



**GERMAN-GEORGIAN STUDENT CONFERENCE
ON APPLIED INFORMATICS, MATHEMATICS
AND ENGINEERING
(GGSC 2026)**

Conference Proceedings

October 10, 2026

Georgian Technical University (GTU), Tbilisi, Georgia

Organized jointly by

Georgian Technical University (GTU)

and

University of Applied Sciences Dresden (HTW Dresden)

Participants:

Students of GTU and HTW Dresden

Breathing Stresslower

Anri Aidiniani and Kristine Tatevosiani

Georgian Technical University, Tbilisi, Georgia

Abstract—In the modern world, chronic stress and anxiety constitute a significant global issue, negatively affecting the population's psycho-emotional and physical health. Statistical data indicate that approximately 37% of the global population experiences stress daily, while 39% experiences anxiety. Chronic stress leads to sleep disturbances, high blood pressure, cardiovascular diseases, a weakened immune system, and impaired memory and concentration. Furthermore, 46% of people resort to unhealthy behaviors when under stress. In response to this challenge, the presented project aims to develop a simple, effective, portable, and affordable device that helps users monitor stress levels in real-time and restore emotional balance.

The device's system architecture is built upon the energy-efficient ESP32-C3 microcontroller. The system comprises a sensor unit, a communication interface, and a lithium-ion (Li-Ion) battery with a TP4056 charging module. To comprehensively assess physiological stress indicators, the system utilizes a CJMCU-6701 (GSR) sensor—which measures skin electrical conductance (up to 10 M Ω)—and a MAX30102 optical sensor for pulse monitoring (30–200 beats per minute). Based on the processing of analog and digital signals from the sensors, the system determines the stress level and displays appropriate breathing exercise techniques on a low-power OLED (SSD1306) screen. Additionally, an integrated vibration motor uses mechanical vibrations to signal the optimal breathing rhythm to the user.

Compared to existing market alternatives (such as Mendi or Apollo Neuro)—which are characterized by high prices and additional annual subscription fees—the proposed device offers a budget-friendly, compact, convenient, and autonomous alternative. Future development prospects for the project include housing the device in an ergonomic casing, enabling wireless data transmission via a Wi-Fi/Bluetooth module, and allowing for the customization of individual settings. The presented project will make a significant contribution to improving public health and advancing technologies for daily stress management.

Index Terms—Anxiety reduction, biofeedback, ESP32-C3, GSR sensor, MAX30102, portable device, stress management.

I. INTRODUCTION

Chronic stress and anxiety affect over 37% and 39% of the global population daily, leading to cardiovascular issues, sleep disturbances, and impaired cognitive functions. While traditional diagnostic methods rely on subjective self-reporting or clinical settings, wearable biomedical technologies provide a promising route for continuous, non-invasive stress management. The primary challenge in consumer-grade

biofeedback systems is balancing physiological accuracy, energy efficiency, and affordability. Existing high-end market alternatives (e.g., Apollo Neuro or Mendi) suffer from high retail costs and mandatory subscription plans, making them inaccessible for broad public use. This paper presents the design and implementation of *Breathing Stresslower*, an autonomous, portable device that bridges this gap by offering real-time multimodal stress detection and tactile/visual feedback for emotional regulation.

II. RELATED WORK

Wearable physiological monitoring generally relies on electrodermal activity (EDA/GSR) and photoplethysmography (PPG). Commercial devices like Mendi utilize functional near-infrared spectroscopy (fNIRS), whereas Apollo Neuro uses localized haptic stimulation. While effective, these commercial systems are restricted by closed ecosystems, high hardware overhead, and lack of direct multi-sensor integration at budget-friendly price points. In academic literature, combining GSR (which reflects sympathetic nervous system arousal) with Heart Rate Variability (HRV) derived from PPG signals is established as a reliable dual-metric approach for autonomic nervous system (ANS) monitoring. Our work builds upon these biomedical principles to construct an open, low-power, integrated hardware platform.

III. PROPOSED METHOD

The proposed *Breathing Stresslower* architecture utilizes a dual-parameter sensing approach to quantify sympathetic nervous system activation:

- 1) **Electrodermal Activity Monitoring:** Captures rapid variations in skin conductance resulting from sweat gland activity during psychological stress.
- 2) **Optical Photoplethysmography:** Measures peripheral blood volume pulse to extract real-time heart rate (HR).

