepancy, for example by reshaping the funnel to reduce drop velocity or by raising the platform's front wall, is identified as a candidate improvement in Section 5.1.

4.8.2 Multi-Pill Dispensing and Shared Drop Path

The jamming behaviour first observed with the original compartment's non-square slot did not disappear in the two new compartments; if anything, the results from Phase 4 complicate the picture. With the mitigation routine active, all three compartments jammed in every one of the five loading trials performed, so the oscillation-based mitigation did not eliminate jamming for any of the three medications tested, even at a considerably more aggressive configuration than the one used in the main firmware. The compartments did differ in how many single-unit dispenses succeeded, on average, before a jam occurred: Ben-u-ron 1000 mg reached the highest mean position, the 3D-printed substitute an intermediate one, and Ben-u-ron 500 mg the lowest. This last result runs counter to the initial expectation that a circular tablet, having no single preferred long axis, might be less prone to orientation-dependent jamming than an elongated one; in practice, the circular Ben-u-ron 500 mg tablet jammed earliest and most consistently of the three, which Section 4.6 attributes to a combination of shared cylinder-body geometry and a more random settling pattern for circular units, rather than to orientation at the slot itself. A firmware-level oscillation therefore appears insufficient on its own; the manual observation that agitating the whole compartment, not just the cylinder, reliably cleared a jam suggests that a mechanical intervention at the reservoir level is a more promising direction, discussed further in Section 5.1.

Cross-compartment interaction at the shared collector funnel and main tunnel, that is, whether a unit still in transit from one compartment can interfere with a unit dispensed shortly afterwards from another, was not part of this round of testing and remains an open question, to be addressed as future work.

4.8.3 System-Level Assessment

Taken together, Phases 1 through 4 show a mechanism that is mechanically sound at its core, a rotating cylinder, actuated by a servo, releasing a single medication unit per cycle, detected by a load cell, but not yet fully reliable at its edges: landing rate depends noticeably on tablet weight, and jamming persists across all three compartments despite a firmware-level mitigation attempt. The load cell mounting and gram-based detection logic, by contrast, were validated successfully and are no longer considered an open item.

Phase 5 closes the loop on the software and data layers: all eighteen functional checks and all seven edge case scenarios passed on the final, three compartment assembly, dispensing real medications rather than the coin used in the preliminary bench trial. This confirms that the application, Firebase synchronisation, offline caching, alarm escalation, and notification pipeline behave correctly end to end under conditions representative of deployment, including simultaneous connectivity loss, device power loss, and command re-entrancy. Combined with the mechanical results from Phases 1 through 4, the overall picture is one of a system whose software and sensing logic are solid and thoroughly exercised, sitting on top of a mechanical mechanism that is functionally proven but still exhibits weight-dependent landing reliability and unresolved multi-unit jamming, both of which are carried forward as concrete, scoped future work (Section 5.1) rather than left as open-ended gaps.

4.9 Limitations

- **All planned testing phases completed:** rotation testing, load cell calibration, dispensing/landing rate testing, jamming characterisation, and the end-to-end functional and edge-case tests with real medications across all three compartments (Sections 4.3 through 4.7.2) were all completed, with results reported throughout this chapter;
- **Load cell mounting and threshold validation:** the rigid, screw-mounted load cell fixation implemented in Section 3.3.5 resolved the instability observed informally with earlier platform iterations (Section 4.2.1); both Ben-u-ron tablets tested were weighed with a coefficient of variation below 1%, supporting a reliable gram-based detection threshold (Section 4.4);
- **Landing rate depends on tablet weight:** the lighter units, the 3D-printed substitute and Ben-u-ron 500 mg, landed successfully in the collection tray in only 66% and 70% of 50 dispensing cycles respectively, compared to 96% for the heavier Ben-u-ron 1000 mg (Section 4.5). This is attributed to the lower mass making lighter units more susceptible to bouncing or rolling out of the tray, rather than to the servo speed itself;

- **Non-square slot geometry and multi-unit jamming:** jamming was observed in every one of the five loading trials performed for all three compartments, even with an oscillation-based mitigation routine active (Section 4.6). Ben-u-ron 500 mg, despite its circular geometry, jammed earliest and most consistently of the three, contrary to the initial expectation that a circular tablet would be less orientation-dependent. A firmware-only mitigation was found insufficient; a mechanical redesign, or an agitation mechanism acting on the reservoir rather than the cylinder alone, is identified as necessary future work;
- **Shared drop path across compartments:** the main tunnel and secondary collector funnel introduced for the final assembly (Section 3.3.5) route all three compartments onto a single load cell; whether units dispensed from different compartments in close succession can interact within this shared path was not tested and remains an open question (Section 4.6);
- **Firestore write reliability during long idle periods:** the firmware checks that the Firestore client reports itself ready before attempting a write, but does not verify that each individual write actually succeeded. During bench testing, this occasionally produced silent write failures to `/history`, `/agenda`, and `/toma_atual` after long periods without any Firestore traffic, such as an extended collection-waiting window, when the underlying secure connection had been closed in the background without the client's own readiness check detecting it. This was diagnosed but deliberately left unresolved, as a scoped trade-off given the time available; adding an explicit success check with a local retry cache, mirroring the mechanism already used for genuine offline periods, is a direct and low-risk fix identified for future work;
- **Environmental sensor validation method:** the DHT11 temperature and humidity readings were judged plausible against ambient conditions during testing, rather than validated against a calibrated reference thermometer or hygrometer;
- **Data security (NFR08):** Firestore Realtime Database rules grant unrestricted read/write access during development;
- **Synchronous firmware architecture:** the dispensing routine blocks the `loop()` thread for its full duration, regardless of which of the three compartments is dispensing;
- **Fail-safe hatch (*alçapão*):** autonomous retraction of uncollected doses was not implemented. When the collection window elapses without the dose being retrieved, the firmware marks the dose as `missed`, dispatches the overdue-dose alert, and stops actively monitoring the platform for that dispensing cycle, but the medication remains physically present and accessible on the platform. This is a deliberate scope limitation, not an oversight: the collection window (`atraso`), configurable per dose via the application, is intended to represent a clinically meaningful tolerance period of the order of hours in real-world use; short windows (e.g. minutes) were used only for development and bench-scale testing, to allow each test cycle to complete in a practical time. Without the fail-safe hatch, however, the

system has no way to confirm whether a medication marked as `missed` was subsequently retrieved late or never taken at all, which is a genuine safety gap for an unsupervised elderly user and is identified as a priority feature for future development;

- **Biocompatible surface treatment:** 3D-printed parts were not coated; a compliant coating (FDA 21 CFR 175.300, EU Regulation 10/2011) would be required for clinical deployment.

Chapter 5

Future Work and Conclusions

This final chapter closes the dissertation in two parts. Section 5.1 consolidates the priority items for future development identified throughout the mechanical, electronic, and software work described in Chapter 3, and validated, or found wanting, in Chapter 4. Section 5.2 then draws together the engineering contribution of this dissertation as a whole, weighing what was demonstrated against what remains open, and reflecting on the extent to which the work addresses the therapeutic adherence problem that motivated it from the outset.

5.1 Future Work

This section consolidates the priority items for future development, organised into four groups: mechanical, electronic and firmware, software and notifications, and clinical and usability validation. Two items that were previously scoped as future work in earlier stages of this project, the rigid load cell fixation and the three-compartment structural redesign, have since been implemented as described in Section 3.3.9 and Section 3.3.5; their validation is reported in Section 4.5 and Section 4.6, and is summarised, rather than repeated, here.

5.1.1 Mechanical

Redesigning the slot geometry, and reservoir-level agitation. Jamming was observed in every loading trial performed across all three compartments, even with an aggressive firmware-level oscillation routine active (Section 4.6), including for Ben-u-ron 500 mg, whose circular tablets settled unpredictably and jammed earlier than the other two, non-circular units. This suggests that oscillating the cylinder alone is not enough, since it does not disturb the pile of units resting above it in the reservoir. Manually agitating the whole compartment was observed to reliably clear a jam, pointing toward a hybrid mechanism, the current rotating cylinder combined with a small amount of reservoir-level agitation, as a more promising direction than either the cylinder oscillation tried here or the fully vibrating platform of the PICK-SHAKE-DROP concept discarded in Section 3.3.2.2. A compartment or cavity shape tailored specifically to circular tablets is also worth exploring independently of this.

Fail-safe hatch (*alçapão*) mechanism. A servo channel (GPIO 14) is already reserved in the firmware for a fail-safe hatch capable of physically isolating an uncollected dose, but the mechanical design and actuation logic remain unimplemented. As discussed in Section 4.9, without this mechanism the system cannot distinguish a dose collected late from one never collected at all, which is a genuine safety gap for an unsupervised elderly user. This was also explicitly endorsed as a desirable safety feature by both pharmacists interviewed (Section 3.1).

Interchangeable, medication-specific cylinder slots for further expansion. The three compartments implemented in this dissertation (Section 3.3.9) cover the original test medication and two real medications of the same active substance at different dosages. Supporting additional compartments, or medications of substantially different tablet geometry, would require declaring further servo channels at the firmware level (Section 3.4.2.2) and dimensioning each additional cylinder following the same criteria already validated for the present three compartments (Section 3.3.4).

Guided dispensing trajectory. The landing rate measured on the final, enlarged platform was 96% for Ben-u-ron 1000 mg but only 66% to 70% for the two lighter units tested (Section 4.5), attributable to the residual lateral velocity and bounce of a lighter tablet in the current free-fall geometry. A trajectory guided by a channel, a reshaped funnel that reduces drop velocity, or a taller front wall on the collection platform, could reduce this weight-dependent variability and is a candidate improvement independent of the mitigation explored for jamming above.

Biocompatible surface coating. The 3D-printed parts were tested uncoated, for variable isolation during rapid prototyping (Section 3.3.8). Before any deployment in contact with medication intended for ingestion, the internal surfaces would need a biocompatible coating compliant with FDA 21 CFR 175.300 and EU Regulation 10/2011.

Further scaling toward production-scale compartment counts. The three-compartment assembly validated in this dissertation demonstrates that the rotating-cylinder mechanism scales across compartments using a shared collector funnel and a single load cell. Scaling further toward the approximately 10 compartments estimated as necessary for typical polypharmacy regimens (Section 3.1) would require revisiting the collector funnel geometry and the servo/GPIO budget of the microcontroller (Section 3.4.2.2).

5.1.2 Electronics and Firmware

Non-blocking firmware architecture. The dispensing routine currently executes synchronously and occupies the main `loop()` thread for its full duration, including the collection waiting period (Section 3.4.4), regardless of which of the three compartments is dispensing. Converting this to a fully non-blocking, state-machine-based architecture would allow background tasks, such as sensor publication and manual command polling, to continue uninterrupted during a dispensing event, and is a priority item for a production-grade firmware.

Higher-precision environmental sensing. The DHT11's accuracy ($\pm 2^\circ\text{C}$, $\pm 5\%\text{RH}$) is sufficient to validate the end-to-end alert pipeline but would not meet the tighter tolerances required to reliably flag conditions that could compromise medication stability in a production device.

Migrating to the DHT22 ($\pm 0.5^{\circ}\text{C}$, $\pm 2\%\text{RH}$) requires no firmware changes beyond the existing compile-time `TIPO_DHT` flag (Section 3.4.2.4).

Magnetic tamper sensors. A magnetic reed switch (MC38) was originally anticipated on each compartment's lid, to detect unauthorised opening, and was validated independently on the bench but not integrated into the three-compartment prototype (Section 3.4.7). Integrating this sensor into a future multi-compartment device is not expected to present significant technical difficulty.

Visual alert escalation. A five-stage visual escalation, from green to red LEDs, complementing the acoustic alarm, was designed but not implemented, since the current prototype is powered via USB rather than batteries and the additional current draw was not justified for a bench setup (Section 3.4.7). This is revisited together with a battery-powered design below.

Battery-powered operation. The current prototype is powered via USB from a PC. A battery-powered design, with appropriate power budgeting, would be required for a freestanding domestic device and would also enable the visual alert LEDs discussed above.

5.1.3 Software and Notifications

Elderly-facing user interface. At the requirements-elicitation stage (Section 3.1), a second, authentication-free interface was scoped for the elderly user, designed according to inclusive design principles such as enlarged typography, simplified navigation, and no configuration exposure, and intended to present only the next scheduled dose and any active alert. This interface was not implemented or tested within the scope of the present prototype, which instead focused on the caregiver-facing interface and the underlying dispensing, sensing, and notification pipeline. Implementing and validating this profile, including usability testing with representative elderly users, is identified as a priority item for future work, and would directly address FR10 and NFR06 as originally scoped (Section 3.2.3).

Database security rules. The Firebase Realtime Database rules used during development grant unrestricted read and write access (NFR08, Section 4.9), a deliberate simplification adopted to accelerate prototyping. Implementing proper access control, scoped to authenticated caregiver accounts and the device's own service account, is a priority item before any deployment involving real patient data.

Formal notification latency measurement. Push notification latency was observed qualitatively during testing rather than formally timed across repeated trials; a single measured data point, 26 s for the device offline notification (Section 4.7.2), is consistent with the architecture's design but does not substitute for a repeated, formal measurement across all notification types. This remains identified as future work.

Verified Firebase writes with retry on silent failure. As discussed in Section 4.9, the firmware currently checks that the Firebase client reports itself ready before a write, but does not verify that the write itself succeeded, which occasionally allowed writes to `/history`, `/agenda`, and `/toma_atual` to fail silently after long periods of Firebase inactivity. Extending the existing

offline-caching mechanism to also retry writes that report failure, not only writes attempted while genuinely offline, is a direct, low-risk fix identified for future work.

5.1.4 Clinical and Usability Validation

Extended validation with real medication. The speed characterisation and initial drop-detection thresholds established earlier in this dissertation were obtained using 3D-printed reference objects of 0.427 g and 0.6 g, lighter than most commercial tablets; these were repeated with the real Ben-u-ron tablets in the final assembly (Section 4.5). Extending this validation to medications with substantially different tablet geometries remains future work.

Usability and clinical validation with target users. The reliability reported in this dissertation is based on bench-scale, in-vitro testing. Validating the system with elderly users and their caregivers in a realistic home setting, addressing both the usability of the currently unimplemented elderly interface and the longer-term reliability of the dispensing mechanism under everyday use, is identified as essential future work before any clinical or commercial deployment.

5.2 Conclusions

This dissertation set out to address a well-documented and clinically significant problem, namely poor medication adherence among elderly patients living independently, by developing MedBox, a connected automatic medication dispenser combining single-unit dispensing, physical confirmation of both drop and collection, and remote caregiver monitoring and alerting. The work was grounded from the outset in field input from two community pharmacists, whose interviews directly shaped central design decisions, from the per-medication compartmentalisation strategy to the progressive alarm escalation policy and the prioritisation of a fail-safe hatch mechanism.

The core engineering contribution, a cost-effective, FDM-printable, mechanically robust rotating-cylinder singulation mechanism adapted from a two-decade-old industrial principle into an accessible, open implementation, was demonstrated across three compartments in the final assembly, with 100% mechanical rotation success and a servo speed that dispenses each unit quickly and predictably. Landing the dispensed unit inside the collection tray, however, proved to be weight-dependent rather than uniformly reliable: the heavier Ben-u-ron 1000 mg tablet landed successfully in 96% of 50 trials, while the two lighter units tested landed in only 66% to 70% of trials, a gap identified in this dissertation as a direction for future mechanical refinement rather than as a solved problem. The software and data layers built around this mechanism, namely the Firebase-centred, decoupled four-layer architecture, the offline-resilient firmware, the Flutter caregiver application, and the server-side notification watchdog, were validated on the final assembly with real medications dispensed across all three compartments, passing all eighteen functional checks and all seven edge case scenarios tested, including simultaneous connectivity loss, device power loss, and command re-entrancy. This confirmed that the earlier, preliminary bench trial using a coin as a stand-in reference object had been a valid proxy for these software and data layers throughout development.

Two specific mechanical limitations identified during the earlier phases of this dissertation, the load cell's sensitivity to its own mounting rigidity and an orientation-dependent jamming failure mode in the original compartment's dispensing slot, were investigated diagnostically and addressed directly in the final assembly, with mixed results. A rigid, screw-mounted load cell fixation replaced the provisional adhesive-tape mounting and resolved the earlier measurement instability, supporting a reliable gram-based detection threshold for both real medications tested. Jamming, however, was not resolved: even with an aggressive oscillation-based mitigation routine, all three compartments jammed in every loading trial performed, and the circular Ben-u-ron 500 mg tablet jammed earliest of the three, contrary to the initial expectation that its geometry would make it less orientation-dependent. This is reported here as an open problem rather than a solved one, and the observation that agitating the whole compartment, rather than the cylinder alone, reliably cleared a jam is carried forward as a concrete direction for future work (Section 5.1).

This work demonstrates that a reliable, single-unit medication dispensing mechanism can be built from cost-effective, commodity components and an open, FDM-printable design, and that the core singulation principle scales, with only medication-specific resizing of a small number of parameters, from a single compartment to a multi-compartment assembly sharing a common detection sensor. It also demonstrates that a software architecture built around independent, loosely coupled layers can provide meaningful resilience against the connectivity and power failures that characterise real-world, unsupervised deployment. At the same time, this work is honest about what remains unresolved: multi-unit jamming and weight-dependent landing reliability are not yet solved, and their resolution is left as concrete, well-defined future work rather than an incidental gap. In a domain where, as both interviewed pharmacists emphasised, no existing mechanism genuinely supports therapeutic adherence in elderly patients living at home, this work represents a meaningful step toward that goal.

References

- [1] MedReady Inc. *MedReady 1700 Medication Dispenser*. <https://medreadyinc.net/product-category/medication-dispensers/>. Accessed 2 June 2026. 2026.
- [2] LiveFine. *Automatic Pill Dispenser with Bluetooth®*. <https://www.livefineproduct.com/products/automatic-pill-dispenser-with-bluetooth>. Accessed 2 June 2026. 2026.
- [3] MedMinder. *MedMinder Automatic Pill Dispenser for Medication Adherence*. <https://medminder.com>. Accessed 2 June 2026. 2026.
- [4] Evondos Group. *How Automated Medicine Dispensers Work*. <https://www.evondos.com/our-service/how-it-works>. Accessed 2 June 2026. 2026.
- [5] Hero Health. *How does the Hero dispense medications?* <https://support.herohealth.com/en/articles/8058271-how-does-the-hero-dispense-medications>. Hero Help Center, accessed 2 June 2026. 2026.
- [6] Tech-enhanced Life. *MedaCube Medication Dispenser*. <https://www.techenhancedlife.com/reviews/medacube-medication-dispenser>. Accessed 2 June 2026. 2025.
- [7] Fatma Aldila and Ramesh L. Walpola. “Medicine self-administration errors in the older adult population: A systematic review”. In: *Research in Social and Administrative Pharmacy* (2021). DOI: 10.1016/j.sapharm.2021.03.008.
- [8] Elien B. Uitvlugt et al. “Medication-Related Hospital Readmissions Within 30 Days of Discharge: Prevalence, Preventability, Type of Medication Errors and Risk Factors”. In: *Frontiers in Pharmacology* 12 (2021), p. 567424. DOI: 10.3389/fphar.2021.567424.
- [9] Nafiseh Ghassab-Abdollahi et al. “Exploring the reasons for self-administration medication errors among illiterate and low-literate community-dwelling older adults with polypharmacy: a qualitative study”. In: *BMC Geriatrics* 24 (2024), p. 1010. DOI: 10.1186/s12877-024-05595-w.
- [10] Eurostat. *Digital Economy and Society Statistics — Households and Individuals*. https://ec.europa.eu/eurostat/statistics-explained/index.php/Digital_economy_and_society_statistics. Accessed June 2026. 2024.
- [11] Pew Research Center. *Internet/Broadband Fact Sheet: Tech Use Among Older Adults*. <https://www.pewresearch.org/internet/fact-sheet/internet-broadband/>. Accessed June 2026. 2021.
- [12] United Nations Department of Economic and Social Affairs, Population Division. *World Population Prospects 2024: Summary of Results*. Tech. rep. UN DESA/POP/2024/TR/NO. 9. New York: United Nations, 2024.