The system continuously processes analog skin conductance data and digital optical signals to categorize stress into baseline levels. Once stress exceeds a pre-defined threshold, the device initiates a biofeedback loop: visual instructions (paced breathing patterns) are displayed on an OLED screen while an integrated haptic motor provides mechanical vibrations to dictate optimal inhalation and exhalation rhythms without requiring constant visual focus.

IV. SYSTEM ARCHITECTURE AND IMPLEMENTATION

Hardware:

- ESP32-C3 microcontroller (main processing unit with built-in ADC and I²C support);
- CJMCU-6701 Galvanic Skin Response (GSR) sensor connected to analog pin (GPIO 0) for electrodermal activity monitoring;
- 0.96" OLED display (128×64, SSD1306) driven via I²C interface (SDA: GPIO 5, SCL: GPIO 6) for visual output;
- Vibration motor connected to digital output (GPIO 1) for tactile haptic feedback during breathing phases;
- Push button with internal pull-up resistor (GPIO 2) for manual exercise trigger and state switching;
- Power supply unit (3.3 V/5 V regulated power source).

Software:

- Real-time analog biosignal acquisition and thresholding for stress level detection;
- Software debouncing algorithm for reliable tactile button input handling;
- Non-blocking multi-phase timer engine (4-second interval cycles for Inhale, Hold, and Exhale sequence);
- PWM/digital modulation driver for phase-specific haptic vibration patterns (continuous, silence, and pulsed output);
- Dynamic graphical UI rendering on the OLED display with high-frequency updates (10 Hz) for real-time visual pacing and system state display.

```

1 #include <Wire.h>
2 #include <Adafruit_GFX.h>
3 #include <Adafruit_SSD1306.h>
4 #include "MAX30105.h"
5
6
7 #define SCREEN_WIDTH 128
8 #define SCREEN_HEIGHT 64
9 #define OLED_RESET -1
10 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
11
12 MAX30105 particleSensor;
13
14
15 #define I2C_SDA_PIN 5
16 #define I2C_SCL_PIN 6
17 #define GSR_PIN 0
18 #define VIBRO_PIN 1

```

V. CONCLUSION

This article presents a low-cost stress reliever. Experimental testing of the prototype confirmed its functionality, high sensitivity to micro-changes in the autonomic nervous system, and accurate visualization of the obtained data.

REFERENCES

- [1] “რა არის სტრესი და როგორ დავძლიოთ ის?” Aversi. [Online]. Available: <https://www.aversi.ge/ka/cnobar/1161/ra-aris-stresi-da-rogor-davdzliot-is->
- [2] “Stress (biology),” Wikipedia. [Online]. Available: [https://en.wikipedia.org/wiki/Stress_\(biology\)](https://en.wikipedia.org/wiki/Stress_(biology))
- [3] “Top 7 biofeedback devices for anxiety and stress relief,” Mendi.io. [Online]. Available: <https://www.mendi.io/blogs/brain-health/top-7-biofeedback-devices-for-anxiety-and-stress-relief>
- [4] “Stress: statistics,” Mental Health Foundation. [Online]. Available: <https://www.mentalhealth.org.uk/explore-mental-health/statistics/stress-statistics>

- [5] “Stress: what it does to the body,” Healthline. [Online]. Available: <https://www.healthline.com/health/stress/effects-on-body>

Is Async Truly Zero-Cost? Benchmarking Latency and Jitter in `no_std` Rust vs. FreeRTOS on ARM Cortex-M

Nikoloz Akhalaia

Georgian Technical University, Tbilisi, Georgia

Abstract—This paper compares a Rust asynchronous embedded runtime based on Embassy with a C implementation based on FreeRTOS on the same STM32F411CEU6 microcontroller. The experiment fixes the system clock at 16 MHz and collects 1,000 samples for timestamp overhead, CRC-32 processing, and a two-task request/reply round trip. Cycle counts are measured with the Cortex-M4 DWT cycle counter, and logging is performed only after measurement. Rust/Embassy produced a median round-trip latency of 760 cycles (47.50 us), while FreeRTOS produced 1,077 cycles (67.31 us). The FreeRTOS results also showed a wider latency distribution, with a 95th percentile of 1,825 cycles. These results show that the tested Embassy design had lower and more deterministic synchronization latency in this controlled workload. The study does not claim that Rust is generally faster than C; it compares two specific runtime implementations and configurations.

Index Terms—async embedded systems, Embassy, FreeRTOS, Rust, STM32

I. INTRODUCTION

Embedded software must respond to events with low and predictable latency while using limited processor time and memory. C with a real-time operating system remains a common design choice, but Rust is increasingly used for bare-metal firmware because it provides low-level control together with compile-time memory-safety mechanisms [1]. Rust can also run without the standard library in a `no_std` environment, which makes it suitable for microcontrollers [2].

This work studies a narrower question: how do two practical concurrency models behave when they perform the same small request/reply workload on the same microcontroller? The comparison uses Embassy's asynchronous executor for Rust and FreeRTOS task notifications for C. Embassy uses statically allocated async tasks and does not require a heap [3]. FreeRTOS task notifications send events directly to tasks and are designed as a lightweight synchronization mechanism [4].

The contribution of this paper is a controlled cycle-level microbenchmark on an STM32F411CEU6. Both implementations use a 16 MHz HSI clock, the same 256-byte CRC-32 workload, a 100-iteration warm-up, and 1,000 measured iterations. The study reports the full latency distribution rather than only a single average. The goal is not to rank programming languages in general, but to measure the cost and timing behavior of these two concrete runtime designs under matched hardware conditions.

II. RELATED WORK AND BACKGROUND

The STM32F411CEU6 is based on an Arm Cortex-M4 core and provides up to 512 KB of Flash and 128 KB of SRAM [5]. The Cortex-M family provides hardware support useful for low-level performance measurement, including the Data Watchpoint and Trace cycle counter used in this experiment. Running the processor at a fixed clock allows measured cycle counts to be converted directly into time.

FreeRTOS is a preemptive real-time kernel widely used on microcontrollers. Its direct-to-task notification mechanism can unblock a task without creating a separate queue or semaphore object. FreeRTOS documentation describes task notifications as a lighter and faster alternative to several communication objects [4], [6]. For the controlled test in this paper, preemption is enabled, static task allocation is used, and equal-priority time slicing is disabled.

Embassy is an embedded Rust framework built around `async/await`. Its executor statically allocates tasks and wakes only tasks that have work to do [3]. This model differs from a traditional RTOS task scheduler: asynchronous tasks cooperatively return control at await points, while an RTOS schedules independent task stacks. Because these execution models are different, the experiment focuses on observable request/reply latency rather than claiming that the internal operations are identical.

III. EXPERIMENTAL METHOD

A Black Pill development board with an STM32F411CEU6 was used for both implementations. The processor was configured to use the internal HSI oscillator at 16 MHz with AHB, APB1, and APB2 dividers set to 1. The Rust run recorded `rcc_cfgr = 0` and `flash_acr = 0`. The FreeRTOS code checked the same clock assumptions before starting the benchmark. Both firmware images were built using optimized release configurations.

Parameter	Rust	FreeRTOS
MCU	STM32F411CEU6	STM32F411CEU6
CPU clock	16 MHz HSI	16 MHz HSI
Samples	1,000	1,000
Warm-up	100 iterations	100 iterations
CRC input	256 bytes	256 bytes
Timing source	DWT CYCNT	DWT CYCNT
Concurrency	Async tasks	RTOS tasks
Synchronization	Async signal/request-reply	Task notifications

TABLE I. CONTROLLED EXPERIMENT CONFIGURATION

Each measured sample contains three values. First, an empty timestamp measurement estimates the overhead of the timing sequence itself. Second, CRC-32 is calculated over a fixed 256-byte array. Third, a request/reply round trip is measured between two execution contexts. The worker receives a request and immediately signals the controller back. The CRC result is checked against a known value to prevent invalid or optimized-away work.

The DWT cycle counter is read with compiler barriers and instruction/data synchronization barriers around the timestamp operation. All 1,000 samples are stored before results are emitted. This avoids serial output or debugger logging inside the timed region. FreeRTOS results were copied from RAM only after a completion flag was set. The Rust program similarly reported results after collection.