- [13] Stein Emil Vollset et al. “Fertility, mortality, migration, and population scenarios for 195 countries and territories from 2017 to 2100: a forecasting analysis for the Global Burden of Disease Study”. In: *The Lancet* 396 (2020), pp. 1285–1306. DOI: [10.1016/S0140-6736\(20\)30677-2](https://doi.org/10.1016/S0140-6736(20)30677-2).
- [14] Julie Meyer. *Centenarians: 2020*. 2020 Census Special Reports C2020SR-02. U.S. Census Bureau, 2025.
- [15] Daniele Zaccaria et al. “Social Isolation and Loneliness among Near-Centenarians and Centenarians: Results from the Fordham Centenarian Study”. In: *International Journal of Environmental Research and Public Health* 19.10 (2022), p. 5940. DOI: [10.3390/ijerph19105940](https://doi.org/10.3390/ijerph19105940).
- [16] Saeid Safiri, K. Motlagh Asghari, and Mark J. M. Sullman. “The Global Burden of Diseases and Injuries Among Older Adults”. In: *International Journal of Aging* 1 (2023), e16. DOI: [10.34172/ija.2023.e16](https://doi.org/10.34172/ija.2023.e16).
- [17] Grazia Cicirelli et al. “Ambient Assisted Living: A Review of Technologies, Methodologies and Future Perspectives for Healthy Aging of Population”. In: *Sensors* 21.10 (2021), p. 3549. DOI: [10.3390/s21103549](https://doi.org/10.3390/s21103549).
- [18] Elena Gatt Bonanno et al. “Polypharmacy Prevalence Among Older Adults Based on the Survey of Health, Ageing and Retirement in Europe: An Update”. In: *Journal of Clinical Medicine* 14.1330 (2025). DOI: [10.3390/jcm14041330](https://doi.org/10.3390/jcm14041330).
- [19] Devarashetty Shreya, Peter N. Fish, and Doantrang Du. “Navigating the Future of Elderly Healthcare: A Comprehensive Analysis of Aging Populations and Mortality Trends Using National Inpatient Sample (NIS) Data (2010-2024)”. In: *Cureus* (2025). DOI: [10.7759/cureus.80442](https://doi.org/10.7759/cureus.80442).
- [20] Robert William Hoel, Ryan M. Giddings Connolly, and Paul Y. Takahashi. “Polypharmacy Management in Older Patients”. In: *Mayo Clinic Proceedings* 96.1 (Jan. 2021), pp. 242–256. DOI: [10.1016/j.mayocp.2020.06.012](https://doi.org/10.1016/j.mayocp.2020.06.012).
- [21] Farhad Pazan and Martin Wehling. “Polypharmacy in older adults: a narrative review of definitions, epidemiology and consequences”. In: *European Geriatric Medicine* 12.3 (2021), pp. 443–452. DOI: [10.1007/s41999-021-00479-3](https://doi.org/10.1007/s41999-021-00479-3).
- [22] Mina Khezrian et al. “An overview of prevalence, determinants and health outcomes of polypharmacy”. In: *Therapeutic Advances in Drug Safety* 11 (2020).
- [23] Dennis McDermot. *Health outcomes and costs associated with polypharmacy in Washington state, 2022*. Research brief 117. OFM Health Care Research Center, 2024.
- [24] et al. Omole. “The Impact of Polypharmacy on Health Outcomes in the Aged: A Retrospective Cohort Study”. In: *PLOS ONE* (Feb. 2025). DOI: [10.1371/journal.pone.0317907](https://doi.org/10.1371/journal.pone.0317907). URL: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0317907>.
- [25] Jungmi Chae et al. “The Association Between Continuous Polypharmacy and Hospitalization, Emergency Department Visits, and Death in Older Adults: A Nationwide Large Cohort Study”. In: *Frontiers in Pharmacology* 15 (July 2024), p. 1382990. DOI: [10.3389/fphar.2024.1382990](https://doi.org/10.3389/fphar.2024.1382990).
- [26] Nafiseh Ghassab-Abdollahi et al. “Self-administration medication errors at home and its predictors among illiterate and low-literate community-dwelling older adults with polypharmacy: A negative binomial hierarchical regression”. In: *PLOS ONE* 19.4 (2024), e0302177. DOI: [10.1371/journal.pone.0302177](https://doi.org/10.1371/journal.pone.0302177).

- [27] Fatema A. Alqenae, Douglas Steinke, and Richard N. Keers. “Prevalence and Nature of Medication Errors and Medication-Related Harm Following Discharge from Hospital to Community Settings: A Systematic Review”. In: *Drug Safety* 43 (2020), pp. 517–537. DOI: [10.1007/s40264-020-00918-3](https://doi.org/10.1007/s40264-020-00918-3).
- [28] Rachel Ann Elliott et al. “Economic analysis of the prevalence and clinical burden of medication errors”. In: *BMJ Quality & Safety* (2020). DOI: [10.1136/bmjqs-2019-010206](https://doi.org/10.1136/bmjqs-2019-010206).
- [29] Anusha Natarajan and Bhargavi Kumar. “Prevalence of adverse events due to self-administration errors among patients in a tertiary care hospital: a cross-sectional study utilizing the SAME tool”. In: *International Journal of Research in Medical Sciences* 13.10 (2025), pp. 4277–4283. DOI: [10.18203/2320-6012.ijrms20253178](https://doi.org/10.18203/2320-6012.ijrms20253178).
- [30] Statistics Portugal (INE). *Estimativas de População Residente em Portugal, 2024*. <https://www.ine.pt>. Released June 2025, accessed June 2026. 2025.
- [31] Ana Leite, Tiago Novais, et al. “Prevalence of polypharmacy in the older adult population within primary care in Portugal: a nationwide cross-sectional study”. In: *Archives of Medical Science* (2020). DOI: [10.5114/aoms.2020.92404](https://doi.org/10.5114/aoms.2020.92404).
- [32] Sara Martins et al. “The Prevalence of Polypharmacy and Potentially Inappropriate Medications and Its Relationship with Cognitive Status in Portuguese Institutionalized Older Adults: A Cross-Sectional Study”. In: *Pharmacy* 10.2 (2022), p. 39. DOI: [10.3390/pharmacy10020039](https://doi.org/10.3390/pharmacy10020039).
- [33] Daniel Gomes, Ana Isabel Placido, Rita Mó, et al. “Daily Medication Management and Adherence in the Polymedicated Elderly: A Cross-Sectional Study in Portugal”. In: *International Journal of Environmental Research and Public Health* 17.1 (2020), p. 200. DOI: [10.3390/ijerph17010200](https://doi.org/10.3390/ijerph17010200).
- [34] Ana Isabel Placido et al. “Health Literacy and Medication Adherence Among Low-Income Older Adults in the Inner Center of Portugal: A Questionnaire-Based Study”. In: *Healthcare* (2023). DOI: [10.3390/healthcare11010001](https://doi.org/10.3390/healthcare11010001).
- [35] HLS19 Consortium. *Health literacy in the WHO European Region: investigations by the Action Network on Measuring Population and Organizational Health Literacy (M-POHL)*. Copenhagen: World Health Organization. Regional Office for Europe, 2021.
- [36] Instituto Nacional de Estatística. *Censos 2021 – Resultados Definitivos*. Tech. rep. Lisboa, Portugal: INE, 2022. URL: <https://censos.ine.pt>.
- [37] Aida Isabel Tavares. “Health and life satisfaction factors of Portuguese older adults”. In: *Archives of Gerontology and Geriatrics* 99 (2022), p. 104600. DOI: [10.1016/j.archger.2021.104600](https://doi.org/10.1016/j.archger.2021.104600).
- [38] Ministério da Saúde. *Portaria n.º 263/2023, de 17 de agosto*. Diário da República n.º 159/2023, Série I. Estabelece os termos e as condições para a renovação da dispensa de medicamentos nas farmácias comunitárias. 2023.
- [39] European Commission. *DigComp 2.2: The Digital Competence Framework for Citizens*. <https://publications.jrc.ec.europa.eu/repository/handle/JRC128415>. Accessed June 2026. 2022.
- [40] “The Elderly Digital Divide: Digital Exclusion Versus the Right not To Use the Internet”. In: *International Journal for the Semiotics of Law – Revue Internationale de Sémiotique Juridique* (2025). DOI: [10.1007/s11196-025-10334-4](https://doi.org/10.1007/s11196-025-10334-4).

- [41] Gaby Anne Wildenbos et al. “Mobile Health for Older Adult Patients: Using an Aging Barriers Framework to Classify Usability Problems”. In: *JMIR mHealth and uHealth* 11 (2023), e43186. DOI: [10.2196/43186](https://doi.org/10.2196/43186).
- [42] BrainHQ. *Helping Older Adults Overcome the Challenges of Technology*. BrainHQ News. Aug. 2020. URL: <https://www.brainhq.com/news/latest-news/helping-older-adults-overcome-the-challenges-of-technology/>.
- [43] Hyo Jung Kim et al. “Older Adults’ Technology Anxiety as a Barrier to Digital Inclusion: A Scoping Review”. In: *Educational Gerontology* 49.12 (2023). DOI: [10.1080/03601277.2023.2202080](https://doi.org/10.1080/03601277.2023.2202080).
- [44] Amanda Hunsaker and Eszter Hargittai. “A review of internet use among older adults”. In: *New Media & Society* 20.10 (2018), pp. 3937–3954. DOI: [10.1177/1461444818787348](https://doi.org/10.1177/1461444818787348).
- [45] Koen Saleminck, Dirk Strijker, and Gary Bosworth. “Rural development in the digital age: A systematic literature review on unequal ICT availability, adoption, and use in rural areas”. In: *Journal of Rural Studies* 54 (2017), pp. 360–371. DOI: [10.1016/j.jrurstud.2015.09.001](https://doi.org/10.1016/j.jrurstud.2015.09.001).
- [46] Bettye Rose Connell et al. *The Principles of Universal Design, Version 2.0*. Tech. rep. The Center for Universal Design, North Carolina State University, Apr. 1997. URL: <https://design.ncsu.edu/research/center-for-universal-design/>.
- [47] Amit Lampit, Harry Hallock, and Michael Valenzuela. “Computerized cognitive training in cognitively healthy older adults: a systematic review and meta analysis of effect modifiers”. In: *PLOS Medicine* 11.11 (2014), e1001756. DOI: [10.1371/journal.pmed.1001756](https://doi.org/10.1371/journal.pmed.1001756).
- [48] Falak Masood et al. “A Review on existing IoT Architecture and Communication Protocols used in Healthcare Monitoring System”. In: *Journal of The Institution of Engineers (India): Series B* 103 (2022), pp. 245–257. DOI: [10.1007/s40031-021-00632-3](https://doi.org/10.1007/s40031-021-00632-3).
- [49] S. M. Riazul Islam et al. “The Internet of Things for Health Care: A Comprehensive Survey”. In: *IEEE Access* 3 (2015), pp. 678–708. DOI: [10.1109/ACCESS.2015.2437951](https://doi.org/10.1109/ACCESS.2015.2437951).
- [50] Bahar Farahani et al. “Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare”. In: *Future Generation Computer Systems* 78 (2018), pp. 659–676. DOI: [10.1016/j.future.2017.04.036](https://doi.org/10.1016/j.future.2017.04.036).
- [51] V.K. Quy et al. “An Efficient Edge Computing Management Mechanism for Sustainable Smart Cities”. In: *Sustainable Computing: Informatics and Systems* 37 (2022), p. 100867. DOI: [10.1016/j.suscom.2023.100867](https://doi.org/10.1016/j.suscom.2023.100867).
- [52] Md Motaharul Islam et al. “Edge and Fog Computing in Healthcare – A Review”. In: *Scalable Computing: Practice and Experience* 20.2 (2019), pp. 191–206.
- [53] Google LLC. *Firebase: Build and run your app*. <https://firebase.google.com>. Accessed June 2026. 2023.
- [54] ... Efendi et al. “IoT-Based Elderly Health Monitoring System Using Firebase Cloud Computing”. In: *Health Science Reports* 8 (2025). DOI: [10.1002/hsr2.70498](https://doi.org/10.1002/hsr2.70498).
- [55] Google LLC. *Optimize for Doze and App Standby*. <https://developer.android.com/training/monitoring-device-state/doze-standby>. Accessed June 2026. 2023.
- [56] Sadaf Faisal, Jessica Ivo, and Tejal Patel. “A review of features and characteristics of smart medication adherence products”. In: *Canadian Pharmacists Journal / Revue des Pharmaciens du Canada* 154.5 (2021), pp. 312–323. DOI: [10.1177/17151635211034198](https://doi.org/10.1177/17151635211034198).

- [57] Nichapat Boonnuddar and Pongpisit Wuttidittachotti. “Alarm System for Elder Patients Medication with IoT-Enabled Pill Bottle”. In: *Proceedings of the International Conference on Information and Communication Technology Convergence*. Arduino and load-cell pill-box tracking medicine intake via weight change. 2017.
- [58] Meghla, Refath Tasnia and Deowan, Mahbulul E. and Nuhel, Asif K. and Sazid, Mahin M. and Ekbal, Md. N. and Mahamud, Md. H. “DoseGuard: An IoT-Enabled Smart Drug Dispenser for Improving Medication Adherence and Real-Time Monitoring”. In: *International Journal of Engineering Research and Sustainable Technologies* (2025).
- [59] Dimitrios Karagiannis, Konstantinos Mitsis, and Konstantina S. Nikita. “Development of a Low-Power IoMT Portable Pillbox for Medication Adherence Improvement and Remote Treatment Adjustment”. In: *Sensors* 22.15 (2022), p. 5818. DOI: [10.3390/s22155818](https://doi.org/10.3390/s22155818).
- [60] Maria Cvach. “Monitor Alarm Fatigue: An Integrative Review”. In: *Biomedical Instrumentation & Technology* 46.4 (2012), pp. 268–277. DOI: [10.2345/0899-8205-46.4.268](https://doi.org/10.2345/0899-8205-46.4.268).
- [61] Sue Sendelbach and Marjorie Funk. “Alarm Fatigue: A Patient Safety Concern”. In: *AACN Advanced Critical Care* 24.4 (2013), pp. 378–386. DOI: [10.1097/NCI.0b013e3182a903f9](https://doi.org/10.1097/NCI.0b013e3182a903f9).
- [62] Tejal Patel et al. “A Prospective Study of Usability and Workload of Electronic Medication Adherence Products by Older Adults, Caregivers, and Health Care Providers”. In: *Journal of Medical Internet Research* 22.6 (2020), e18073. DOI: [10.2196/18073](https://doi.org/10.2196/18073).
- [63] Fred D. Davis. “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology”. In: *MIS Quarterly* 13.3 (1989), pp. 319–340. DOI: [10.2307/249008](https://doi.org/10.2307/249008).
- [64] Tanja R. Zijp, Daan J. Touw, and Job F. M. van Boven. “User Acceptability and Technical Robustness Evaluation of a Novel Smart Pill Bottle Prototype Designed to Support Medication Adherence”. In: *Patient Preference and Adherence* 14 (2020), pp. 625–634. DOI: [10.2147/PPA.S240443](https://doi.org/10.2147/PPA.S240443).
- [65] Balvia Ecosystems. *Balvia Ecosystems*. M2PI – Instituto Politécnico de Bragança. Acessado em julho de 2026. 2024. URL: <https://m2pi.ipb.pt/projects/balvia-ecosystems/>.
- [66] Bruno Costa et al. “Medication Dispensing System Architecture”. In: *1st Symposium of Applied Science for Young Researchers*. 2021, pp. 35–39. DOI: [10.34620/sasyr_proceedings_2021](https://doi.org/10.34620/sasyr_proceedings_2021).
- [67] Isaac Van-Deste et al. “Event-Driven Management System for Smart Pill Dispensers”. In: *Computer and Communication Engineering*. Ed. by Ferrante Neri et al. Cham: Springer Nature Switzerland, 2025, pp. 217–228. ISBN: 978-3-031-71079-7. DOI: [10.1007/978-3-031-71079-7_18](https://doi.org/10.1007/978-3-031-71079-7_18).
- [68] Manuel António Cruz Patrício. “A Smartwatch-Based Intervention for Reducing Medication Non-Adherence: A Digital Health Approach”. Master in Informatics; supervisor: Rui Pedro Lopes. MA thesis. Instituto Politécnico de Bragança, School of Technology and Management, 2025. URL: <https://bibliotecadigital.ipb.pt/>.
- [69] Yúri dos Ramos Gué. “Aplicação móvel Balvia para dispensador de medicamento”. Orientador: José Eduardo Fernandes. MA thesis. Instituto Politécnico de Bragança, Escola Superior de Tecnologia e Gestão, 2025. URL: <http://hdl.handle.net/10198/35136>.
- [70] Liliana Batista Vieira et al. “The use of an electronic medication organizer device with alarm to improve medication adherence of older adults with hypertension”. In: *Einstein (São Paulo)* 19 (2021), eAO6011. DOI: [10.31744/einstein_journal/2021AO6011](https://doi.org/10.31744/einstein_journal/2021AO6011).

- [71] Farmácia de Lomar. *Organizador semanal de medicação*. https://bo.farmaciadelomar.pt/fileuploads/uploads/mceclip0_1706703676193.jpg. Acedido em 11 de junho de 2026. 2024.
- [72] McKesson Automation Systems. *Variable Size Pill Dispenser*. Canadian Patent CA2417103C. Espacenet. 2003.
- [73] William R. Ketterhagen and Bruno C. Hancock. “Optimizing the design of eccentric feed hoppers for tablet presses using DEM”. In: *Computers & Chemical Engineering* 34.7 (2010), pp. 1072–1081. DOI: [10.1016/j.compchemeng.2010.03.003](https://doi.org/10.1016/j.compchemeng.2010.03.003).
- [74] Aosong (Guangzhou) Electronics Co., Ltd. *DHT11 Digital Temperature and Humidity Sensor: Product Manual*. Accessed 14 June 2026. 2018. URL: https://akizukidenshi.com/goodsaffix/DHT11_20180119.pdf.
- [75] Aosong (Guangzhou) Electronics Co., Ltd. *DHT22 (AM2302) Digital-Output Relative Humidity and Temperature Sensor: Product Manual*. Accessed 14 June 2026. 2015. URL: <https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT22.pdf>.
- [76] Autoridade Nacional do Medicamento e Produtos de Saúde (Infarmed). *Listas de Preços Aprovadas (Approved Price Lists)*. <https://www.infarmed.pt/web/infarmed/servicos-on-line/pesquisa-do-medicamento>. Publicly available monthly document, accessed April 2026. 2026.

Appendix A

Detailed Use Case Descriptions

This appendix provides the detailed description of the two use cases most directly relevant to the system's core dispensing and safety logic — UC6 (Dispense Medication) and UC4 (Request Anticipated Dispensing) — referenced from Section 3.2.2. The remaining use cases (UC1, UC2, UC3, UC5) are summarised in Section 3.2.2 and are not detailed further here, as their flows are straightforward on the `/agenda`, `/inventory`, and `/alertas` nodes already described in Sections 3.5.1 and 3.5.4.

Table A.1: Detailed description of UC6: Dispense Medication.

Use Case	UC6: Dispense Medication
Actor	System (ESP32-C6 firmware)
Related requirements	FR02, FR03, FR04, FR05, NFR01, NFR05
Preconditions	A dose in the local schedule cache has <code>status = pending</code> and its scheduled time matches the current time (or the dose was triggered by UC4).
Main flow	1. The firmware detects that the current time matches a pending dose for a given compartment. 2. The corresponding servo motor rotates the dispensing cylinder to release one unit. 3. The load cell measures the platform weight before and after rotation to confirm the drop. 4. The firmware writes the dispensing outcome to <code>/toma_atual (estado, hora_dispensa, confirmacao_queda)</code> and updates <code>/agenda (dispensada = true)</code>. 5. The firmware waits for the user to collect the medication, monitored via the load cell. 6. Upon collection, the firmware writes the final outcome to <code>/history</code> and <code>/toma_atual (confirmacao_recolha, hora_recolha, estado = taken)</code>.
Postconditions	The dose is recorded in <code>/history</code> with one of the outcomes <code>taken</code> , <code>failure</code> , or <code>device_offline</code> ; the corresponding <code>/agenda</code> entry has <code>dispensada = true</code> .
Alternative flows	• Mechanical failure: if the load cell does not detect a weight increase after rotation, <code>estado</code> is set to <code>falha_mecanica</code> and the corresponding <code>/alertas</code> flag may be raised. • Not collected within the tolerance window: the firmware activates the progressive acoustic alarm (1, 2, or 3 beeps) and sets <code>/toma_atual/alerta_ativo = true</code> (and, after the correction described in Section 3.4.4, also <code>/alertas/dose_em_atraso = true</code>). If the window expires without collection, <code>estado</code> is set to <code>nao_recolhido</code>. • No Internet connection: the dispensing routine proceeds normally based on the local cache and the RTC fallback (NFR01); the result is stored locally and synchronised to the RTDB once connectivity is restored, with <code>device_offline</code> recorded if the original write attempt failed. • Re-entrancy guard: if the dispensing event for the current minute has already been executed (tracked via <code>ultimaHoraDispensada</code>), the routine does not execute again, preventing duplicate dispensing after a reconnection (NFR05).