For descriptive statistics, this paper reports the median, mean, standard deviation, minimum, 95th percentile, 99th percentile,

and maximum. Median values are emphasized because the FreeRTOS distribution contains occasional high-latency observations. Time in microseconds is calculated from cycles / 16 MHz.

IV. RESULTS AND EVALUATION

Runtime	Median	Mean	SD	Min	P95	P99	Max
Rust	760	760.00	0.00	760	760	760	760
FreeRTOS	1077	1140.76	208.00	1077	1825	1927	1932

TABLE II. ROUND-TRIP LATENCY IN CPU CYCLES (1,000 SAMPLES)

Rust/Embassy produced a round-trip value of 760 cycles for every measured sample. At 16 MHz, this is 47.50 us. FreeRTOS had a median of 1,077 cycles (67.31 us) and a mean of 1,140.76 cycles (71.30 us). Its 95th percentile was 1,825 cycles (114.06 us), and its maximum was 1,932 cycles (120.75 us). The median FreeRTOS round trip was therefore 317 cycles, or 19.81 us, higher than the Rust/Embassy median in this experiment.

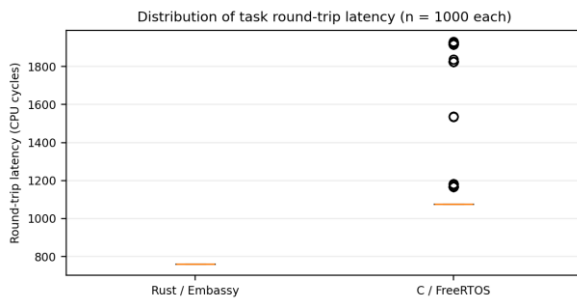


Fig. 1. Round-trip latency distribution for both implementations.

The median ratio is 1.417, meaning that the measured FreeRTOS median was about 1.42 times the Rust/Embassy median. After subtracting each implementation's median empty-timestamp cost, the adjusted values are 754 cycles for Rust/Embassy and 1,064 cycles for FreeRTOS. The adjusted ratio remains similar at 1.41. This suggests that the difference is not explained by the timestamp operation alone.

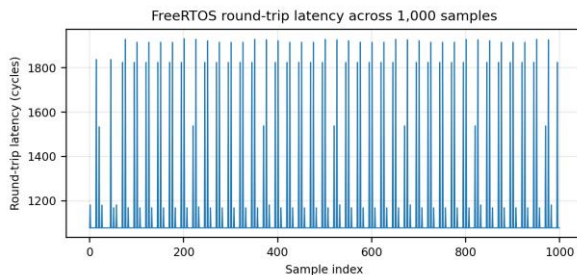


Fig. 2. FreeRTOS round-trip latency by sample index.

Runtime	Metric	Median	Mean	P95	Max
Rust	Empty	6	6.00	6	6
Rust	Crc	7647	7647.00	7647	7647
FreeRTOS	Empty	13	13.76	13	774
FreeRTOS	Crc	17286	17356.65	18049	18152

TABLE III. TIMESTAMP AND CRC-32 MEASUREMENTS IN CPU CYCLES

The CRC-32 measurement also differed strongly: the Rust median was 7,647 cycles, while the FreeRTOS median was 17,286 cycles. This result should not be interpreted as RTOS scheduling overhead because the CRC section does not perform a

task switch. It instead reflects differences in the generated code, CRC implementation, compiler/toolchain, and surrounding firmware. For this reason, the CRC result is useful as a computational baseline but is not used as direct evidence about scheduler performance.

The FreeRTOS results show more timing variation than the Rust run. Most FreeRTOS round trips are close to the 1,077-cycle mode, but some samples rise above 1,800 cycles. Periodic kernel tick interrupts and other runtime activity are plausible sources of this variation, although the current experiment does not isolate each source. The Rust run was completely constant for all three recorded measurements, which indicates high determinism under this specific test configuration.

V. DISCUSSION AND LIMITATIONS

The experiment supports a limited but useful conclusion: for this two-context request/reply microbenchmark on the tested STM32F411 configuration, the Embassy implementation had lower median latency and lower observed jitter than the FreeRTOS implementation. This result is consistent with Embassy's event-driven executor design, where only woken tasks are polled [3], while FreeRTOS performs task scheduling and context switching between independent task stacks.

Several limitations are important. First, this is a microbenchmark, not a complete embedded application. Real systems include interrupts, drivers, multiple priorities, timers, I/O, and longer workloads. Second, the two implementations use different languages, compilers, runtime architectures, and synchronization primitives, so the experiment cannot isolate language cost from runtime cost. Third, the FreeRTOS firmware includes a periodic kernel tick, while the Embassy workload does not use an equivalent periodic scheduler tick. This difference can affect jitter. Fourth, only one board, clock configuration, and run per implementation are analyzed here. Repeating the experiment across multiple power cycles and boards would provide stronger evidence.

Memory footprint should also be interpreted carefully. The measured FreeRTOS Release image after integrating the benchmark used 9,084 bytes of text, 16 bytes of initialized data, and 33,464 bytes of BSS, but much of the BSS is the intentionally allocated 1,000-sample result buffer and benchmark stacks. A fair runtime-memory comparison would require building stripped production-like variants with the measurement buffers removed. Therefore, this paper does not claim a general memory-footprint advantage for either runtime.

VI. CONCLUSION

This paper presented a controlled benchmark of Rust/Embassy and C/FreeRTOS on the same STM32F411CEU6 microcontroller at 16 MHz. Both implementations executed a 256-byte CRC workload and a two-context request/reply test for 1,000 measured iterations after warm-up. Rust/Embassy achieved a median round-trip latency of 760 cycles (47.50 us), compared with 1,077 cycles (67.31 us) for FreeRTOS. FreeRTOS also showed a wider distribution, reaching 1,825 cycles at the 95th percentile.

The results demonstrate that an async Rust design can provide competitive and, in this test, lower synchronization latency than a conventional RTOS task-notification design. However, the result belongs to these specific implementations and settings; it is not evidence that Rust is universally faster than C. Future work should repeat the test across several runs and boards, compare

tickless FreeRTOS configurations, test multiple optimization levels and compiler versions, add realistic I/O workloads, and evaluate memory and energy use in addition to latency.

Reproducibility materials, including source code, raw benchmark data, analysis scripts, and figures, are available at:

<https://github.com/Nikoloz-22/embassy-vs-freertos-benchmark>

VII. REFERENCES

- [1] S. Klabnik, C. Nichols, and C. Krycho, The Rust Programming Language, “Understanding Ownership,” Rust Project, 2026. [Online]. Available: <https://doc.rust-lang.org/book/ch04-00-understanding-ownership.html>
- [2] Rust Embedded Working Group, The Embedded Rust Book, “A no_std Rust Environment,” 2026. [Online]. Available: <https://docs.rust-embedded.org/book/intro/no-std.html>
- [3] Embassy Project, “embassy-executor: Async/await executor designed for embedded usage,” 2026. [Online]. Available: <https://docs.embassy.dev/embassy-executor/>
- [4] FreeRTOS, “Direct-to-Task Notifications: xTaskNotify and xTaskNotifyIndexed,” 2026. [Online]. Available: <https://www.freertos.org/Documentation/02-Kernel/04-API-references/05-Direct-to-task-notifications/04-xTaskNotify>
- [5] STMicroelectronics, “STM32F411xC/xE Datasheet: Arm Cortex-M4 32-bit MCU + FPU,” 2024. [Online]. Available: <https://www.st.com/resource/en/datasheet/stm32f411ce.pdf>
- [6] R. Barry, Mastering the FreeRTOS Real Time Kernel: A Hands-On Tutorial Guide, FreeRTOS, 2018, sec. 9.2, “Task Notifications; Benefits and Limitations.”
- [7] N. Akhalaia, “Embassy vs. FreeRTOS Benchmark,” GitHub repository, 2026. [Online]. Available: <https://github.com/Nikoloz-22/embassy-vs-freertos-benchmark>

From Engineering Precision to Full Immersion: Integrating Complex CAD Models into Virtual Reality