Table A.2: Detailed description of UC4: Request Anticipated Dispensing.

Use Case	UC4: Request Anticipated Dispensing
Actor	Caregiver
Related requirements	FR08 (triggers UC6, see Table A.1)
Preconditions	The caregiver is authenticated; a dose for the target compartment is present in /agenda for the current day, with status = "pending".
Main flow	1. The caregiver selects a specific dose already scheduled for the current day in the agenda and requests that it be dispensed immediately rather than at its scheduled time. 2. The app writes /comando_manual with <code>executar = true</code> and <code>gavetas</code> set to the compartment associated with that dose. 3. In the same operation, the app updates the corresponding /agenda entry, setting <code>dispensada = true</code> and <code>status = "taken"</code>. 4. The ESP32-C6 firmware polls /comando_manual and detects the pending command. 5. The firmware executes the dispensing routine (UC6, Table A.1) for the specified compartment. 6. The event is logged in /history with <code>scheduled_time = "On-Demand"</code>. 7. The firmware resets /comando_manual/executar to false and writes <code>status_final = "sucesso"</code>.
Postconditions	A new entry is recorded in /history with <code>scheduled_time = "On-Demand"</code> . The original /agenda entry for that dose now has <code>status = "taken"</code> , so when the firmware later reaches the originally scheduled time, it finds <code>status ≠ "pending"</code> and does not dispense the dose again (FR08). Duplication is therefore prevented through this app-side bookkeeping rather than by firmware-side deduplication.
Alternative flows	• Device offline: the command remains pending in /comando_manual until the ESP32-C6 reconnects and polls it, at which point the dispensing routine is executed and logged as described above. The /agenda entry, however, has already been marked as taken by the app regardless of device connectivity.

Appendix B

System Requirements

This appendix lists the full functional and non-functional requirements considered for the MedBox prototype, summarised in Section 3.2.3.

Table B.1: Functional requirements of the MedBox system.

ID	Description
FR01	The system shall allow the caregiver to define a weekly schedule of doses per compartment, with configurable time, quantity, and tolerance window.
FR02	The system shall automatically dispense the medication at the scheduled time, actuating the actuator corresponding to the associated compartment.
FR03	The system shall physically confirm the drop of the medication via a load cell before marking the dose as dispensed.
FR04	The system shall wait for the medication to be collected by the user within a configurable time window, issuing a progressive acoustic alert in case of delay.
FR05	The system shall log the outcome of each dispensing event (success, mechanical failure, not collected, device offline) in a queryable history.
FR06	The system shall continuously monitor the temperature and humidity inside the dispenser, comparing them against thresholds configurable by the caregiver.
FR07	The system shall notify the caregiver, via push notification, whenever an anomalous condition occurs (uncollected dose, temperature/humidity outside limits, or loss of communication with the device).
FR08	The system shall allow the caregiver to request an on-demand anticipated dispensing, outside the scheduled time, without duplicating the dose already planned for that day.
FR09	The system shall allow the caregiver to associate medications from the Infomed catalogue with each physical compartment, through search by name or active ingredient.
FR10 ¹	The interface intended for the elderly person shall not require authentication and shall present only the next scheduled dose and active alerts.

Table B.2: Non-functional requirements of the MedBox system.

ID	Description
NFR01	Offline resilience: the dispenser shall continue to execute the dispensing schedule even in the absence of an Internet connection, relying on a local cache and a real-time clock (RTC) as a fallback mechanism.
NFR02	Latency: the system shall reflect state changes (e.g. a new scheduled dose, an environmental alert) in the user interface within less than 5 seconds, leveraging the reactive (streaming) nature of the Firebase RTDB.
NFR03	Mobile client energy efficiency: the detection and triggering of alerts shall not depend on continuous processing on the smartphone, so as not to compromise battery life nor violate the operating system's power management policies (Doze Mode).
NFR04	Alert fatigue prevention: the system shall only notify the caregiver upon anomalous conditions, avoiding redundant notifications for normal events (e.g. a successful dose), in order to preserve the clinical relevance of the alerts received.
NFR05	Re-entrancy fault tolerance: the system shall prevent duplication of dispensing events in scenarios of connectivity loss and recovery within the same time interval.
NFR06 ²	Profile-differentiated usability: the elderly user interface shall follow inclusive design principles (enlarged typography, simplified navigation, absence of authentication barriers).
NFR07	Medication database scalability: searching the medication catalogue (over 6000 entries) shall be efficient in terms of latency and mobile data consumption.
NFR08	Data security: access to the patient's medication schedule, history, and health-related data shall be protected against unauthorised reading and writing.

¹FR10, originally scoped for the elderly-facing interface, is deferred to future work — see Section 5.1.

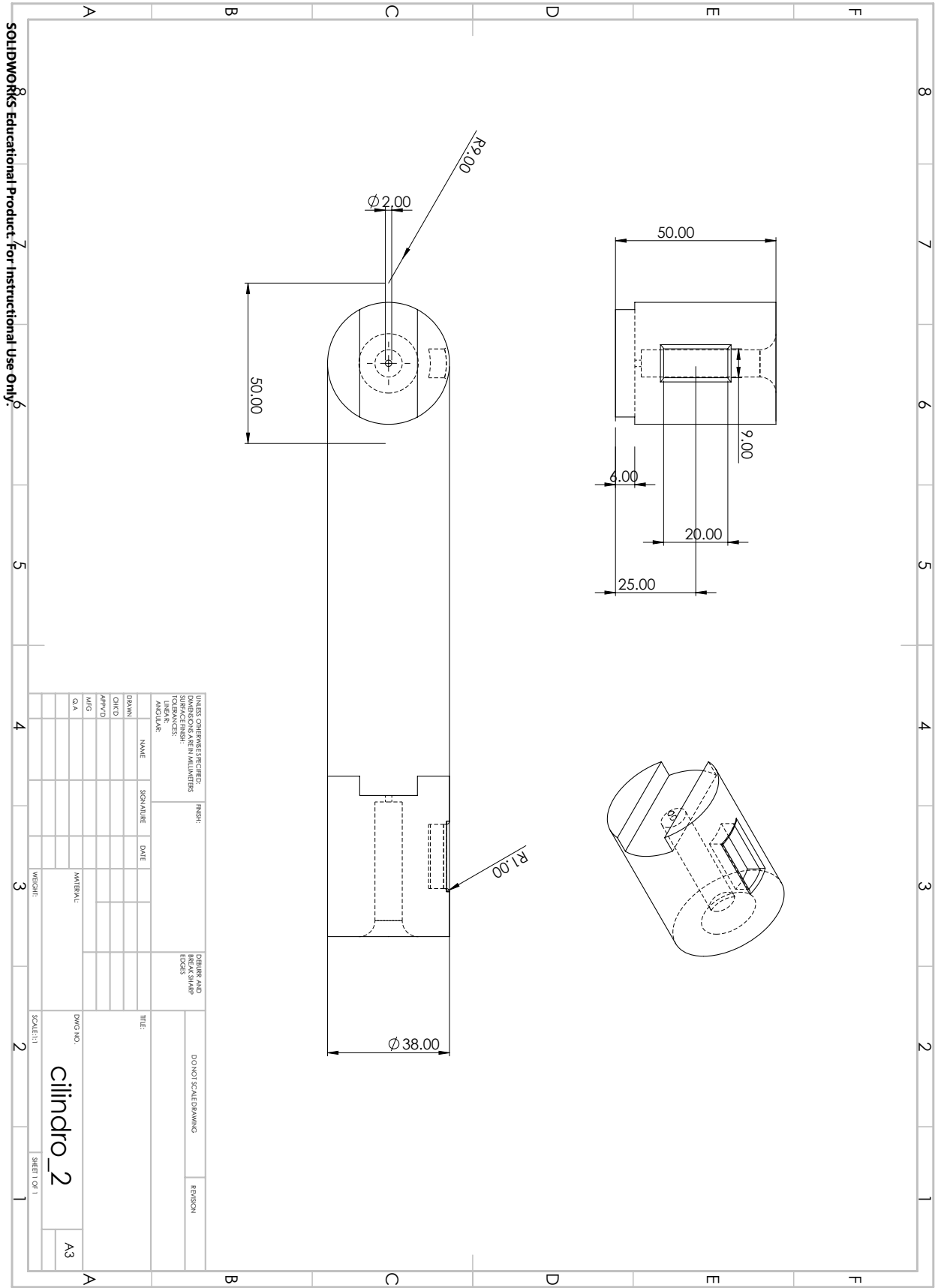
²NFR06, originally scoped for the elderly-facing interface, is deferred to future work — see Section 5.1.

Appendix C

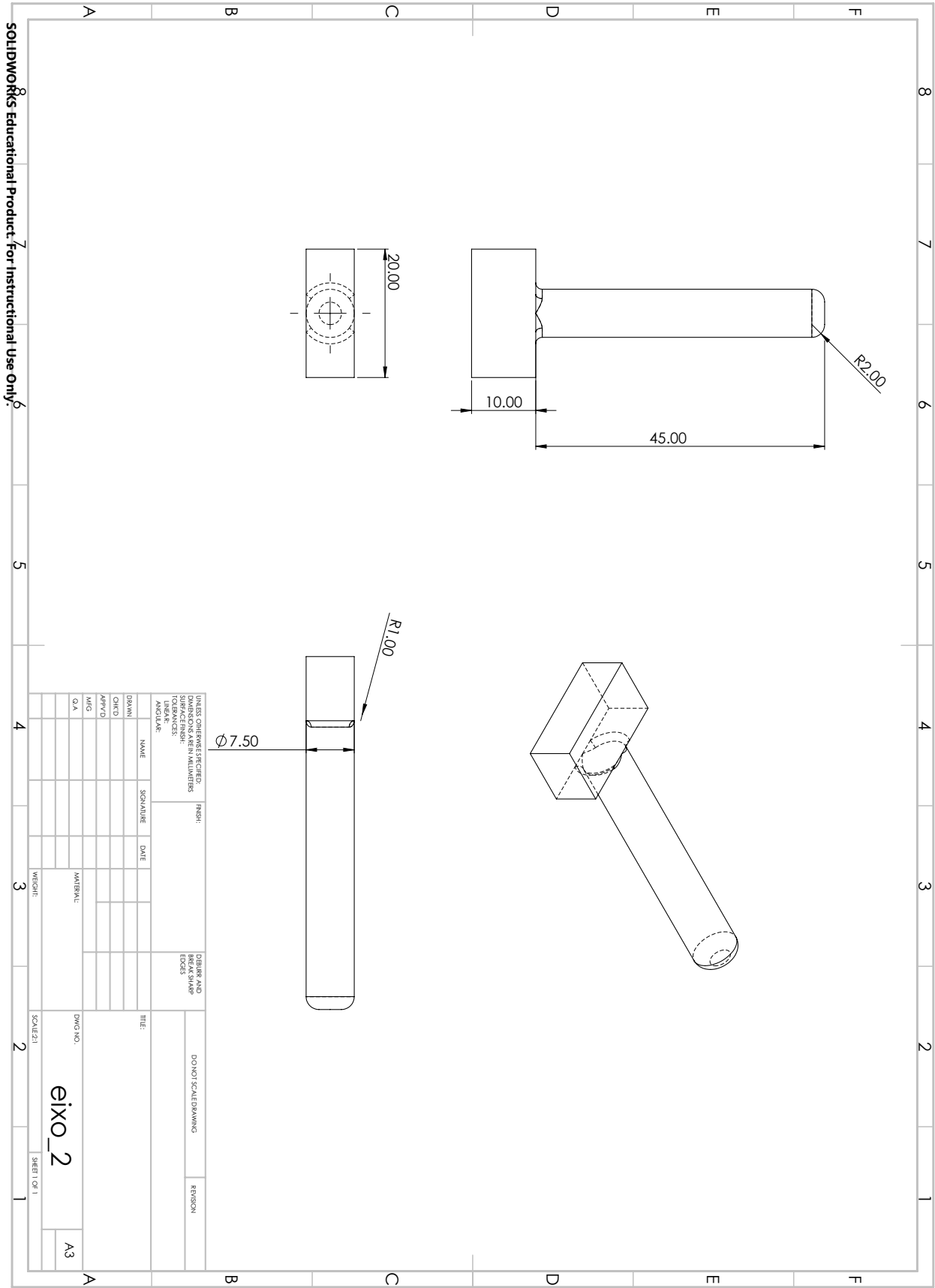
SolidWorks Technical Drawings

This appendix presents the technical drawings, generated from the SolidWorks models described in Section 3.3.5, for the parts that compose the dispensing mechanism: the rotating cylinder, the shaft, the body with the embedded asymmetric funnel, the cap, the upper compartment support, the main tunnel, the secondary collector funnel, the lower support that holds the load cell, and the load cell platform. Each drawing includes the part's principal views (front, top, right, and isometric) and the dimensions referenced throughout Chapter 3, in particular the fixed versus medication-dependent parameters summarised in Table 3.2 (Section 3.3.4).

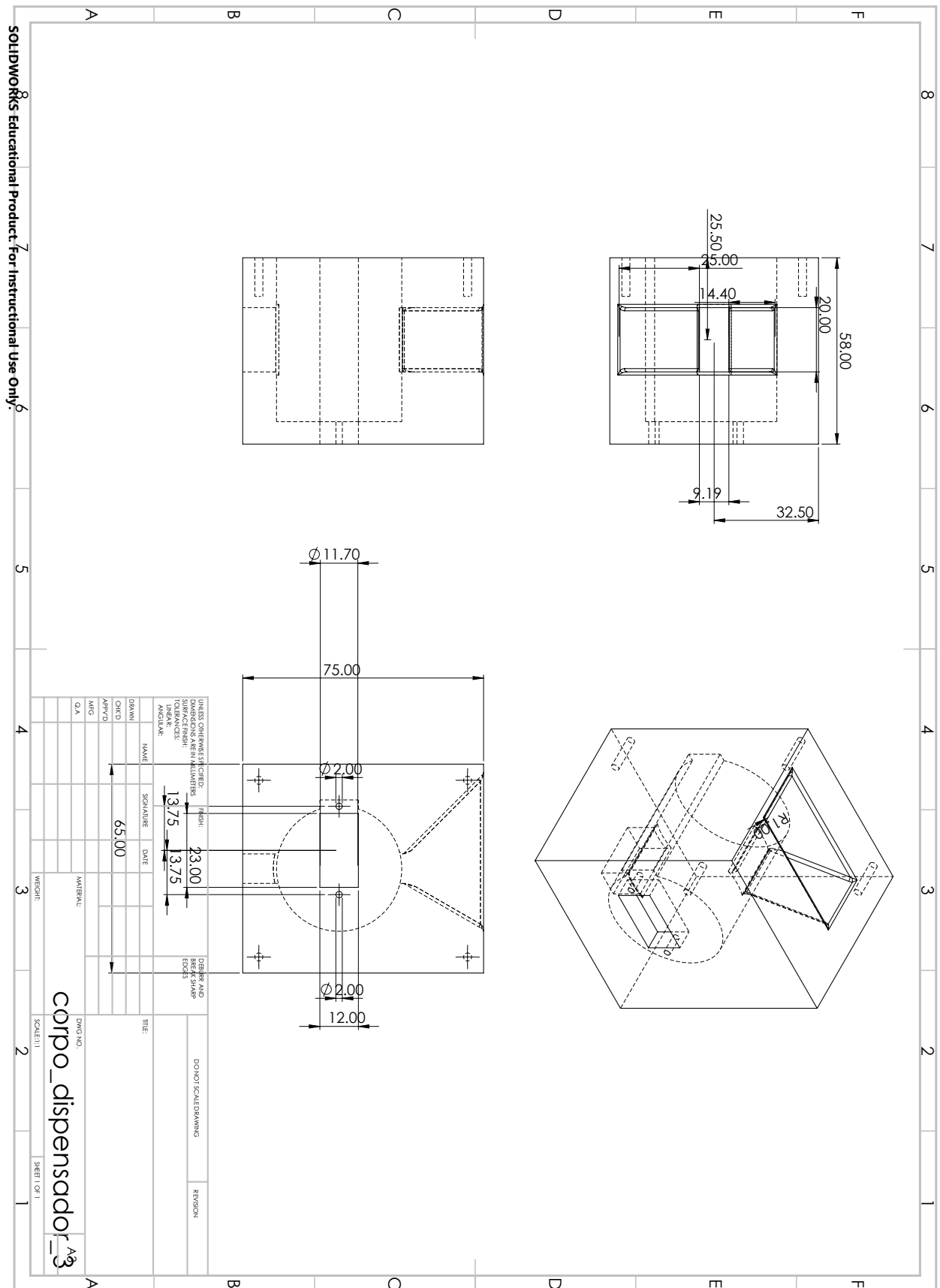
C.1 Rotating Cylinder



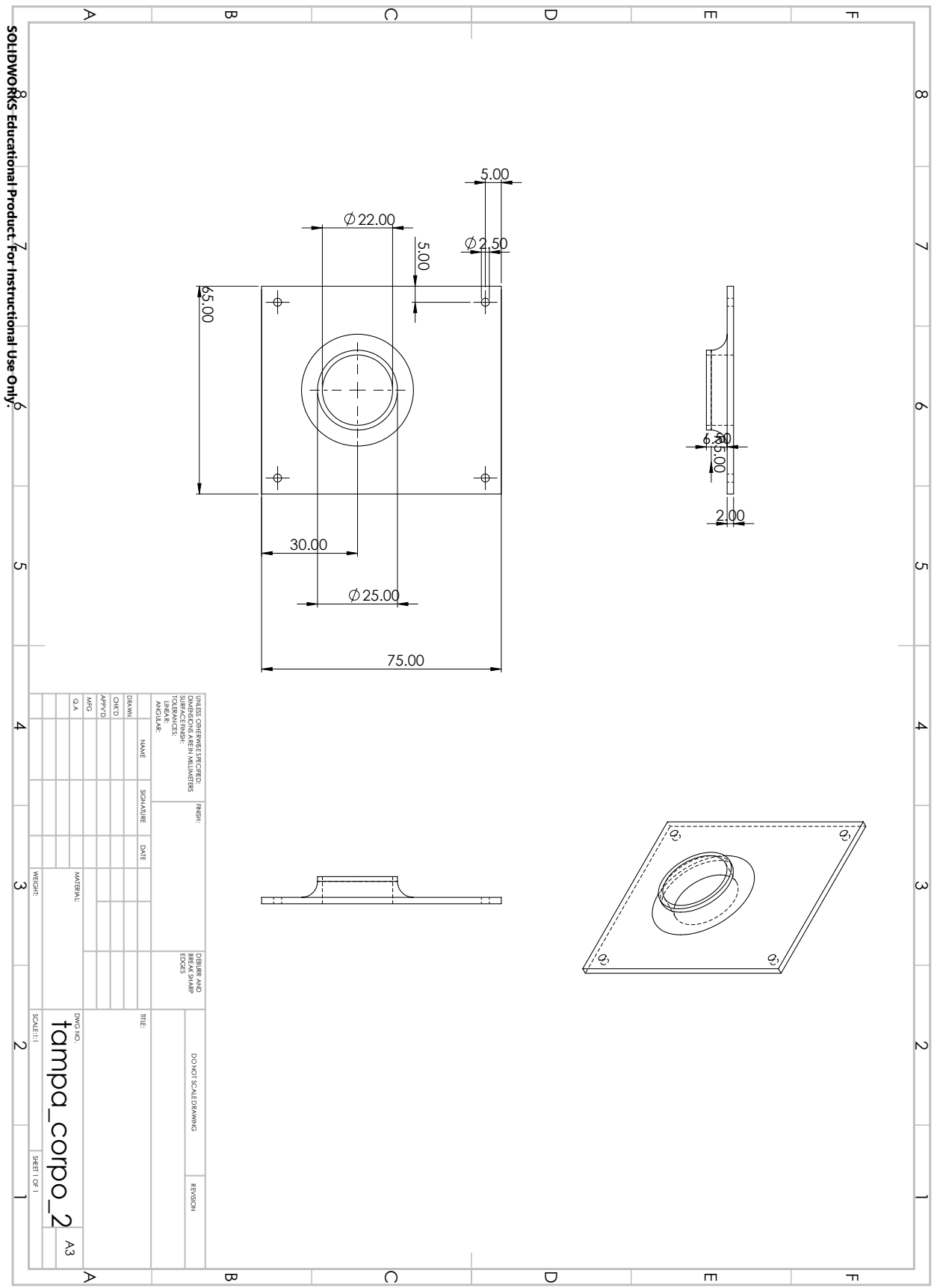
C.2 Shaft



C.3 Body



C.4 Cap

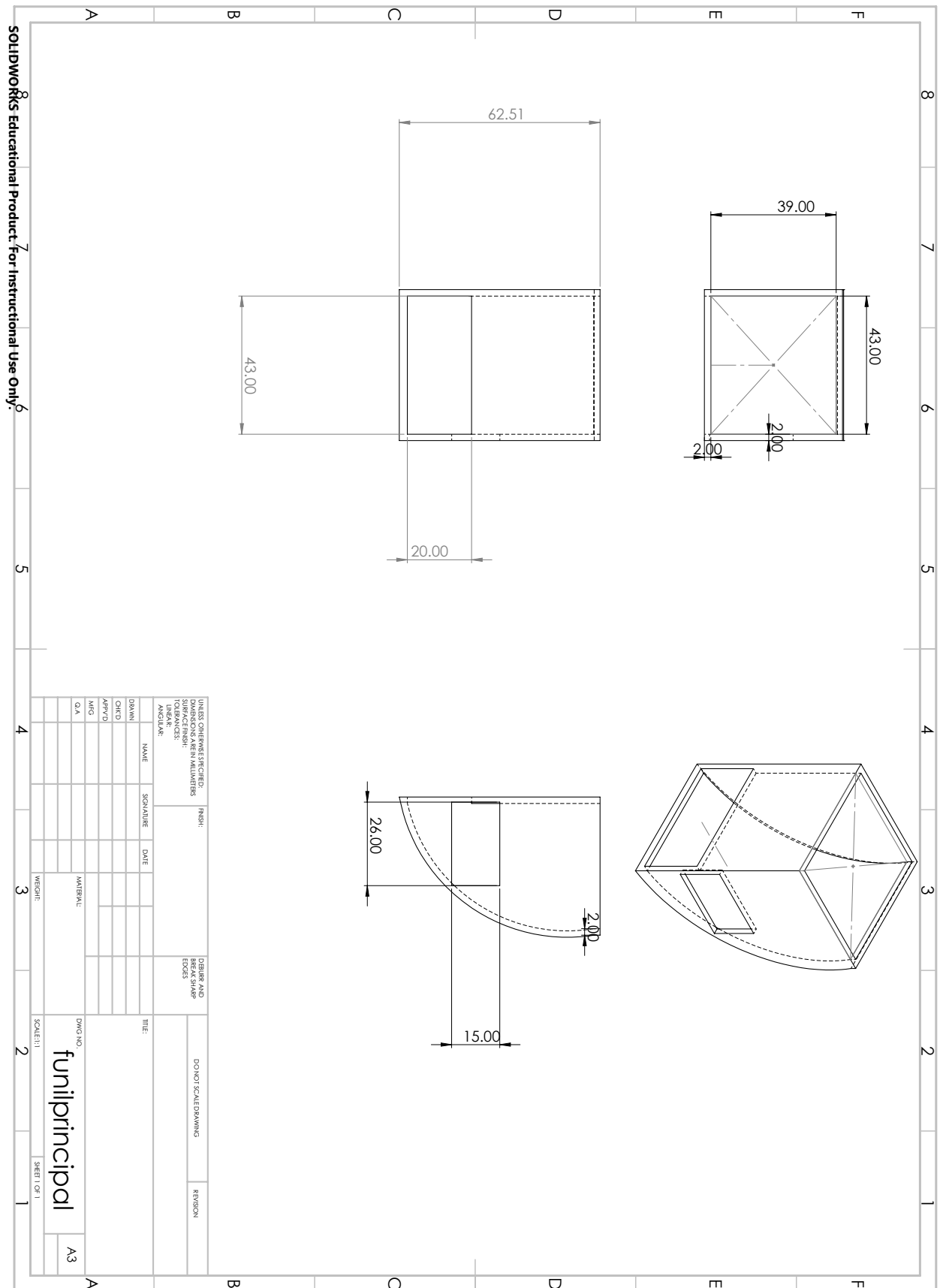


C.5 Upper Compartment Support

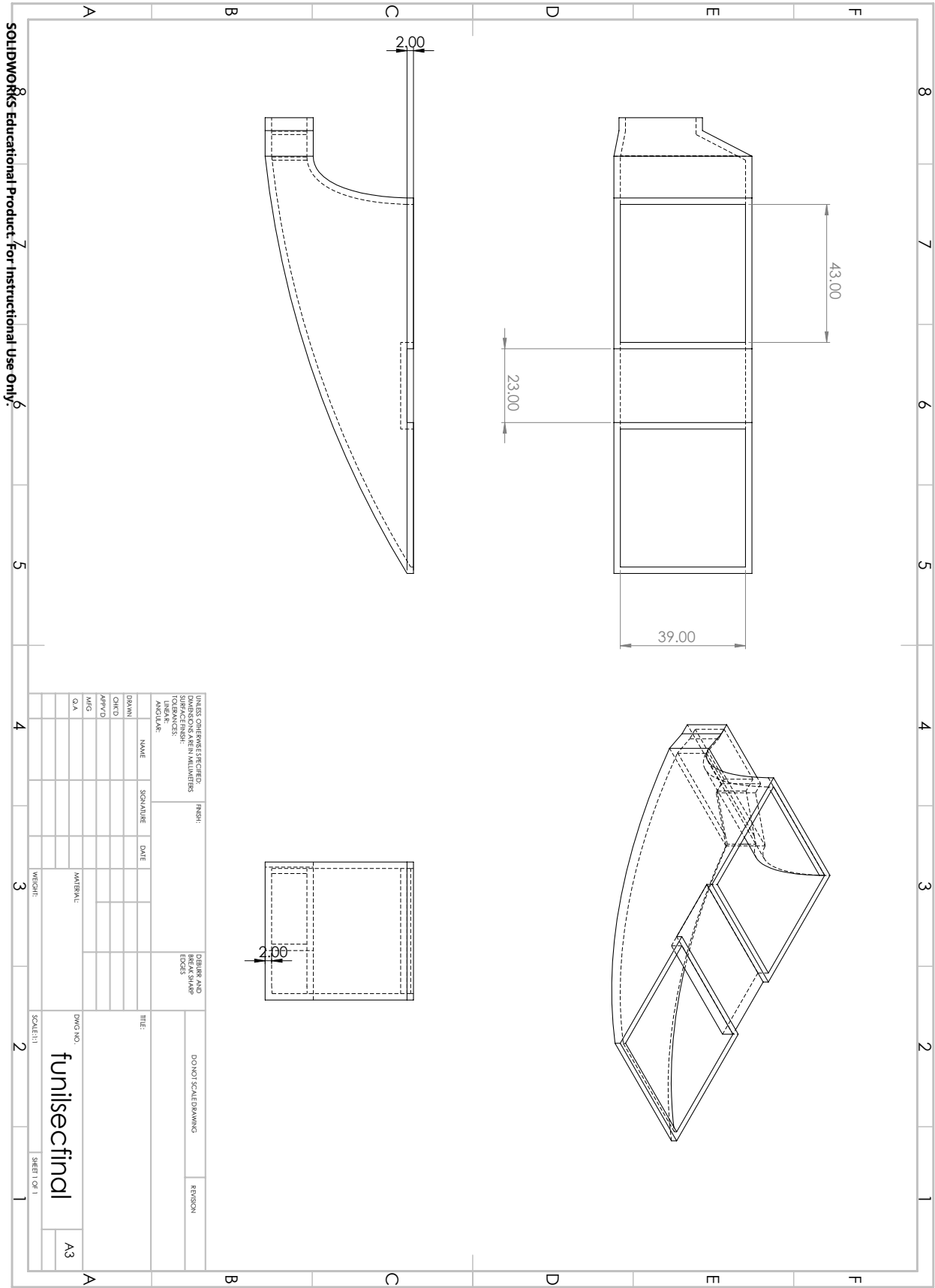
This piece is identical across all three compartments; a single drawing is therefore provided.



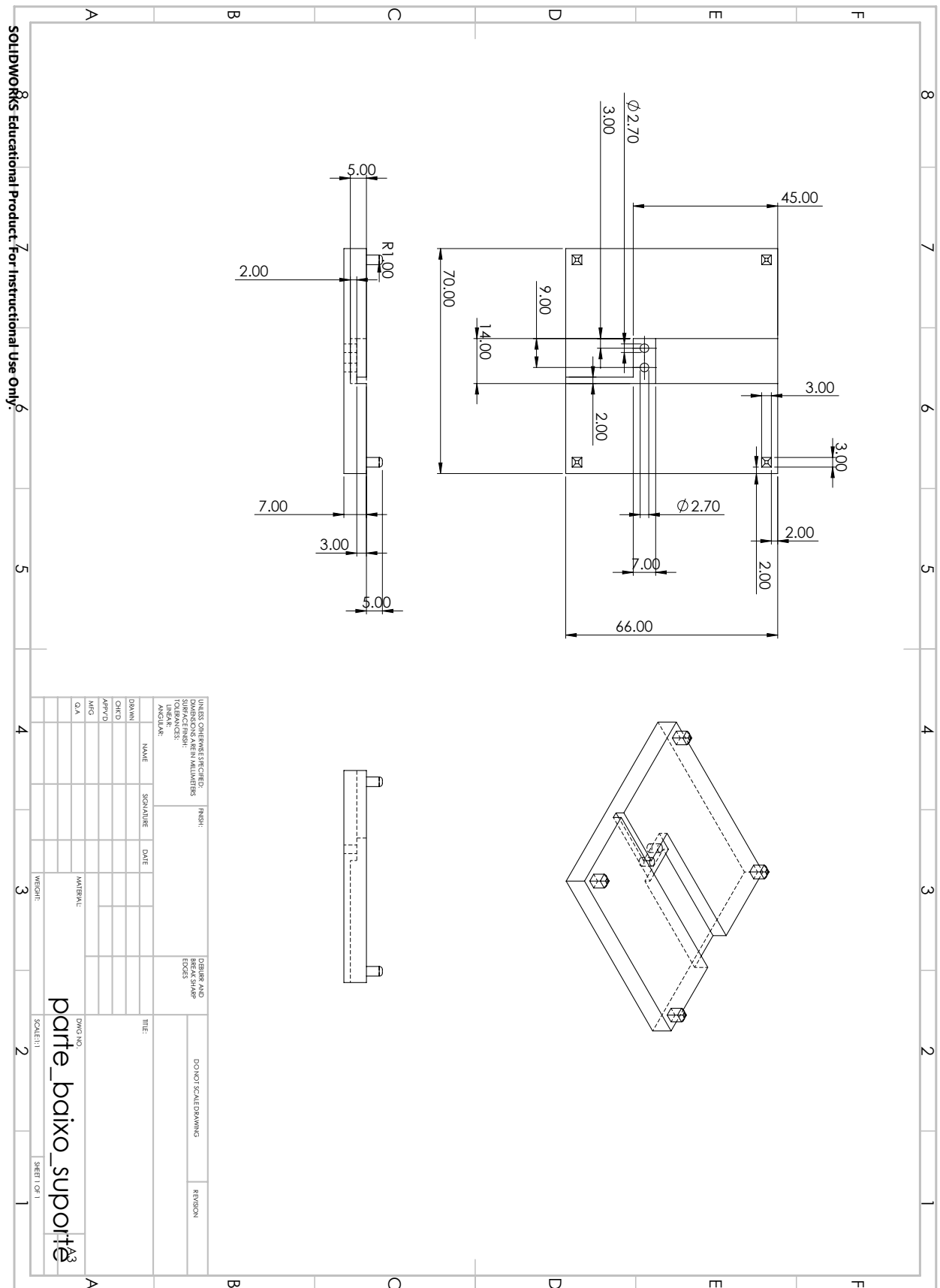
C.6 Main Tunnel



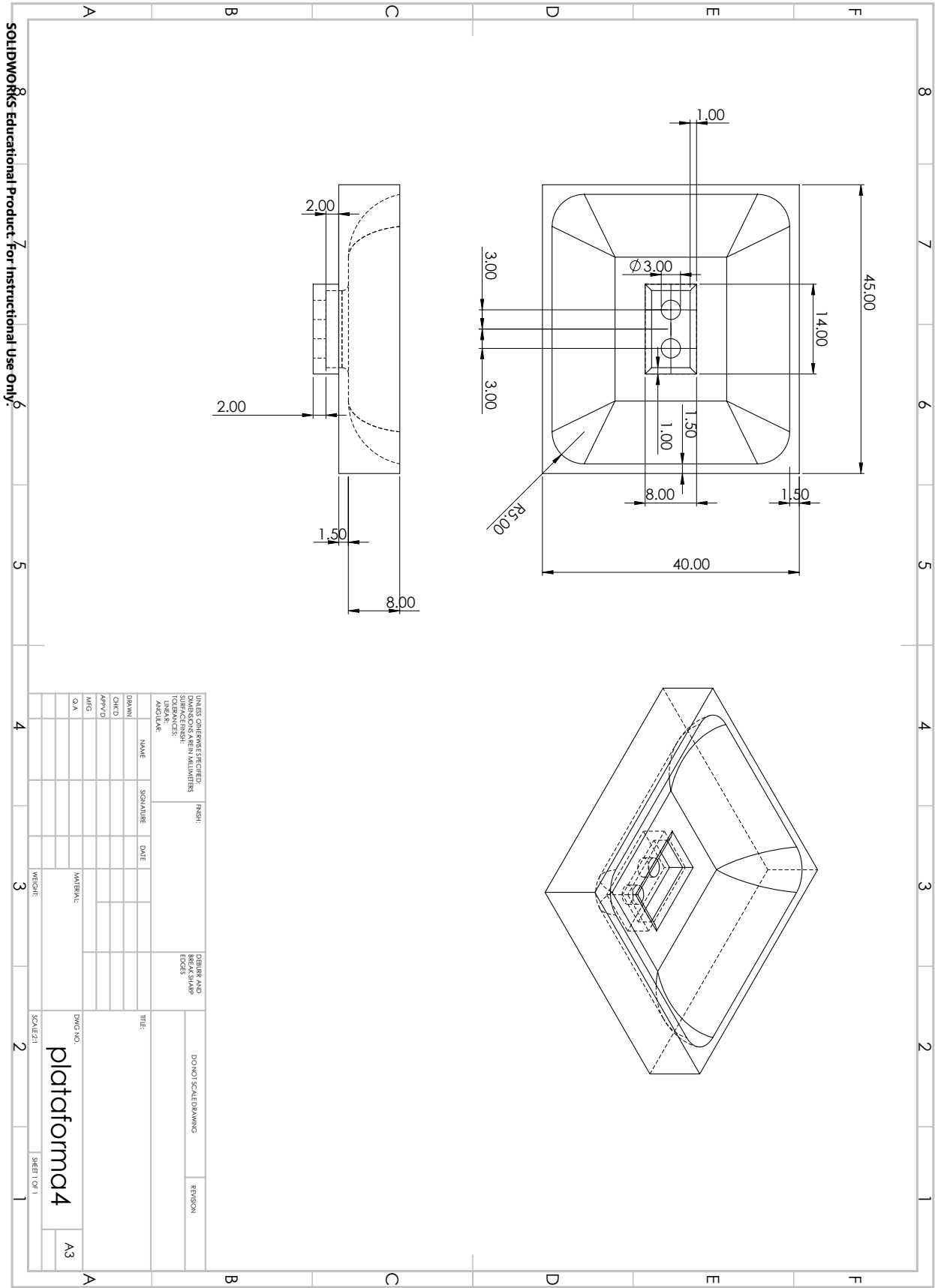
C.7 Secondary Collector Funnel



C.8 Lower Support and Load Cell Mounting



C.9 Load Cell Platform



Appendix D

Firmware Code and Test Logs

This appendix contains the firmware sketches used for the individual mechanical characterisation tests described in Chapter 4, the final production firmware, and the raw Serial Monitor output referenced as supporting evidence for the illustrative examples discussed in Section 4.7.2.

D.1 Load Cell Calibration Sketch

Firmware sketch used to obtain CALIBRATION_FACTOR from the 1 euro coin and to validate the resulting gram reading against a real medication unit, as described in Section 4.4.

```
#include <HX711.h>

// --- Load Cell ---
#define PINO_DOUT 22
#define PINO_SCK 21

// Peso conhecido usado para calibrar (moeda de 1 euro = 7.5 g oficial)
#define PESO_CONHECIDO_G 7.5f
#define NUM_LEITURAS 10 // leituras raw por medicao (camada 1)
#define REPETICOES_CALIB 3 // repeticoes da calibracao (camada 2)
#define REPETICOES_MED 5 // repeticoes da medicao do medicamento

float CALIBRATION_FACTOR = 1.0f; // atualizado depois da calibracao

HX711 scale;

void mostrarMenu() {
  Serial.println("\n===== MENU =====");
  Serial.println("t - Tare simples (zerar, SEM nada na plataforma)");
  Serial.println("c - Calibrar (3 repeticoes com a moeda de 1 euro)");
  Serial.println("m - Medir medicamento (5 repeticoes)");
  Serial.println("r - Ler valor bruto (raw) atual, sem tare");
  Serial.println("p - Ler peso atual (com o factor definido)");
  Serial.println("=====");
}
```

```
}

void esperarEnter(const char* mensagem) {
    Serial.println(mensagem);
    while (Serial.available() == 0) {
        delay(50);
    }
    while (Serial.available() > 0) Serial.read();
}

long lerMediaRawComPrint(int n) {
    long soma = 0;
    int count = 0;

    Serial.println("Leituras individuais (raw):");
    for (int i = 0; i < n; i++) {
        while (!scale.is_ready()) {
            delay(5);
        }
        long valor = scale.get_value(1);
        Serial.printf("  #02d -> %ld\n", i + 1, valor);
        soma += valor;
        count++;
        delay(100);
    }

    long media = soma / count;
    Serial.printf("Media de %d leituras (raw): %ld\n", count, media);
    return media;
}

float lerMediaPesoComPrint(int n) {
    float soma = 0;
    int count = 0;

    Serial.println("Leituras individuais (peso, g):");
    for (int i = 0; i < n; i++) {
        while (!scale.is_ready()) {
            delay(5);
        }
        float valor = scale.get_units(1);
        Serial.printf("  #02d -> %.3f g\n", i + 1, valor);
        soma += valor;
        count++;
        delay(100);
    }

    float media = soma / count;
    Serial.printf("Media de %d leituras (peso): %.3f g\n", count, media);
}
```



```

    return media;
}

void calibrarComMoeda() {
    Serial.println("\n===== CALIBRACAO (3 repeticoes) =====");

    long    rows[REPETICOES_CALIB];
    double somaRaw = 0;

    for (int rep = 0; rep < REPETICOES_CALIB; rep++) {
        Serial.printf("\n--- Repeticao %d de %d ---\n", rep + 1, REPETICOES_CALIB);

        esperarEnter("Tira a moeda da plataforma e envia qualquer tecla para tarar.");
        Serial.println("A tarar... nao toques na plataforma.");
        delay(1000);
        scale.tare(20);
        Serial.println("[OK] Tare feito.");

        esperarEnter("Agora coloca a moeda de 1 euro na plataforma e envia qualquer tecla.");
        delay(1000);

        long media = lerMediaRawComPrint(NUM_LEITURAS);
        rows[rep] = media;
        somaRaw += media;

        Serial.printf(">>> Raw da repeticao %d: %ld\n", rep + 1, media);
    }

    long rawMedio = (long)(somaRaw / REPETICOES_CALIB);

    long minRaw = rows[0], maxRaw = rows[0];
    for (int i = 1; i < REPETICOES_CALIB; i++) {
        if (rows[i] < minRaw) minRaw = rows[i];
        if (rows[i] > maxRaw) maxRaw = rows[i];
    }
    float desvioPct = (rawMedio != 0) ? (100.0f * (maxRaw - minRaw) / (float)rawMedio) : 0;

    if (rawMedio == 0) {
        Serial.println("[ERRO] Raw medio final igual a 0. Repete a calibracao.");
        return;
    }

    CALIBRATION_FACTOR = (float)rawMedio / PESO_CONHECIDO_G;
    scale.set_scale(CALIBRATION_FACTOR);

    Serial.println("\n===== RESULTADO FINAL =====");
    Serial.printf("Raws individuais      : %ld, %ld, %ld\n", rows[0], rows[1], rows[2]);
    Serial.printf("Raw medio final          : %ld\n", rawMedio);
    Serial.printf("Variacao min-max        : %.2f %%\n", desvioPct);