Aleksandr Alikhanov

Georgian Technical University, Tbilisi, Georgia

Abstract — Modern virtual reality (VR) technologies and interactive 3D visualization systems provide a high degree of realism and user immersion. However, the integration of detailed engineering Computer-Aided Design (CAD) models into interactive real-time graphic environments is complicated by critical performance degradation caused by excessive polygon counts and topological complexity. This paper presents and scientifically substantiates a comprehensive three-stage methodology for the reduction of 3D geometric descriptions. The methodology includes the systematic removal of non-visible elements, algorithmic tessellation of parametric surfaces, and topological mesh optimization. Validation of the developed approach on the ATLAS detector dataset from CERN demonstrated a 94% reduction in the total number of objects and a 98.8% decrease in polygon density (from 485.5 million to 5.4 million polygons). This achieved a stable rendering frame rate of 60–90 frames per second in VR environments while preserving the visual fidelity and functional semantics of the model.

Index terms: *CAD model conversion, geometric reduction, polygon mesh optimization, normal alignment, B-Rep tessellation, VR visualization, CERN ATLAS.*

1. Introduction

Virtual reality (VR) technologies and interactive 3D visualization platforms are increasingly utilized in engineering design, fundamental scientific research, and educational applications. In large-scale high-energy physics experiments, such as the ATLAS experiment at the European Organization for Nuclear Research (CERN), 3D spatial geometric models serve as vital tools for engineering design, detector assembly, and science communication [1].

However, high-precision CAD geometry stored in specialized engineering databases (such as CATIA V5) is created specifically for manufacturing purposes rather than real-time graphical rendering. Such models are characterized by millions of hidden internal fasteners, complex structural primitives, and ultra-high-density analytical curves, which impose an unmanageable computational load on Central Processing Units (CPUs) and Graphics Processing Units (GPUs).

To achieve seamless interactive performance in VR environments, maintaining a target frame rate of 60–90 frames per second (FPS) is imperative. Exceeding hardware polygon

budgets—ranging from 1–5 million polygons for standalone mobile headsets to 10–50 million for high-performance workstations—leads to severe latency and graphics pipeline failures. Building upon the established foundations of geometric processing [2], this paper proposes a structured three-stage CAD simplification algorithm designed to bridge the gap between heavy engineering CAD specifications and real-time interactive rendering requirements.

2. Related Work

The transformation of CAD geometry into real-time polygonal meshes is a well-studied field in computer graphics literature [3], [4]. Standard automated tessellation algorithms [5] convert parametric surfaces into triangular meshes; however, they inherently generate excessive geometric density across flat planar faces as well as highly curved cylindrical bends.

Surface simplification techniques based on Quadratic Error Metrics (QEM) [6] effectively reduce polygon counts but, in the absence of strict topological constraints, frequently distort normal vectors or destroy critical structural boundaries. Although commercial converters offer varying degrees of mesh reduction, addressing the specialized requirements of large-scale scientific facilities [1], [7], [2] necessitates dedicated multi-tiered workflows capable of preserving functional design semantics alongside rendering performance.

3. Geometric Simplification Methodology

The proposed methodology for simplifying and adapting engineering CAD models for VR visualization is structured into three consecutive stages:

Stage 1: Component Decomposition and Culling of Non-Informative Elements

1. Functional Grouping: Spatial geometry is partitioned according to target functional subsystems (sensor modules, mechanical support structures, cooling conduits, electrical cabling) and organized into isolated assembly nodes.

2. Elimination of Non-Visible Geometry: Internal components obscured during external visual inspection (internal springs, bolts, screws, wiring connectors, micro-rivets) are systematically removed.

3. Consolidation of Fragmented Objects: Fragmented CAD sub-components representing a single logical entity (e.g., multi-segment cooling pipes or mounting flange assemblies) are merged into unified solid primitives, significantly streamlining the assembly tree.

Stage 2: Controlled Tessellation of B-Rep Descriptions

Proprietary CAD formats (e.g., .CATPart) store Boundary Representations (B-Rep) devoid of polygonal meshes. Direct export to interchange formats (such as WRL) often decomposes single solid CAD bodies into hundreds of disconnected faces and boundary lines. To prevent topological distortions, an intermediate conversion platform (KeyShot) is employed [7] (2). The tessellation tolerance parameter is precisely calibrated in the range of 0.01–0.1 to achieve an optimal balance between initial surface fidelity and polygon redundancy prior to mesh optimization.