```

```

Serial.printf("Peso conhecido usado    : %.3f g\n", PESO_CONHECIDO_G);
Serial.printf("CALIBRATION_FACTOR      : %.4f\n", CALIBRATION_FACTOR);
Serial.println("=====");

if (desvioPct > 2.0f) {
    Serial.println("[AVISO] Variacao entre repeticoes > 2%. Verifica posicionamento da moeda ou e
}
}

void medirMedicamento() {
    Serial.println("\n===== MEDICAO MEDICAMENTO (5 repeticoes) =====");

    if (CALIBRATION_FACTOR == 1.0f) {
        Serial.println("[AVISO] Ainda nao calibraste (factor = 1.0). Os pesos nao vao ser fiaveis.");
    }

    long  raws[REPETICOES_MED];
    float pesos[REPETICOES_MED];
    double somaRaw = 0;
    double somaPeso = 0;

    for (int rep = 0; rep < REPETICOES_MED; rep++) {
        Serial.printf("\n--- Repeticao %d de %d ---\n", rep + 1, REPETICOES_MED);

        esperarEnter("Tira o medicamento da plataforma e envia qualquer tecla para tarar.");
        Serial.println("A tarar... nao toques na plataforma.");
        delay(1000);
        scale.tare(20);
        Serial.println("[OK] Tare feito.");

        esperarEnter("Coloca o medicamento na plataforma e envia qualquer tecla.");
        delay(1000);

        long  mediaRaw  = lerMediaRawComPrint(NUM_LEITURAS);
        float mediaPeso = (float)mediaRaw / CALIBRATION_FACTOR;

        raws[rep]  = mediaRaw;
        pesos[rep] = mediaPeso;
        somaRaw  += mediaRaw;
        somaPeso += mediaPeso;

        Serial.printf(">>> Repeticao %d -> raw: %ld | peso: %.3f g\n", rep + 1, mediaRaw, mediaPeso);
    }

    long  rawMedio  = (long)(somaRaw / REPETICOES_MED);
    float pesoMedio = (float)(somaPeso / REPETICOES_MED);

    double somaQuad = 0;
    for (int i = 0; i < REPETICOES_MED; i++) {

```

```

    double diff = pesos[i] - pesoMedio;
    somaQuad += diff * diff;
}
float desvioPadrao = sqrt(somaQuad / REPETICOES_MED);

Serial.println("\n===== RESULTADO FINAL =====");
Serial.print("Raws individuais      : ");
for (int i = 0; i < REPETICOES_MED; i++) Serial.printf("%ld ", raws[i]);
Serial.println();
Serial.print("Pesos individuais (g)   : ");
for (int i = 0; i < REPETICOES_MED; i++) Serial.printf("%.3f ", pesos[i]);
Serial.println();
Serial.printf("Raw medio final           : %ld\n", rawMedio);
Serial.printf("Peso medio final           : %.3f g\n", pesoMedio);
Serial.printf("Desvio padrao (peso)       : %.3f g\n", desvioPadrao);
Serial.println("=====");
}

void setup() {
    Serial.begin(115200);
    delay(1000);

    scale.begin(PINO_DOUT, PINO_SCK);
    delay(500);

    if (!scale.is_ready()) {
        Serial.println("[ERRO] HX711 nao detectado! Verifica ligacoes.");
        while (true);
    }

    scale.set_scale(1.0f);

    Serial.println("\n[INFO] HX711 pronto.");
    mostrarMenu();
}

void loop() {
    if (Serial.available() > 0) {
        char c = Serial.read();

        switch (c) {
            case 't':
            case 'T': {
                Serial.println("\nA fazer tare... nao toques na plataforma.");
                delay(1000);
                scale.tare(20);
                Serial.println("[OK] Tare feito.");
                break;
            }
        }
    }
}

```

```

    case 'c':
    case 'C':
        calibrarComMoeda();
        break;
    case 'm':
    case 'M':
        medirMedicamento();
        break;
    case 'r':
    case 'R': {
        long raw = lerMediaRawComPrint(NUM_LEITURAS);
        Serial.printf("Raw medio atual (offset ja removido): %ld\n", raw);
        break;
    }
    case 'p':
    case 'P': {
        float peso = scale.get_units(10);
        Serial.printf("Peso atual: %.3f g\n", peso);
        break;
    }
    default:
        break;
}
mostrarMenu();
}
}

```

D.2 Dispensing Weight Test Sketch

Firmware sketch used in Phase 3 (Section 4.5): a single key press over serial triggers one full dispensing rotation, after which the load cell is read repeatedly in grams, so that the resulting weight profile can be observed.

```

#include <HX711.h>
#include <ESP32Servo.h>

#define PINO_DOUT 22
#define PINO_SCK 21
#define PINO_SERVO 15

#define CALIBRATION_FACTOR -10650.0f

const int DELAY_PASSO_SERVO_MS = 5;
const int DELAY_ASSENTAMENTO_MS = 1500;
const int NUM_LEITURAS_POS_QUEDA = 30;
const int DELAY_ENTRE_LEITURAS_MS = 100;

HX711 scale;

```

```
Servo servol;

void rodarCilindro(int anguloOrigem, int anguloDestino) {
  if (anguloOrigem < anguloDestino) {
    for (int a = anguloOrigem; a <= anguloDestino; a++) {
      servol.write(a);
      delay(DELAY_PASSO_SERVO_MS);
    }
  } else {
    for (int a = anguloOrigem; a >= anguloDestino; a--) {
      servol.write(a);
      delay(DELAY_PASSO_SERVO_MS);
    }
  }
}

void executarDispensaEMedir() {
  Serial.println("\nA tarar a plataforma antes da dispensa...");
  scale.tare(20);
  delay(DELAY_ASSENTAMENTO_MS);

  Serial.println("A rodar o cilindro (dispensa)...");
  rodarCilindro(0, 180);
  delay(500);
  rodarCilindro(180, 0);

  Serial.println("Leituras de peso apos a dispensa (g):");
  for (int i = 0; i < NUM_LEITURAS_POS_QUEDA; i++) {
    float peso = scale.get_units(1);
    Serial.printf("  #02d -> %.3f g\n", i + 1, peso);
    delay(DELAY_ENTRE_LEITURAS_MS);
  }

  Serial.println("Fim do registo. Envia qualquer tecla para repetir.");
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  servol.attach(PINO_SERVO, 500, 2400);
  servol.write(0);
  delay(600);

  scale.begin(PINO_DOUT, PINO_SCK);
  delay(500);
  if (!scale.is_ready()) {
    Serial.println("[ERRO] HX711 nao detectado! Verifica ligacoes.");
    while (true);
  }
}
```

```

    }
    scale.set_scale(CALIBRATION_FACTOR);

    Serial.println("[INFO] Pronto. Envia qualquer tecla para dispensar e medir.");
}

void loop() {
    if (Serial.available() > 0) {
        while (Serial.available() > 0) Serial.read();
        executarDispensaEMedir();
    }
}

```

D.3 Jamming Mitigation Test Sketch

Firmware sketch used in Phase 4 (Section 4.6), with the final, more aggressive oscillation parameters used for the reported jamming trials: 10 repetitions, 0° to 30°, 25 ms dwell at each extreme, and a 3 ms per degree step for the oscillation (compared to the 5 ms per degree step used for the main dispensing rotation).

```

/*
 * Teste de caracterizacao mecanica - dispensa com mitigacao de encravamento
 * Mecanismo de cilindro rotativo, ESP32-C6
 *
 * Antes de cada rotacao de dispensa (0 -> 180), o cilindro executa
 * um movimento oscilatorio de pequena amplitude, com o objetivo de
 * reposicionar unidades que possam estar bloqueadas no compartimento.
 */

#include <ESP32Servo.h>

#define PINO_SERVO 15

const int DELAY_PASSO_SERVO_MS = 5;
const int DELAY_PASSO_SERVO_JAMING_MS = 3;

#define JAM_WIGGLE_REPETICOES 10
#define JAM_WIGGLE_ANGULO 30
#define JAM_WIGGLE_DELAY_MS 25

#define NUM_CICLOS_DEFAULT 5
#define DELAY_ENTRE_CICLOS_MS 2000

Servo servo1;

void rodarCilindro(int anguloOrigem, int anguloDestino) {
    Serial.printf("Rotacao: %d -> %d graus\n", anguloOrigem, anguloDestino);
    if (anguloOrigem < anguloDestino) {

```

```

    for (int a = anguloOrigem; a <= anguloDestino; a++) { servol.write(a); delay(DELAY_PASSO_SERV
} else {
    for (int a = anguloOrigem; a >= anguloDestino; a--) { servol.write(a); delay(DELAY_PASSO_SERV
}
}

void desencravarCilindro() {
    Serial.printf("Oscilacao preventiva: %d repeticoes, 0-%d graus\n",
                  JAM_WIGGLE_REPETICOES, JAM_WIGGLE_ANGULO);

    for (int i = 0; i < JAM_WIGGLE_REPETICOES; i++) {
        for (int a = 0; a <= JAM_WIGGLE_ANGULO; a++) { servol.write(a); delay(DELAY_PASSO_SERVO_JAMIN
        delay(JAM_WIGGLE_DELAY_MS);
        for (int a = JAM_WIGGLE_ANGULO; a >= 0; a--) { servol.write(a); delay(DELAY_PASSO_SERVO_JAMIN
        delay(JAM_WIGGLE_DELAY_MS);
    }
}

void executarCiclo(int numeroCiclo) {
    Serial.printf("\nCiclo %d\n", numeroCiclo);
    desencravarCilindro();
    rodarCilindro(0, 180);
    delay(500);
    rodarCilindro(180, 0);
    Serial.println("Registrar resultado observado.");
}

void executarSerie(int numCiclos) {
    Serial.printf("\nInicio da serie de testes - %d ciclos\n", numCiclos);
    for (int i = 0; i < numCiclos; i++) {
        executarCiclo(i + 1);
        delay(DELAY_ENTRE_CICLOS_MS);
    }
    Serial.printf("\nSerie concluida - %d ciclos executados\n", numCiclos);
}

void mostrarMenu() {
    Serial.println("\n1 - Executar um ciclo");
    Serial.println("c - Executar serie de 5 ciclos");
    Serial.println("n - Executar serie (numero de ciclos definido pelo utilizador)");
}

void setup() {
    Serial.begin(115200);
    delay(1000);

    servol.attach(PINO_SERVO, 500, 2400);
    servol.write(0);
    delay(600);

```

```

    mostrarMenu();
}

void loop() {
    if (Serial.available() > 0) {
        char c = Serial.read();
        while (Serial.available() > 0) Serial.read();

        switch (c) {
            case '1':
                executarCiclo(1);
                break;
            case 'c':
            case 'C':
                executarSerie(NUM_CICLOS_DEFAULT);
                break;
            case 'n':
            case 'N': {
                Serial.println("Numero de ciclos:");
                while (Serial.available() == 0) delay(50);
                int n = Serial.parseInt();
                if (n <= 0) n = NUM_CICLOS_DEFAULT;
                executarSerie(n);
                break;
            }
            default:
                break;
        }
        mostrarMenu();
    }
}

```

D.4 Final Production Firmware

The complete final firmware, integrating the calibrated gram-based detection logic, the stability-based drop and collection confirmation, the anti-jam wiggle routine, the escalating acoustic alarm, and the Firebase Realtime Database and FCM notification integration, as described throughout [Section 3.4](#).

```

// =====
//  DISPENSADOR IoT - FASE 4 (ESP32-C6 RISC-V)
//  Mecanismo de Cilindro Rotativo + Alarme Escalonado
//
//  Detecção de queda e recolha por peso (gramas), com fator de
//  calibração e leitura de baseline estavel antes de cada dispensa.
//  Inclui rotina de anti-encravamento (oscilacao antes da rotacao
//  principal) e escalada progressiva do alarme sonoro.

```



```
//
//  SERVOS_CONECTADOS = 3, pinos {15, 18, 19}. Nao ha retry
//  periodico de Wi-Fi no arranque: assume-se que a rede esta
//  disponivel quando o dispositivo liga; se falhar, segue em
//  modo offline (cache local + RTC).
//  =====

#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
#include <ESP32Servo.h>
#include "RTCLib.h"
#include <Wire.h>
#include "time.h"
#include "DHT.h"
#include <HX711.h>

//  =====
//  FORWARD DECLARATIONS
//  =====
void verificarAlertasAmbientais(float temp, float hum);
void verificarDosesEmAtraso(const String &diaHoje);
void verificarComandoManual();
void enviarParaHistorico(String medication, int comp, int qtd, String horaAgendada, String status);
void dispararAlertaPush(String title, String message);
void tocarBuzzerAlerta(int bipes);
void rodarCilindroSuave(int compartment, int anguloOrigem, int anguloDestino);
void desencravarCilindro(int compartment);
float lerPesoGramas();
float lerPesoBaseEstavel();

//  =====
//  1. CREDENCIAIS
//  =====
#define WIFI_SSID      "StressLess"
#define WIFI_PASSWORD  "12345678"
#define API_KEY        "AIzaSyB0y2n907CmTmilBSqCnEvsu4KwJkSIQjE"
#define DATABASE_URL   "tesemestrado-5a168-default-rtdb.europe-west1.firebaseioapp"

//  =====
//  2. PINOS
//  =====
#define PINO_DHT11      4
#define TIPO_DHT        DHT11
#define PINO_HX711_DOUT 22
#define PINO_HX711_SCK  21
#define PINO_BUZZER     5
```

```

const int NUM_SERVOS          = 3;
const int PINOS_SERVOS[NUM_SERVOS] = {15, 18, 19};
const int SERVOS_CONECTADOS = 3;

#define PINO_ALCAPAO          14
const bool ATIVAR_ALCAPAO_NESTA_FASE = false;

// =====
// PARÂMETROS CINEMÁTICOS
// 5ms/grau foi o valor que deu melhor taxa de sucesso nos testes
// de caracterização de velocidade (Fase A: 13/15 = 86.7%).
// =====
const int DELAY_PASSO_SERVO_MS      = 5;    // ms por grau de rotação

const int DELAY_ASSENTAMENTO_MED_MS    = 2000;
const int DELAY_ASSENTAMENTO_UNIDADE_MS = 2000;

// =====
// PROTEÇÃO CONTRA ENCRAVAMENTO (JAMMING)
// Antes de cada rotação de dispensa, o cilindro oscila entre 0 e um
// pequeno angulo algumas vezes, de forma a "abanar" um medicamento
// que possa estar mal posicionado ou preso no compartimento, antes
// de avançar com a rotação completa de 180 graus.
// =====
#define JAM_WIGGLE_REPETICOES    3    // No de vaivens antes da rotação de dispensa
#define JAM_WIGGLE_ANGULO        15    // Angulo (graus) do vaivem, a partir de 0
#define JAM_WIGGLE_DELAY_MS      150    // Pausa entre cada movimento do vaivem

// =====
// 3. LOAD CELL - DETECÇÃO POR PESO (GRAMAS), COM FACTOR DE CALIBRAÇÃO
// Calibração feita com moeda de 1 euro (7.5g), 3 repetições com
// tare entre cada:
//   Rep 1: raw = -79878   Rep 2: raw = -80039   Rep 3: raw = -79709
//   Raw medio: -79875   (variação min-max ~0.41%)
//   CALIBRATION_FACTOR = raw_medio / peso_conhecido = -10650.0
//
// Validado com medicamento real (Benuron 1000, 5 repetições):
//   Peso medio: 1.217 g   Desvio padrao: 0.004 g   (CV ~0.3%)
//
// Ruído de fundo (plataforma parada, baseline pos-tare): raw ~ +-220
//   -> em gramas: 220 / 10650 ~ 0.02 g
// =====
#define CALIBRATION_FACTOR      -10650.0000f

#define LEITURAS_MEDIA          10        // Média de leituras por amostra HX711
#define TIMEOUT_QUEDA_MS        6000      // Janela maxima para confirmar a queda
#define INTERVALO_POLLING_MS    300       // Intervalo entre leituras durante detecção

// Threshold de queda em GRAMAS.

```

```

#define PILL_THRESHOLD_G      0.3f

// N° de leituras consecutivas que o delta de peso tem de ficar
// acima de PILL_THRESHOLD_G antes de confirmar a queda. Evita que um
// toque/vibração momentânea seja interpretada como uma queda real:
// uma queda genuína mantém o peso elevado de forma estável, um
// toque é tipicamente instantâneo.
#define LEITURAS_QUEDA_ESTAVEL 5      // 5 x 300ms = 1.5s de peso estável

// Threshold de recolha em GRAMAS.
#define COLLECTION_THRESHOLD_G 0.15f

// Confirmação de recolha: a plataforma deve estar dentro de
// COLLECTION_THRESHOLD_G durante LEITURAS_ESTABILIZACAO leituras
// consecutivas (a 300ms cada = 3 segundos de estabilidade minima).
#define LEITURAS_ESTABILIZACAO 10

// Assentamento apos tare, antes de sequer ler a base.
#define DELAY_ASSENTAMENTO_TARE_MS 1500

// Leitura da base (antes de rodar o servo) feita com varias amostras
// em vez de uma so, para confirmar estabilidade mecanica previa.
#define LEITURAS_BASE_ESTAVEL 4

// =====
// ESCALADA DO ALARME ACUSTICO - thresholds configuraveis
// Valores de PRODUÇÃO (comentar estas 6 linhas e descomentar o
// bloco de TESTE ACELERADO abaixo para validar a escalada sem
// esperar 5/10 minutos reais):
// =====
#define ALARME_NIVEL2_MS      300000UL    // 5 min: 1 -> 2 bipes
#define ALARME_NIVEL3_MS      600000UL    // 10 min: 2 -> 3 bipes
#define ALARME_JANELA_TOQUE_MS 30000UL     // janela de som ativa por ciclo de 5 min
#define ALARME_INTERVALO_N1_MS 5000UL      // intervalo entre bipes, nivel 1
#define ALARME_INTERVALO_N2_MS 3000UL      // intervalo entre bipes, nivel 2
#define ALARME_INTERVALO_N3_MS 1500UL      // intervalo entre bipes, nivel 3

// ---- TESTE ACELERADO (descomentar estas 6 linhas, comentar as 6 acima) ----
// #define ALARME_NIVEL2_MS      30000UL    // 30s em vez de 5 min
// #define ALARME_NIVEL3_MS      60000UL    // 60s em vez de 10 min
// #define ALARME_JANELA_TOQUE_MS 3000UL     // janela de som proporcional (10%)
// #define ALARME_INTERVALO_N1_MS 500UL
// #define ALARME_INTERVALO_N2_MS 300UL
// #define ALARME_INTERVALO_N3_MS 150UL

// =====
// 4. HARDWARE
// =====
Servo      servos[NUM_SERVOS];

```

```

Servo      servoAlcapao;
RTC_DS3231 rtc;
DHT        dht(PINO_DHT11, TIPO_DHT);
HX711      scale;

// =====
// 5. FIREBASE
// =====
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

const char* ntpServer = "pool.ntp.org";
const char* fusoHorario = "WET0WEST,M3.5.0/1,M10.5.0/2";

// =====
// 6. TEMPORIZAÇÃO
// =====
unsigned long tAnteriorDispensa = 0;
unsigned long tAnteriorSensores = 0;
unsigned long tAnteriorComando = 0;

const long INTERVALO_DISPENSA = 5000;
const long INTERVALO_SENTORES = 4000;
const long INTERVALO_COMANDO = 2000;

String ultimaHoraDispensada = "";

// =====
// 7. CACHE LOCAL DE DOSES
// =====
const int MAX_DOSES = 10;

struct Dose {
    String pushID;
    String hora;
    String medicamento;
    int    compartimento;
    int    quantidade;
    bool   dispensada;
    String status;
    bool   escritaPendente;
    String dataCriacao;
    String horaCriacao;
    int    atraso;
};

Dose cacheDoses[MAX_DOSES];
int numDosesCache = 0;

```

```

bool cacheValido    = false;

struct LogHistorico {
    String medicamento;
    int    compartimento;
    int    quantidade;
    String horaAgendada;
    String status;
    String timestamp;
    bool    valido = false;
};

const int MAX_LOGS_PENDENTES = 15;
LogHistorico cacheLogsPendentes[MAX_LOGS_PENDENTES];
int numLogsPendentes = 0;

// =====
// 8. FUNÇÕES DE TEMPO
// =====

String getDiaSemana() {
    const char* dias[] = {"domingo", "segunda-feira", "terca-feira", "quarta-feira", "quinta-feira", "sexta-feira", "sabado", "domingo"};
    struct tm t;
    if (!getLocalTime(&t)) {
        DateTime agora = rtc.now();
        return String(dias[agora.dayOfTheWeek()]);
    }
    return String(dias[t.tm_wday]);
}

String getHoraAtual() {
    struct tm t;
    if (!getLocalTime(&t)) {
        DateTime agora = rtc.now();
        char buf[6];
        sprintf(buf, "%02d:%02d", agora.hour(), agora.minute());
        return String(buf);
    }
    char buf[6];
    sprintf(buf, "%02d:%02d", t.tm_hour, t.tm_min);
    return String(buf);
}

String getTimestamp() {
    struct tm t;
    if (!getLocalTime(&t)) {
        DateTime agora = rtc.now();
        char buf[20];
        sprintf(buf, "%04d-%02d-%02d %02d:%02d:%02d", agora.year(), agora.month(), agora.day(), agora.hour(), agora.minute(), agora.second());
        return String(buf);
    }
    char buf[20];
    sprintf(buf, "%04d-%02d-%02d %02d:%02d:%02d", t.tm_year+1900, t.tm_mon+1, t.tm_mday, t.tm_hour, t.tm_min, t.tm_sec);
    return String(buf);
}

```

```

    }
    char buf[20];
    sprintf(buf, "%04d-%02d-%02d %02d:%02d:%02d", t.tm_year+1900, t.tm_mon+1, t.tm_mday, t.tm_hour,
    return String(buf);
}

String getDataAtual() {
    struct tm t;
    if (!getLocalTime(&t)) {
        DateTime agora = rtc.now();
        char buf[11];
        sprintf(buf, "%04d-%02d-%02d", agora.year(), agora.month(), agora.day());
        return String(buf);
    }
    char buf[11];
    sprintf(buf, "%04d-%02d-%02d", t.tm_year + 1900, t.tm_mon + 1, t.tm_mday);
    return String(buf);
}

// =====
// 9. LOAD CELL
// =====

float lerPesoGramas() {
    if (!scale.is_ready()) return 0.0f;
    return scale.get_units(LEITURAS_MEDIA);
}

float lerPesoBaseEstavel() {
    float leituras[LEITURAS_BASE_ESTAVEL];
    float soma = 0;

    Serial.println("[BASE] A verificar estabilidade antes da dispensa:");
    for (int i = 0; i < LEITURAS_BASE_ESTAVEL; i++) {
        leituras[i] = lerPesoGramas();
        Serial.printf("  #d -> %.3fg\n", i + 1, leituras[i]);
        soma += leituras[i];
        delay(150);
    }

    float media = soma / LEITURAS_BASE_ESTAVEL;
    float minV = leituras[0], maxV = leituras[0];
    for (int i = 1; i < LEITURAS_BASE_ESTAVEL; i++) {
        if (leituras[i] < minV) minV = leituras[i];
        if (leituras[i] > maxV) maxV = leituras[i];
    }
    float variacao = maxV - minV;

    Serial.printf("[BASE] Media=%.3fg | Variacao=%.3fg\n", media, variacao);
    if (variacao > COLLECTION_THRESHOLD_G) {

```

```

        Serial.println("[BASE][AVISO] Variacao elevada antes da dispensa - plataforma pode nao estar
    }

    return media;
}

// =====
// 10. GESTÃO DA AGENDA / RESET
// =====
void resetarDosesDodia(const String &diaHoje) {
    String caminho = "/agenda/" + diaHoje;
    if (!Firebase.RTDB.getJSON(&fbdo, caminho)) return;
    FirebaseJson &json = fbdo.jsonObject();
    size_t numFilhos = json.iteratorBegin();
    for (size_t i = 0; i < numFilhos; i++) {
        String pushID, valorJSON;
        int tipo;
        json.iteratorGet(i, tipo, pushID, valorJSON);
        Firebase.RTDB.setBool(&fbdo, caminho + "/" + pushID + "/dispensada", false);
        Firebase.RTDB.setString(&fbdo, caminho + "/" + pushID + "/status", "pending");
    }
    json.iteratorEnd();
    Firebase.RTDB.setBool(&fbdo, "/alertas/falha_dispositivo", false);
    Firebase.RTDB.setString(&fbdo, "/sistema/ultima_data_reset", getDataAtual());
}

void verificarResetDiario() {
    if (!Firebase.ready()) return;
    String dataHoje = getDataAtual();
    String ultimoReset = "";
    if (Firebase.RTDB.getString(&fbdo, "/sistema/ultima_data_reset"))
        ultimoReset = fbdo.to<String>();
    if (ultimoReset != dataHoje)
        resetarDosesDodia(getDiaSemana());
}

void atualizarCacheDoFirebase() {
    String diaHoje = getDiaSemana();
    String caminho = "/agenda/" + diaHoje;

    if (!Firebase.RTDB.getJSON(&fbdo, caminho)) {
        String erro = fbdo.errorReason();
        if (erro.indexOf("not exist") >= 0 || erro.indexOf("not found") >= 0) {
            numDosesCache = 0;
            cacheValido = true;
        }
        return;
    }
}

```

```

FirebaseJson &json = fbdo.jsonObject();
size_t numFilhos = json.iteratorBegin();
if (numFilhos == 0) {
    json.iteratorEnd();
    numDosesCache = 0;
    cacheValido = true;
    return;
}

int count = 0;
for (size_t i = 0; i < numFilhos && count < MAX_DOSES; i++) {
    String pushID, valorJSON;
    int tipo;
    json.iteratorGet(i, tipo, pushID, valorJSON);
    FirebaseJson dataJson;
    dataJson.setJsonData(valorJSON);
    FirebaseJsonData resHora, resMed, resComp, resDisp, resQtd, resStatus, resDataCriacao, resHor

    dataJson.get(resHora,      "hora");
    dataJson.get(resMed,       "medicamento");
    dataJson.get(resComp,      "compartimento");
    dataJson.get(resQtd,       "quantidade");
    dataJson.get(resDisp,      "dispensada");
    dataJson.get(resStatus,    "status");
    dataJson.get(resDataCriacao, "data_criacao");
    dataJson.get(resHoraCriacao, "hora_criacao");
    dataJson.get(resAtraso,    "atraso");

    if (!resHora.success) continue;

    bool pendente = false;
    for (int j = 0; j < numDosesCache; j++) {
        if (cacheDoses[j].pushID == pushID && cacheDoses[j].escritaPendente) {
            pendente = true;
            break;
        }
    }

    cacheDoses[count].pushID      = pushID;
    cacheDoses[count].hora        = resHora.stringValue;
    cacheDoses[count].medicamento = resMed.success ? resMed.stringValue : "unknown";
    cacheDoses[count].compartimento = resComp.success ? resComp.intValue : 1;
    cacheDoses[count].quantidade  = resQtd.success ? resQtd.intValue : 1;
    cacheDoses[count].dispensada  = resDisp.success ? resDisp.boolValue : false;
    cacheDoses[count].status      = resStatus.success ? resStatus.stringValue : "pending";
    cacheDoses[count].dataCriacao  = resDataCriacao.success ? resDataCriacao.stringValue : "";
    cacheDoses[count].horaCriacao  = resHoraCriacao.success ? resHoraCriacao.stringValue : "";
    cacheDoses[count].atraso       = resAtraso.success ? resAtraso.intValue : 30;
    cacheDoses[count].escritaPendente = pendente;

```



```

        count++;
    }
    json.iteratorEnd();
    numDosesCache = count;
    cacheValido = true;

    Serial.println("\n----- [NOSQL SYNC]: AGENDA -----");
    for (int i = 0; i < numDosesCache; i++) {
        Serial.printf("[%s] %s | Gaveta: %d | Qtd: %d | Janela: %d min | Status: %s\n",
            cacheDoses[i].hora.c_str(), cacheDoses[i].medicamento.c_str(),
            cacheDoses[i].compartimento, cacheDoses[i].quantidade,
            cacheDoses[i].atraso, cacheDoses[i].status.c_str());
    }
    Serial.println("-----");
    verificarDosesEmAtraso(diaHoje);
}

void verificarDosesEmAtraso(const String &diaHoje) {
    if (!Firebase.ready() || !cacheValido) return;
    String horaAgora = getHoraAtual();
    String dataHoje = getDataAtual();
    int detected = 0;
    String listaMeds = "";

    for (int i = 0; i < numDosesCache; i++) {
        if (cacheDoses[i].dispensada) continue;
        if (cacheDoses[i].status == "taken" || cacheDoses[i].status == "missed" ||
            cacheDoses[i].status == "failure" || cacheDoses[i].status == "device_offline") continue;
        if (cacheDoses[i].dataCriacao == dataHoje &&
            cacheDoses[i].hora.compareTo(cacheDoses[i].horaCriacao) < 0) continue;
        if (cacheDoses[i].hora.compareTo(horaAgora) < 0) {
            enviarParaHistorico(cacheDoses[i].medicamento, cacheDoses[i].compartimento,
                cacheDoses[i].quantidade, cacheDoses[i].hora, "device_offline");
            String caminho = "/agenda/" + diaHoje + "/" + cacheDoses[i].pushID;
            Firebase.RTDB.setBool(&fbdo, caminho + "/dispensada", false);
            Firebase.RTDB.setString(&fbdo, caminho + "/status", "device_offline");
            cacheDoses[i].dispensada = false;
            cacheDoses[i].status = "device_offline";
            if (detected > 0) listaMeds += ", ";
            listaMeds += cacheDoses[i].medicamento;
            detected++;
        }
    }
    if (detected > 0) {
        Firebase.RTDB.setBool(&fbdo, "/alertas/falha_dispositivo", true);
        dispararAlertaPush("MedBox Power Alert",
            String(detected) + " dose(s) missed (device was off): " + listaMeds);
    }
}

```

```

void sincronizarEscritasPendentes() {
    String diaHoje = getDiaSemana();
    for (int i = 0; i < numDosesCache; i++) {
        if (!cacheDoses[i].escritaPendente) continue;
        String caminho = "/agenda/" + diaHoje + "/" + cacheDoses[i].pushID;
        Firebase.RTDB.setBool(&fbdo, caminho + "/dispensada", cacheDoses[i].dispensada);
        Firebase.RTDB.setString(&fbdo, caminho + "/status", cacheDoses[i].status);
        cacheDoses[i].escritaPendente = false;
    }
    if (Firebase.ready() && numLogsPendentes > 0) {
        for (int i = 0; i < MAX_LOGS_PENDENTES; i++) {
            if (!cacheLogsPendentes[i].valido) continue;
            FirebaseJson history;
            history.set("medication", cacheLogsPendentes[i].medicamento);
            history.set("drawer", cacheLogsPendentes[i].compartimento);
            history.set("quantity", cacheLogsPendentes[i].quantidade);
            history.set("scheduled_time", cacheLogsPendentes[i].horaAgendada);
            history.set("status", cacheLogsPendentes[i].status);
            history.set("timestamp", cacheLogsPendentes[i].timestamp);
            if (Firebase.RTDB.pushJSON(&fbdo, "/history", &history)) {
                cacheLogsPendentes[i].valido = false;
                numLogsPendentes--;
            }
        }
    }
}

// =====
// 11. ACTUAÇÃO DO SERVO
// =====
void rodarCilindroSuave(int compartment, int anguloOrigem, int anguloDestino) {
    if (compartment < 1 || compartment > SERVOS_CONECTADOS) return;
    int idx = compartment - 1;
    Serial.printf("[SERVO] Compartimento %d: %d -> %d graus (%dms/grau)\n",
        compartment, anguloOrigem, anguloDestino, DELAY_PASSO_SERVO_MS);
    if (anguloOrigem < anguloDestino) {
        for (int a = anguloOrigem; a <= anguloDestino; a++) { servos[idx].write(a); delay(DELAY_PASSO_SERVO_MS); }
    } else {
        for (int a = anguloOrigem; a >= anguloDestino; a--) { servos[idx].write(a); delay(DELAY_PASSO_SERVO_MS); }
    }
}

void desencravarCilindro(int compartment) {
    if (compartment < 1 || compartment > SERVOS_CONECTADOS) return;
    int idx = compartment - 1;

    Serial.printf("[JAM] Compartimento %d: a agitar cilindro (%d vaivens, 0-%d graus)\n",
        compartment, JAM_WIGGLE_REPETICOES, JAM_WIGGLE_ANGULO);
}

```

```

    for (int i = 0; i < JAM_WIGGLE_REPETICOES; i++) {
        for (int a = 0; a <= JAM_WIGGLE_ANGULO; a++) { servos[idx].write(a); delay(DELAY_PASSO_SERVO);
        delay(JAM_WIGGLE_DELAY_MS);
        for (int a = JAM_WIGGLE_ANGULO; a >= 0; a--) { servos[idx].write(a); delay(DELAY_PASSO_SERVO);
        delay(JAM_WIGGLE_DELAY_MS);
    }

    Serial.println("[JAM] Agitacao concluida, cilindro de volta a 0.");
}

// =====
// 12. HISTÓRICO E ALERTAS
// =====
void enviarParaHistorico(String medication, int comp, int qtd, String horaAgendada, String status) {
    String tsAtual = getTimestamp();
    if (!Firebase.ready()) {
        if (numLogsPendentes < MAX_LOGS_PENDENTES) {
            for (int i = 0; i < MAX_LOGS_PENDENTES; i++) {
                if (!cacheLogsPendentes[i].valido) {
                    cacheLogsPendentes[i] = {medication, comp, qtd, horaAgendada, status, tsAtual, true};
                    numLogsPendentes++;
                    break;
                }
            }
        }
        return;
    }
    FirebaseJson history;
    history.set("medication", medication);
    history.set("drawer", comp);
    history.set("quantity", qtd);
    history.set("scheduled_time", horaAgendada);
    history.set("status", status);
    history.set("timestamp", tsAtual);
    Firebase.RTDB.pushJSON(&fbdo, "/history", &history);
}

void dispararAlertaPush(String title, String message) {
    Serial.printf("[PUSH] %s | %s\n", title.c_str(), message.c_str());
}

void tocarBuzzerAlerta(int bipes) {
    for (int i = 0; i < bipes; i++) {
        digitalWrite(PINO_BUZZER, HIGH); delay(120);
        digitalWrite(PINO_BUZZER, LOW); delay(80);
    }
    Serial.printf("[BUZZER] %d bipes\n", bipes);
}

```

```
// =====
// 13. DISPENSA ANTECIPADA (MANUAL)
// =====

void verificarComandoManual() {
    if (!Firebase.ready()) return;
    if (!Firebase.RTDB.getBool(&fbdo, "/comando_manual/executar")) return;
    if (!fbdo.to<bool>()) return;

    String gavetasStr = "";
    if (Firebase.RTDB.getString(&fbdo, "/comando_manual/gavetas"))
        gavetasStr = fbdo.to<String>();
    int compAlvo = gavetasStr.toInt();
    if (compAlvo < 1 || compAlvo > SERVOS_CONECTADOS) {
        Firebase.RTDB.setBool(&fbdo, "/comando_manual/executar", false);
        return;
    }

    tocarBuzzerAlerta(4);
    scale.tare();
    delay(DELAY_ASSENTAMENTO_TARE_MS);
    float pesoBase = lerPesoBaseEstavel();

    desencravarCilindro(compAlvo);
    rodarCilindroSuave(compAlvo, 0, 180);

    unsigned long tInicio = millis();
    bool quedaMed = false;
    int contadorQueda = 0;
    while (millis() - tInicio < TIMEOUT_QUEDA_MS) {
        float pesoAtual = lerPesoGramas();
        float deltaPeso = pesoAtual - pesoBase;
        Serial.printf("[MANUAL QUEDA] base=%.3fg | atual=%.3fg | delta=%.3fg | threshold=%.3fg | esta
                        pesoBase, pesoAtual, deltaPeso, PILL_THRESHOLD_G, contadorQueda, LEITURAS_QUEDA);
        if (deltaPeso > PILL_THRESHOLD_G) {
            contadorQueda++;
            if (contadorQueda >= LEITURAS_QUEDA_ESTAVEL) { quedaMed = true; break; }
        } else {
            if (contadorQueda > 0) {
                Serial.println("[MANUAL QUEDA][RESET] delta caiu abaixo do threshold - toque momentaneo,
            }
            contadorQueda = 0;
        }
    }
    delay(INTERVALO_POLLING_MS);
}

rodarCilindroSuave(compAlvo, 180, 0);
enviarParaHistorico("Dispensa Antecipada", compAlvo, 1, "On-Demand", quedaMed ? "taken" : "fail");
Firebase.RTDB.setBool(&fbdo, "/comando_manual/executar", false);
```

```

    Firebase.RTDB.setString(&fbdo, "/comando_manual/status_final", quedaMed ? "sucesso" : "falha_qu
    Firebase.RTDB.setString(&fbdo, "/toma_atual/estado", quedaMed ? "tomado_antecipado" :
}

// =====
// 14. ROTINA PRINCIPAL DE DISPENSA
// =====
void dispensarDosesAgrupadas(int* indices, int numDoses, const String &diaHoje) {

    String medNomes[MAX_DOSES], medHoras[MAX_DOSES], medPushIDs[MAX_DOSES];
    int    medComps[MAX_DOSES], medQtDs[MAX_DOSES];
    for (int d = 0; d < numDoses; d++) {
        medNomes[d]    = cacheDoses[indices[d]].medicamento;
        medComps[d]    = cacheDoses[indices[d]].compartimento;
        medQtDs[d]     = cacheDoses[indices[d]].quantidade;
        medHoras[d]    = cacheDoses[indices[d]].hora;
        medPushIDs[d] = cacheDoses[indices[d]].pushID;
    }
    String listaMeds = "";
    for (int d = 0; d < numDoses; d++) { if (d) listaMeds += ", "; listaMeds += medNomes[d]; }

    int atrasoEscolhido = cacheDoses[indices[0]].atraso;
    for (int d = 1; d < numDoses; d++) {
        if (cacheDoses[indices[d]].atraso < atrasoEscolhido)
            atrasoEscolhido = cacheDoses[indices[d]].atraso;
    }

    Serial.printf("\n[DISPENSA] %d dose(s): %s | Janela: %d min\n", numDoses, listaMeds.c_str(), at

    scale.tare();
    delay(DELAY_ASSENTAMENTO_TARE_MS);
    bool quedaConfirmada[MAX_DOSES] = {};

    for (int d = 0; d < numDoses; d++) {
        int compAlvo = medComps[d];
        if (compAlvo > SERVOS_CONECTADOS) continue;
        int unidadesConfirmadas = 0;

        for (int q = 0; q < medQtDs[d]; q++) {
            float pesoBase = lerPesoBaseEstavel();

            desencravarCilindro(compAlvo);
            rodarCilindroSuave(compAlvo, 0, 180);

            unsigned long tQueda = millis();
            bool unidadeCaiu = false;
            int contadorQueda = 0;
            while (millis() - tQueda < TIMEOUT_QUEDA_MS) {
                float pesoAtual = lerPesoGramas();

```

```

float deltaPeso = pesoAtual - pesoBase;
Serial.printf("[QUEDA %dms] base=%.3fg | atual=%.3fg | delta=%.3fg | threshold=%.3fg | es
              (int)(millis()-tQueda), pesoBase, pesoAtual, deltaPeso, PILL_THRESHOLD_G, c
if (deltaPeso > PILL_THRESHOLD_G) {
    contadorQueda++;
    if (contadorQueda >= LEITURAS_QUEDA_ESTAVEL) { unidadeCaiu = true; break; }
} else {
    if (contadorQueda > 0) {
        Serial.println("[QUEDA][RESET] delta caiu abaixo do threshold - toque momentaneo, nao
    }
    contadorQueda = 0;
}
delay(INTERVALO_POLLING_MS);
}

rodarCilindroSuave(compAlvo, 180, 0);
if (unidadeCaiu) unidadesConfirmadas++;

if (q < medQtDs[d] - 1) delay(DELAY_ASSENTAMENTO_UNIDADE_MS);
}

if (unidadesConfirmadas == medQtDs[d]) quedaConfirmada[d] = true;

if (d < numDoses - 1) delay(DELAY_ASSENTAMENTO_MED_MS);
}

String horaDispensa = getTimestamp();
bool algumaQuedaConfirmada = false;
for (int d = 0; d < numDoses; d++) if (quedaConfirmada[d]) { algumaQuedaConfirmada = true; brea

bool recolhaConfirmada = false;
String horaRecolha = "";
bool atingiuNivel3Alarme = false;

if (algumaQuedaConfirmada) {
    if (Firebase.ready())
        Firebase.RTDB.setString(&fbdo, "/toma_atual/estado", "aguarda_recolha");

    unsigned long tInicioRecolha = millis();
    unsigned long timeoutRecolha = (unsigned long)atrasoEscolhido * 60000UL;

    unsigned long tUltimoBipe = millis();
    unsigned long tUltimoBlocoToque = 0;
    bool emJanelaDeToque = true;

    int contadorRecolha = 0;

    while (millis() - tInicioRecolha < timeoutRecolha) {
        float pesoAtual = lerPesoGramas();

```

```

    unsigned long elapsed = millis() - tInicioRecolha;

    Serial.printf("[RECOLHA PESO] t=%lus | peso=%.3fg | threshold=+-.3fg | estavel=%d/%d\n",
        elapsed / 1000, pesoAtual, COLLECTION_THRESHOLD_G,
        contadorRecolha, LEITURAS_ESTABILIZACAO);

    if (fabs(pesoAtual) < COLLECTION_THRESHOLD_G) {
        contadorRecolha++;
        if (contadorRecolha >= LEITURAS_ESTABILIZACAO) {
            recolhaConfirmada = true;
            horaRecolha = getTimestamp();
            Serial.printf("[RECOLHA CONFIRMADA] peso=%.3fg | Plataforma estavel durante %d leituras
                pesoAtual, LEITURAS_ESTABILIZACAO, horaRecolha.c_str());

            break;
        }
    } else {
        if (contadorRecolha > 0) {
            Serial.printf("[RECOLHA RESET] peso=%.3fg fora do threshold\n", pesoAtual);
            contadorRecolha = 0;
        }
    }

    if (elapsed >= ALARME_NIVEL3_MS) atingiuNivel3Alarme = true;

    if (elapsed - tUltimoBlocoToque >= ALARME_NIVEL2_MS) {
        tUltimoBlocoToque = elapsed;
        emJanelaDeToque = true;
    }
    if (emJanelaDeToque && (elapsed - tUltimoBlocoToque > ALARME_JANELA_TOQUE_MS)) {
        emJanelaDeToque = false;
    }
    if (emJanelaDeToque) {
        unsigned long intervalo = ALARME_INTERVALO_N1_MS;
        int bipes = 1;
        if (elapsed >= ALARME_NIVEL3_MS) { intervalo = ALARME_INTERVALO_N3_MS; bipes = 3; }
        else if (elapsed >= ALARME_NIVEL2_MS) { intervalo = ALARME_INTERVALO_N2_MS; bipes = 2; }
        if (millis() - tUltimoBipe >= intervalo) {
            tUltimoBipe = millis();
            tocarBuzzerAlerta(bipes);
        }
    }

    delay(INTERVALO_POLLING_MS);
}

if (!recolhaConfirmada) {
    Serial.println("[TIMEOUT] Dose nao recolhida dentro da janela.");
}
}

```

```

bool tomadoTarde = recolhaConfirmada && atingiuNivel3Alarme;

String estadoFinal = (!algumaQuedaConfirmada) ? "falha_mecanica"
        : (!recolhaConfirmada) ? "nao_recolhido"
        : tomadoTarde ? "tomado_atrasado"
        : "tomado";
String statusAgenda = (estadoFinal == "tomado") ? "taken"
        : (estadoFinal == "tomado_atrasado") ? "taken_late"
        : (estadoFinal == "nao_recolhido") ? "missed"
        : "failure";
bool foiTomado = (estadoFinal == "tomado" || estadoFinal == "tomado_atrasado");

for (int d = 0; d < numDoses; d++) {
    if (medPushIDs[d].isEmpty()) continue;
    for (int i = 0; i < numDosesCache; i++) {
        if (cacheDoses[i].pushID == medPushIDs[d]) {
            cacheDoses[i].dispensada = foiTomado;
            cacheDoses[i].status = statusAgenda;
            break;
        }
    }
    enviarParaHistorico(medNomes[d], medComps[d], medQtDs[d], medHoras[d], statusAgenda);
    if (Firebase.ready()) {
        String caminho = "/agenda/" + diaHoje + "/" + medPushIDs[d];
        Firebase.RTDB.setBool(&fbdo, caminho + "/dispensada", foiTomado);
        Firebase.RTDB.setString(&fbdo, caminho + "/status", statusAgenda);
    } else {
        for (int i = 0; i < numDosesCache; i++) {
            if (cacheDoses[i].pushID == medPushIDs[d]) { cacheDoses[i].escritaPendente = true; break; }
        }
    }
}

if (Firebase.ready()) {
    Firebase.RTDB.setString(&fbdo, "/toma_atual/estado", estadoFinal);
    Firebase.RTDB.setString(&fbdo, "/toma_atual/hora_dispensa", horaDispensa);
    Firebase.RTDB.setString(&fbdo, "/toma_atual/medicamentos", listaMeds);
    Firebase.RTDB.setInt(&fbdo, "/toma_atual/num_doses", numDoses);
    Firebase.RTDB.setBool(&fbdo, "/toma_atual/confirmacao_queda", algumaQuedaConfirmada);
    Firebase.RTDB.setBool(&fbdo, "/toma_atual/confirmacao_recolha", recolhaConfirmada);
    Firebase.RTDB.setString(&fbdo, "/toma_atual/hora_recolha", horaRecolha);
    Firebase.RTDB.setBool(&fbdo, "/toma_atual/alerta_ativo", !recolhaConfirmada);
    Firebase.RTDB.setBool(&fbdo, "/alertas/dose_em_atraso", !recolhaConfirmada);
    Firebase.RTDB.setBool(&fbdo, "/alertas/tomado_em_atraso", tomadoTarde);
}

Serial.printf("[DISPENSA CONCLUIDA] Estado: %s | Queda: %s | Recolha: %s\n",
    estadoFinal.c_str(),
    algumaQuedaConfirmada ? "confirmada" : "falha",

```



```

        recolhaConfirmada      ? "confirmada" : "nao confirmada");
    }

    // =====
    // 15. VERIFICAÇÃO DE DISPENSA (timer 5s)
    // =====
    void verificarDispensa() {
        String diaHoje    = getDiaSemana();
        String horaAgora = getHoraAtual();

        if (horaAgora == ultimaHoraDispensada) return;

        if (Firebase.ready()) {
            sincronizarEscritasPendentes();
            atualizarCacheDoFirebase();
        } else {
            if (!cacheValido) return;
        }

        int indicesAgora[MAX_DOSES];
        int numAgora = 0;
        for (int i = 0; i < numDosesCache; i++) {
            if (horaAgora == cacheDoses[i].hora && !cacheDoses[i].dispensada && cacheDoses[i].status == " ")
                indicesAgora[numAgora++] = i;
        }
        if (numAgora > 0) {
            ultimaHoraDispensada = horaAgora;
            dispensarDosesAgrupadas(indicesAgora, numAgora, diaHoje);
        }
    }

    // =====
    // 16. DHT11
    // =====
    void lerEPublicarDHT11() {
        float temp = dht.readTemperature();
        float hum  = dht.readHumidity();

        if (isnan(temp) || isnan(hum)) {
            Serial.printf("[DHT11 ERROR] Falha de leitura no GPIO %d\n", PINO_DHT11);
            if (Firebase.ready()) {
                Firebase.RTDB.setString(&fbdo, "/sensores/dht11_interno/status", "read_error");
                Firebase.RTDB.setString(&fbdo, "/sensores/dht11_interno/ultima_leitura", getTimestamp());
            }
            return;
        }

        Serial.printf("[DHT11] Temp: %d C | Hum: %d%%\n", (int)temp, (int)hum);
        if (Firebase.ready()) {

```

```

        Firebase.RTDB.setInt(    &fbdo, "/sensores/dht11_interno/temperatura",    (int)temp);
        Firebase.RTDB.setInt(    &fbdo, "/sensores/dht11_interno/humidade",      (int)hum);
        Firebase.RTDB.setString(&fbdo, "/sensores/dht11_interno/ultima_leitura", getTimestamp());
        Firebase.RTDB.setString(&fbdo, "/sensores/dht11_interno/status",        "ok");
    }
    verificarAlertasAmbientais(temp, hum);
}

void verificarAlertasAmbientais(float temp, float hum) {
    float tempMax = 30.0, tempMin = 10.0, humMax = 65.0;
    if (Firebase.RTDB.getFloat(&fbdo, "/config/temp_max"))    tempMax = fbdo.to<float>();
    if (Firebase.RTDB.getFloat(&fbdo, "/config/temp_min"))    tempMin = fbdo.to<float>();
    if (Firebase.RTDB.getFloat(&fbdo, "/config/humidade_max")) humMax = fbdo.to<float>();
    Firebase.RTDB.setBool(&fbdo, "/alertas/temperatura_excedida", (temp > tempMax || temp < tempMin);
    Firebase.RTDB.setBool(&fbdo, "/alertas/humidade_excedida",   (hum > humMax));
}

// =====
// SETUP
// =====
void setup() {
    Serial.begin(115200);
    delay(3000);
    Serial.println("[BOOT] MedBox FASE 4 - ESP32-C6");

    pinMode(PINO_BUZZER, OUTPUT);
    digitalWrite(PINO_BUZZER, LOW);

    for (int i = 0; i < NUM_SERVOS; i++) {
        if (i < SERVO_CONECTADOS) {
            servos[i].attach(PINOS_SERVOS[i], 500, 2400);
            servos[i].write(0);
            delay(600);
            Serial.printf("[SERVO %d] GPIO %d - activo, posicao 0\n", i+1, PINOS_SERVOS[i]);
        } else {
            Serial.printf("[SERVO %d] GPIO %d - reservado (software placeholder)\n", i+1, PINOS_SERVOS[i]);
        }
    }

    scale.begin(PINO_HX711_DOUT, PINO_HX711_SCK);
    delay(500);
    if (scale.is_ready()) {
        scale.set_scale(CALIBRATION_FACTOR);
        scale.tare();
        Serial.println("[HX711] Activa (modo PESO, calibrada em gramas).");
    } else {
        Serial.println("[HX711] ERRO: sensor nao detectado.");
    }
}

```

```

Wire.begin(8, 9);
if (rtc.begin(&Wire)) Serial.println("[RTC] DS3231 sincronizado.");
else Serial.println("[RTC] ERRO: DS3231 nao detectado.");

dht.begin();

Serial.println("[WIFI] A ligar...");
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
int tries = 0;
while (WiFi.status() != WL_CONNECTED && tries < 40) { delay(250); tries++; }

if (WiFi.status() != WL_CONNECTED) {
    Serial.println("[WIFI] Timeout - modo offline (cache + RTC activos).");
    return;
}

Serial.println("[WIFI] Conectado. A sincronizar NTP...");
configTime(0, 0, ntpServer);
setenv("TZ", fusoHorario, 1);
tzset();

config.api_key      = API_KEY;
config.database_url = DATABASE_URL;
auth.user.email     = "idoso@dispensador.pt";
auth.user.password  = "tese123456";
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);

tries = 0;
while (!Firebase.ready() && tries < 30) { delay(500); tries++; }
if (Firebase.ready()) {
    Serial.println("[FIREBASE] Conectado.");
    verificarResetDiario();
}
}

// =====
// LOOP
// =====
void loop() {
    unsigned long agora = millis();

    if (agora - tAnteriorDispensa >= INTERVALO_DISPENSA) {
        tAnteriorDispensa = agora;
        verificarDispensa();
    }

    if (agora - tAnteriorComando >= INTERVALO_COMANDO) {
        tAnteriorComando = agora;
        verificarComandoManual();
    }
}

```

```

    }
    static unsigned long tAnteriorReset = 0;
    if (Firebase.ready() && agora - tAnteriorReset >= 3600000UL) {
        tAnteriorReset = agora;
        verificarResetDiario();
    }
    if (Firebase.ready() && agora - tAnteriorSensores >= INTERVALO_SENSORES) {
        tAnteriorSensores = agora;
        lerEPublicarDHT11();
    }
}
}

```

D.5 Example Drop Detection Log (Ben-u-ron 1000 mg)

Raw Serial Monitor output for the two example dispensing cycles discussed in Section 4.5.2, obtained with the sketch in Appendix D.2.

```

20:44:17.826 -> A rodar o cilindro (dispensa)...
20:44:20.121 -> Leituras de peso apos a dispensa (g):
20:44:20.121 -> #01 -> 1.173 g
20:44:20.196 -> #02 -> 1.231 g
20:44:20.327 -> #03 -> 1.194 g
20:44:20.405 -> #04 -> 1.228 g
20:44:20.536 -> #05 -> 1.200 g
20:44:20.612 -> #06 -> 1.166 g
20:44:20.696 -> #07 -> 1.198 g
20:44:20.825 -> #08 -> 1.191 g
20:44:20.915 -> #09 -> 1.228 g
20:44:20.996 -> #10 -> 1.164 g
20:44:21.124 -> #11 -> 1.204 g
20:44:21.216 -> #12 -> 1.196 g
20:44:21.295 -> #13 -> 1.239 g
20:44:21.433 -> #14 -> 1.218 g
20:44:21.509 -> #15 -> 1.148 g
20:44:21.631 -> #16 -> 9.323 g
20:44:21.696 -> #17 -> 1.180 g
20:44:21.832 -> #18 -> 1.232 g
20:44:21.921 -> #19 -> 1.161 g
20:44:21.996 -> #20 -> 1.224 g
20:44:22.124 -> #21 -> 1.204 g
20:44:22.213 -> #22 -> 1.223 g
20:44:22.295 -> #23 -> 1.196 g
20:44:22.418 -> #24 -> 1.160 g
20:44:22.495 -> #25 -> 1.200 g
20:44:22.628 -> #26 -> 1.163 g
20:44:22.716 -> #27 -> 1.214 g
20:44:22.795 -> #28 -> 1.161 g
20:44:22.927 -> #29 -> 1.231 g
20:44:23.018 -> #30 -> 1.187 g
20:44:23.098 -> Fim do registo. Envia qualquer tecla para repetir.
20:44:26.365 ->
20:44:26.365 -> A tarar a plataforma antes da dispensa...
20:44:28.038 -> A rodar o cilindro (dispensa)...

```

```

20:44:30.355 -> Leituras de peso apos a dispensa (g):
20:44:30.355 -> #01 -> 1.239 g
20:44:30.475 -> #02 -> 1.171 g
20:44:30.566 -> #03 -> 1.202 g
20:44:30.647 -> #04 -> 1.231 g
20:44:30.773 -> #05 -> 1.184 g
20:44:30.848 -> #06 -> 1.246 g
20:44:30.982 -> #07 -> 1.196 g
20:44:31.073 -> #08 -> 1.230 g
20:44:31.147 -> #09 -> 1.181 g
20:44:31.276 -> #10 -> 1.223 g
20:44:31.360 -> #11 -> 1.193 g
20:44:31.484 -> #12 -> 1.218 g
20:44:31.568 -> #13 -> 1.221 g
20:44:31.647 -> #14 -> 1.212 g
20:44:31.784 -> #15 -> 1.248 g
20:44:31.873 -> #16 -> 1.195 g
20:44:31.947 -> #17 -> 1.235 g
20:44:32.080 -> #18 -> 1.158 g
20:44:32.168 -> #19 -> 1.215 g
20:44:32.247 -> #20 -> 1.182 g
20:44:32.373 -> #21 -> 1.649 g
20:44:32.448 -> #22 -> 1.251 g
20:44:32.583 -> #23 -> 1.195 g
20:44:32.675 -> #24 -> 1.230 g
20:44:32.747 -> #25 -> 1.193 g
20:44:32.881 -> #26 -> 1.237 g
20:44:32.965 -> #27 -> 1.150 g
20:44:33.047 -> #28 -> 1.219 g
20:44:33.174 -> #29 -> 1.157 g
20:44:33.263 -> #30 -> 1.212 g
20:44:33.349 -> Fim do registo. Envia qualquer tecla para repetir.

```

D.6 End-to-End Grouped Dispensing Log

Raw Serial Monitor output for the two end-to-end sessions discussed in Section 4.7.2: three medications scheduled at the same time, and two units of the same medication in one event.

```

19:19:00.459 -> [DISPENSA] 3 dose(s): Amoxicilina + Ácido Clavulânico Bluepharma 875 mg + 125 mg, Ben-U-Ron 1
19:19:02.079 -> [BASE] A verificar estabilidade antes da dispensa:
19:19:02.169 -> #1 -> -0.021g
19:19:02.386 -> #2 -> -0.013g
19:19:02.640 -> #3 -> -0.015g
19:19:02.895 -> #4 -> -0.021g
19:19:03.066 -> [BASE] Media=-0.018g | Variacao=0.008g
19:19:03.066 -> [JAM] Compartimento 3: a agitar cilindro (3 vaivens, 0-15 graus)
19:19:04.418 -> [JAM] Agitacao concluida, cilindro de volta a 0.
19:19:04.418 -> [SERVO] Compartimento 3: 0 -> 180 graus (5ms/grau)
19:19:05.431 -> [QUEDA 94ms] base=-0.018g | atual=-0.007g | delta=0.010g | threshold=0.300g | estavel=0/5
19:19:05.795 -> [QUEDA 485ms] base=-0.018g | atual=-0.007g | delta=0.011g | threshold=0.300g | estavel=0/5
19:19:06.216 -> [QUEDA 876ms] base=-0.018g | atual=-0.004g | delta=0.014g | threshold=0.300g | estavel=0/5
19:19:06.594 -> [QUEDA 1268ms] base=-0.018g | atual=7.825g | delta=7.842g | threshold=0.300g | estavel=0/5
19:19:07.011 -> [QUEDA 1659ms] base=-0.018g | atual=0.603g | delta=0.621g | threshold=0.300g | estavel=1/5
19:19:07.361 -> [QUEDA 2051ms] base=-0.018g | atual=0.590g | delta=0.607g | threshold=0.300g | estavel=2/5
19:19:07.756 -> [QUEDA 2442ms] base=-0.018g | atual=0.599g | delta=0.617g | threshold=0.300g | estavel=3/5

```

```
19:19:08.180 -> [QUEDA 2834ms] base=-0.018g | atual=0.597g | delta=0.614g | threshold=0.300g | estavel=4/5
19:19:08.180 -> [SERVO] Compartimento 3: 180 -> 0 graus (5ms/grau)
19:19:11.072 -> [BASE] A verificar estabilidade antes da dispensa:
19:19:11.161 -> #1 -> 0.596g
19:19:11.378 -> #2 -> 0.604g
19:19:11.636 -> #3 -> 0.583g
19:19:11.888 -> #4 -> 0.590g
19:19:12.017 -> [BASE] Media=0.593g | Variacao=0.021g
19:19:12.017 -> [JAM] Compartimento 1: a agitar cilindro (3 vaivens, 0-15 graus)
19:19:13.435 -> [JAM] Agitacao concluida, cilindro de volta a 0.
19:19:13.435 -> [SERVO] Compartimento 1: 0 -> 180 graus (5ms/grau)
19:19:14.400 -> [QUEDA 94ms] base=0.593g | atual=0.608g | delta=0.015g | threshold=0.300g | estavel=0/5
19:19:14.788 -> [QUEDA 486ms] base=0.593g | atual=0.595g | delta=0.002g | threshold=0.300g | estavel=0/5
19:19:15.216 -> [QUEDA 877ms] base=0.593g | atual=0.575g | delta=-0.018g | threshold=0.300g | estavel=0/5
19:19:15.603 -> [QUEDA 1269ms] base=0.593g | atual=0.579g | delta=-0.015g | threshold=0.300g | estavel=0/5
19:19:15.990 -> [QUEDA 1660ms] base=0.593g | atual=0.580g | delta=-0.013g | threshold=0.300g | estavel=0/5
19:19:16.396 -> [QUEDA 2052ms] base=0.593g | atual=0.591g | delta=-0.002g | threshold=0.300g | estavel=0/5
19:19:16.784 -> [QUEDA 2443ms] base=0.593g | atual=1.179g | delta=0.586g | threshold=0.300g | estavel=0/5
19:19:17.162 -> [QUEDA 2835ms] base=0.593g | atual=1.185g | delta=0.592g | threshold=0.300g | estavel=1/5
19:19:17.554 -> [QUEDA 3226ms] base=0.593g | atual=1.179g | delta=0.586g | threshold=0.300g | estavel=2/5
19:19:17.932 -> [QUEDA 3618ms] base=0.593g | atual=1.193g | delta=0.600g | threshold=0.300g | estavel=3/5
19:19:18.319 -> [QUEDA 4009ms] base=0.593g | atual=1.187g | delta=0.594g | threshold=0.300g | estavel=4/5
19:19:18.319 -> [SERVO] Compartimento 1: 180 -> 0 graus (5ms/grau)
19:19:21.253 -> [BASE] A verificar estabilidade antes da dispensa:
19:19:21.334 -> #1 -> 1.199g
19:19:21.546 -> #2 -> 1.195g
19:19:21.832 -> #3 -> 1.203g
19:19:22.033 -> #4 -> 1.186g
19:19:22.204 -> [BASE] Media=1.196g | Variacao=0.017g
19:19:22.204 -> [JAM] Compartimento 2: a agitar cilindro (3 vaivens, 0-15 graus)
19:19:23.565 -> [JAM] Agitacao concluida, cilindro de volta a 0.
19:19:23.565 -> [SERVO] Compartimento 2: 0 -> 180 graus (5ms/grau)
19:19:24.565 -> [QUEDA 95ms] base=1.196g | atual=1.186g | delta=-0.009g | threshold=0.300g | estavel=0/5
19:19:24.979 -> [QUEDA 487ms] base=1.196g | atual=2.399g | delta=1.204g | threshold=0.300g | estavel=0/5
19:19:25.372 -> [QUEDA 878ms] base=1.196g | atual=2.384g | delta=1.189g | threshold=0.300g | estavel=1/5
19:19:25.761 -> [QUEDA 1270ms] base=1.196g | atual=2.391g | delta=1.196g | threshold=0.300g | estavel=2/5
19:19:26.148 -> [QUEDA 1661ms] base=1.196g | atual=2.398g | delta=1.202g | threshold=0.300g | estavel=3/5
19:19:26.542 -> [QUEDA 2053ms] base=1.196g | atual=2.399g | delta=1.203g | threshold=0.300g | estavel=4/5
19:19:26.542 -> [SERVO] Compartimento 2: 180 -> 0 graus (5ms/grau)
19:19:27.900 -> [RECOLHA PESO] t=0s | peso=2.388g | threshold=+-0.150g | estavel=0/10
19:19:28.318 -> [RECOLHA PESO] t=0s | peso=2.382g | threshold=+-0.150g | estavel=0/10
19:19:28.716 -> [RECOLHA PESO] t=0s | peso=2.383g | threshold=+-0.150g | estavel=0/10
19:19:29.112 -> [RECOLHA PESO] t=1s | peso=2.396g | threshold=+-0.150g | estavel=0/10
19:19:29.465 -> [RECOLHA PESO] t=1s | peso=2.384g | threshold=+-0.150g | estavel=0/10
19:19:29.895 -> [RECOLHA PESO] t=2s | peso=2.391g | threshold=+-0.150g | estavel=0/10
19:19:30.286 -> [RECOLHA PESO] t=2s | peso=2.396g | threshold=+-0.150g | estavel=0/10
19:19:30.680 -> [RECOLHA PESO] t=2s | peso=2.388g | threshold=+-0.150g | estavel=0/10
19:19:31.051 -> [RECOLHA PESO] t=3s | peso=2.423g | threshold=+-0.150g | estavel=0/10
19:19:31.422 -> [RECOLHA PESO] t=3s | peso=2.430g | threshold=+-0.150g | estavel=0/10
19:19:31.814 -> [RECOLHA PESO] t=4s | peso=2.430g | threshold=+-0.150g | estavel=0/10
19:19:32.248 -> [RECOLHA PESO] t=4s | peso=2.389g | threshold=+-0.150g | estavel=0/10
19:19:32.639 -> [RECOLHA PESO] t=4s | peso=2.397g | threshold=+-0.150g | estavel=0/10
19:19:33.030 -> [RECOLHA PESO] t=5s | peso=2.398g | threshold=+-0.150g | estavel=0/10
19:19:33.207 -> [BUZZER] 1 bipes
19:19:33.594 -> [RECOLHA PESO] t=5s | peso=2.393g | threshold=+-0.150g | estavel=0/10
19:19:33.972 -> [RECOLHA PESO] t=6s | peso=2.400g | threshold=+-0.150g | estavel=0/10
19:19:34.392 -> [RECOLHA PESO] t=6s | peso=2.397g | threshold=+-0.150g | estavel=0/10
```

```
19:19:34.778 -> [RECOLHA PESO] t=6s | peso=2.397g | threshold=+-0.150g | estavel=0/10
19:19:35.147 -> [RECOLHA PESO] t=7s | peso=2.398g | threshold=+-0.150g | estavel=0/10
19:19:35.564 -> [RECOLHA PESO] t=7s | peso=2.408g | threshold=+-0.150g | estavel=0/10
19:19:35.946 -> [RECOLHA PESO] t=8s | peso=9.685g | threshold=+-0.150g | estavel=0/10
19:19:36.365 -> [RECOLHA PESO] t=8s | peso=16.355g | threshold=+-0.150g | estavel=0/10
19:19:36.712 -> [RECOLHA PESO] t=8s | peso=14.578g | threshold=+-0.150g | estavel=0/10
19:19:37.121 -> [RECOLHA PESO] t=9s | peso=1.195g | threshold=+-0.150g | estavel=0/10
19:19:37.522 -> [RECOLHA PESO] t=9s | peso=1.210g | threshold=+-0.150g | estavel=0/10
19:19:37.917 -> [RECOLHA PESO] t=10s | peso=1.211g | threshold=+-0.150g | estavel=0/10
19:19:38.297 -> [RECOLHA PESO] t=10s | peso=1.223g | threshold=+-0.150g | estavel=0/10
19:19:38.515 -> [BUZZER] 1 bipes
19:19:38.885 -> [RECOLHA PESO] t=11s | peso=1.222g | threshold=+-0.150g | estavel=0/10
19:19:39.305 -> [RECOLHA PESO] t=11s | peso=1.220g | threshold=+-0.150g | estavel=0/10
19:19:39.653 -> [RECOLHA PESO] t=11s | peso=1.226g | threshold=+-0.150g | estavel=0/10
19:19:40.066 -> [RECOLHA PESO] t=12s | peso=1.198g | threshold=+-0.150g | estavel=0/10
19:19:40.439 -> [RECOLHA PESO] t=12s | peso=1.199g | threshold=+-0.150g | estavel=0/10
19:19:40.828 -> [RECOLHA PESO] t=13s | peso=3.620g | threshold=+-0.150g | estavel=0/10
19:19:41.241 -> [RECOLHA PESO] t=13s | peso=20.769g | threshold=+-0.150g | estavel=0/10
19:19:41.633 -> [RECOLHA PESO] t=13s | peso=12.538g | threshold=+-0.150g | estavel=0/10
19:19:42.019 -> [RECOLHA PESO] t=14s | peso=7.008g | threshold=+-0.150g | estavel=0/10
19:19:42.394 -> [RECOLHA PESO] t=14s | peso=-1.559g | threshold=+-0.150g | estavel=0/10
19:19:42.808 -> [RECOLHA PESO] t=14s | peso=0.594g | threshold=+-0.150g | estavel=0/10
19:19:43.179 -> [RECOLHA PESO] t=15s | peso=0.605g | threshold=+-0.150g | estavel=0/10
19:19:43.568 -> [RECOLHA PESO] t=15s | peso=0.605g | threshold=+-0.150g | estavel=0/10
19:19:43.781 -> [BUZZER] 1 bipes
19:19:44.203 -> [RECOLHA PESO] t=16s | peso=0.605g | threshold=+-0.150g | estavel=0/10
19:19:44.589 -> [RECOLHA PESO] t=16s | peso=0.590g | threshold=+-0.150g | estavel=0/10
19:19:44.982 -> [RECOLHA PESO] t=17s | peso=0.584g | threshold=+-0.150g | estavel=0/10
19:19:45.379 -> [RECOLHA PESO] t=17s | peso=0.597g | threshold=+-0.150g | estavel=0/10
19:19:45.758 -> [RECOLHA PESO] t=17s | peso=10.126g | threshold=+-0.150g | estavel=0/10
19:19:46.135 -> [RECOLHA PESO] t=18s | peso=13.505g | threshold=+-0.150g | estavel=0/10
19:19:46.509 -> [RECOLHA PESO] t=18s | peso=19.251g | threshold=+-0.150g | estavel=0/10
19:19:46.921 -> [RECOLHA PESO] t=19s | peso=0.590g | threshold=+-0.150g | estavel=0/10
19:19:47.335 -> [RECOLHA PESO] t=19s | peso=11.238g | threshold=+-0.150g | estavel=0/10
19:19:47.715 -> [RECOLHA PESO] t=19s | peso=24.069g | threshold=+-0.150g | estavel=0/10
19:19:48.096 -> [RECOLHA PESO] t=20s | peso=-0.008g | threshold=+-0.150g | estavel=0/10
19:19:48.486 -> [RECOLHA PESO] t=20s | peso=-0.006g | threshold=+-0.150g | estavel=1/10
19:19:48.872 -> [RECOLHA PESO] t=21s | peso=-0.016g | threshold=+-0.150g | estavel=2/10
19:19:49.086 -> [BUZZER] 1 bipes
19:19:49.465 -> [RECOLHA PESO] t=21s | peso=-0.003g | threshold=+-0.150g | estavel=3/10
19:19:49.842 -> [RECOLHA PESO] t=22s | peso=-0.002g | threshold=+-0.150g | estavel=4/10
19:19:50.247 -> [RECOLHA PESO] t=22s | peso=-0.000g | threshold=+-0.150g | estavel=5/10
19:19:50.625 -> [RECOLHA PESO] t=22s | peso=0.023g | threshold=+-0.150g | estavel=6/10
19:19:51.060 -> [RECOLHA PESO] t=23s | peso=0.006g | threshold=+-0.150g | estavel=7/10
19:19:51.419 -> [RECOLHA PESO] t=23s | peso=-0.003g | threshold=+-0.150g | estavel=8/10
19:19:51.805 -> [RECOLHA PESO] t=23s | peso=-0.013g | threshold=+-0.150g | estavel=9/10
19:19:51.805 -> [RECOLHA CONFIRMADA] peso=-0.013g | Plataforma estavel durante 10 leituras. 2026-07-09 19:19:
19:19:56.314 -> [DISPENSA CONCLUIDA] Estado: tomado | Queda: confirmada | Recolha: confirmada

[DISPENSA] 1 dose(s): Ben-U-Ron 1000 mg | Janela: 30 min
19:37:04.944 -> [BASE] A verificar estabilidade antes da dispensa:
19:37:05.035 -> #1 -> 0.002g
19:37:05.285 -> #2 -> 0.005g
19:37:05.506 -> #3 -> 0.001g
19:37:05.769 -> #4 -> 0.009g
19:37:05.892 -> [BASE] Media=0.004g | Variacao=0.008g
```

```
19:37:05.892 -> [JAM] Compartimento 1: a agitar cilindro (3 vaivens, 0-15 graus)
19:37:07.288 -> [JAM] Agitacao concluida, cilindro de volta a 0.
19:37:07.288 -> [SERVO] Compartimento 1: 0 -> 180 graus (5ms/grau)
19:37:08.302 -> [QUEDA 94ms] base=0.004g | atual=0.000g | delta=-0.004g | threshold=0.300g | estavel=0/5
19:37:08.697 -> [QUEDA 486ms] base=0.004g | atual=-0.014g | delta=-0.019g | threshold=0.300g | estavel=0/5
19:37:09.078 -> [QUEDA 877ms] base=0.004g | atual=-0.005g | delta=-0.009g | threshold=0.300g | estavel=0/5
19:37:09.463 -> [QUEDA 1268ms] base=0.004g | atual=1.120g | delta=1.115g | threshold=0.300g | estavel=0/5
19:37:09.831 -> [QUEDA 1660ms] base=0.004g | atual=1.212g | delta=1.208g | threshold=0.300g | estavel=1/5
19:37:10.247 -> [QUEDA 2051ms] base=0.004g | atual=1.203g | delta=1.199g | threshold=0.300g | estavel=2/5
19:37:10.635 -> [QUEDA 2443ms] base=0.004g | atual=1.202g | delta=1.198g | threshold=0.300g | estavel=3/5
19:37:11.017 -> [QUEDA 2834ms] base=0.004g | atual=1.200g | delta=1.196g | threshold=0.300g | estavel=4/5
19:37:11.017 -> [SERVO] Compartimento 1: 180 -> 0 graus (5ms/grau)
19:37:13.914 -> [BASE] A verificar estabilidade antes da dispensa:
19:37:13.997 -> #1 -> 1.203g
19:37:14.281 -> #2 -> 1.203g
19:37:14.493 -> #3 -> 1.202g
19:37:14.755 -> #4 -> 1.195g
19:37:14.920 -> [BASE] Media=1.201g | Variacao=0.008g
19:37:14.920 -> [JAM] Compartimento 1: a agitar cilindro (3 vaivens, 0-15 graus)
19:37:16.280 -> [JAM] Agitacao concluida, cilindro de volta a 0.
19:37:16.280 -> [SERVO] Compartimento 1: 0 -> 180 graus (5ms/grau)
19:37:17.301 -> [QUEDA 95ms] base=1.201g | atual=1.247g | delta=0.046g | threshold=0.300g | estavel=0/5
19:37:17.650 -> [QUEDA 486ms] base=1.201g | atual=1.232g | delta=0.031g | threshold=0.300g | estavel=0/5
19:37:18.071 -> [QUEDA 878ms] base=1.201g | atual=1.257g | delta=0.057g | threshold=0.300g | estavel=0/5
19:37:18.437 -> [QUEDA 1269ms] base=1.201g | atual=1.238g | delta=0.038g | threshold=0.300g | estavel=0/5
19:37:18.862 -> [QUEDA 1661ms] base=1.201g | atual=1.322g | delta=0.122g | threshold=0.300g | estavel=0/5
19:37:19.246 -> [QUEDA 2052ms] base=1.201g | atual=1.313g | delta=0.112g | threshold=0.300g | estavel=0/5
19:37:19.628 -> [QUEDA 2444ms] base=1.201g | atual=1.225g | delta=0.025g | threshold=0.300g | estavel=0/5
19:37:19.999 -> [QUEDA 2835ms] base=1.201g | atual=2.390g | delta=1.189g | threshold=0.300g | estavel=0/5
19:37:20.421 -> [QUEDA 3227ms] base=1.201g | atual=2.401g | delta=1.200g | threshold=0.300g | estavel=1/5
19:37:20.786 -> [QUEDA 3618ms] base=1.201g | atual=2.404g | delta=1.203g | threshold=0.300g | estavel=2/5
19:37:21.203 -> [QUEDA 4010ms] base=1.201g | atual=2.402g | delta=1.201g | threshold=0.300g | estavel=3/5
19:37:21.589 -> [QUEDA 4401ms] base=1.201g | atual=2.398g | delta=1.198g | threshold=0.300g | estavel=4/5
19:37:21.589 -> [SERVO] Compartimento 1: 180 -> 0 graus (5ms/grau)
19:37:23.120 -> [RECOLHA PESO] t=0s | peso=2.522g | threshold=+-0.150g | estavel=0/10
19:37:23.512 -> [RECOLHA PESO] t=0s | peso=2.403g | threshold=+-0.150g | estavel=0/10
19:37:23.903 -> [RECOLHA PESO] t=0s | peso=2.396g | threshold=+-0.150g | estavel=0/10
19:37:24.316 -> [RECOLHA PESO] t=1s | peso=2.393g | threshold=+-0.150g | estavel=0/10
19:37:24.712 -> [RECOLHA PESO] t=1s | peso=2.405g | threshold=+-0.150g | estavel=0/10
19:37:25.110 -> [RECOLHA PESO] t=2s | peso=2.399g | threshold=+-0.150g | estavel=0/10
19:37:25.499 -> [RECOLHA PESO] t=2s | peso=2.401g | threshold=+-0.150g | estavel=0/10
19:37:25.892 -> [RECOLHA PESO] t=2s | peso=2.401g | threshold=+-0.150g | estavel=0/10
19:37:26.290 -> [RECOLHA PESO] t=3s | peso=2.396g | threshold=+-0.150g | estavel=0/10
19:37:26.687 -> [RECOLHA PESO] t=3s | peso=2.398g | threshold=+-0.150g | estavel=0/10
19:37:27.074 -> [RECOLHA PESO] t=4s | peso=2.406g | threshold=+-0.150g | estavel=0/10
19:37:27.439 -> [RECOLHA PESO] t=4s | peso=2.403g | threshold=+-0.150g | estavel=0/10
19:37:27.818 -> [RECOLHA PESO] t=4s | peso=2.397g | threshold=+-0.150g | estavel=0/10
19:37:28.243 -> [RECOLHA PESO] t=5s | peso=2.746g | threshold=+-0.150g | estavel=0/10
19:37:28.408 -> [BUZZER] 1 bipes
19:37:28.833 -> [RECOLHA PESO] t=5s | peso=31.713g | threshold=+-0.150g | estavel=0/10
19:37:29.215 -> [RECOLHA PESO] t=6s | peso=24.960g | threshold=+-0.150g | estavel=0/10
19:37:29.606 -> [RECOLHA PESO] t=6s | peso=11.851g | threshold=+-0.150g | estavel=0/10
19:37:29.990 -> [RECOLHA PESO] t=6s | peso=0.009g | threshold=+-0.150g | estavel=0/10
19:37:30.402 -> [RECOLHA PESO] t=7s | peso=0.015g | threshold=+-0.150g | estavel=1/10
19:37:30.790 -> [RECOLHA PESO] t=7s | peso=0.002g | threshold=+-0.150g | estavel=2/10
19:37:31.182 -> [RECOLHA PESO] t=8s | peso=0.000g | threshold=+-0.150g | estavel=3/10
19:37:31.567 -> [RECOLHA PESO] t=8s | peso=0.007g | threshold=+-0.150g | estavel=4/10
```



```
19:37:31.963 -> [RECOLHA PESO] t=8s | peso=0.006g | threshold=+-0.150g | estavel=5/10
19:37:32.325 -> [RECOLHA PESO] t=9s | peso=0.010g | threshold=+-0.150g | estavel=6/10
19:37:32.753 -> [RECOLHA PESO] t=9s | peso=0.005g | threshold=+-0.150g | estavel=7/10
19:37:33.141 -> [RECOLHA PESO] t=10s | peso=-0.006g | threshold=+-0.150g | estavel=8/10
19:37:33.515 -> [RECOLHA PESO] t=10s | peso=-0.000g | threshold=+-0.150g | estavel=9/10
19:37:33.515 -> [RECOLHA CONFIRMADA] peso=-0.000g | Plataforma estavel durante 10 leituras. 2026-07-09 19:37:
19:37:35.795 -> [DISPENSA CONCLUIDA] Estado: tomado | Queda: confirmada | Recolha: confirmada
```