Stage 3: Topological Mesh Optimization and Normal Vector Realignment

Within a 3D polygonal modeling environment (Blender), the tessellated mesh undergoes three distinct optimization steps:

1. **Coplanar Reduction:** Redundant vertices and coplanar triangles across flat planar surfaces are collapsed. Replacing dense planar triangulations with minimal polygon representations yields up to a 95% reduction in planar polygon density.

2. **Curved Geometry Optimization:** Radial vertex resolution for cylindrical structures (pipes, barrels) is optimized. For instance, reducing the base vertex count from 26 to 13 cuts polygon counts by half without discernible visual degradation. Redundant radial loops along pipe bends are systematically removed.

3. **Surface Normal Vector Adjustment:** By adjusting the split normal angles between adjacent polygons, low-poly faceted surfaces accurately diffuse light, visually simulating smooth curvature under dynamic lighting conditions.

4. Software-Algorithmic System Architecture

The end-to-end pipeline for scene preparation comprises a strictly deterministic sequence of processing steps executed across specialized software platforms:

- **Step 1. CAD Geometry Analysis and Initial Cleaning (CATIA V5):** The full structural hierarchy of the engineering assembly is inspected. Utilizing three-stage filtering, hidden fasteners are culled, functional subsystems are reorganized, and fragmented primitives are merged, significantly reducing the node count of the scene graph.

- **Step 2. Parametric Tessellation (KeyShot):** Grouped B-Rep assemblies are imported into KeyShot. To prevent GPU memory overflow during heavy dataset processing, Performance Mode is enabled with a controlled tessellation tolerance of 0.01. This converts analytical B-Rep curves into continuous triangular meshes without breaking unified solids into disconnected faces.

- **Step 3. Topological Mesh Reduction and Normal Alignment (Blender):** Tessellated mesh files are imported into Blender for coplanar reduction, cylindrical optimization, and normal vector smoothing following the algorithms in [2]. Normal smoothing allows low-polygon geometry to visually simulate high-resolution detail.

- **Step 4. Spatial Hierarchization and Asset Export (VR-Ready FBX):** In the final step [2], optimized mesh components are structured into a standardized three-level logical scene

hierarchy ('Detectors', 'Mechanical Support', and 'Services'), matching CERN's official data architecture. Color-coded materials are assigned to functional subsystems, and the assembly is exported into VR-ready FBX assets.

5. Experimental Evaluation and Results Discussion

The proposed methodology was experimentally validated through the development of an interactive virtual tour of the ATLAS detector complex (CERN) on the VRChat platform. The test dataset encompassed five major structural subsystems: Cavern, Service Platforms, Cooling System, Muon Detector, and Supports.

1. **Structural Object Reduction:** The total count of CAD components was reduced from 500,000 to 72 consolidated 3D objects, representing a 99.99% reduction in individual scene graph nodes.

2. **Polygonal Density Optimization:** The overall number of triangulated polygons was reduced from 485,553,507 to 5,449,365, achieving a 98.8% reduction while fully retaining key visual features and recognition metrics.

The resulting optimized 3D models were successfully deployed in an interactive virtual environment on VRChat. The application maintained a stable rendering frame rate of 60–90 FPS on standard desktop hardware as well as standalone Meta Quest 3 VR headsets.

6. Conclusion

This paper presented and experimentally validated an effective methodology for transforming complex engineering CAD models into optimized 3D assets suitable for real-time virtual reality applications. The synergistic combination of early non-visible element culling, controlled B-Rep tessellation, coplanar reduction, and normal vector smoothing yielded a 98.8% reduction in polygon density. The results confirm the suitability of this approach for high-performance immersive rendering across diverse hardware platforms. Future work will focus on automating normal map generation for baking micro-surface details.

REFERENCES

- [1] Sharmazanashvili A. Geometric Modeling in High Energy Physics // CERN Report. — 2022.
- [2] Alikhanov A. Methods for Processing 3D Geometric Descriptions for Visualization Applications: Master's thesis. — Tbilisi: Georgian Technical University, February 2025.
- [3] Foley J. D. et al. Computer Graphics: Principles and Practice. — 1st ed. — Addison-Wesley, 1982.
- [4] Bochkov A. Fundamentals of 3D Modeling: AutoCAD, KOMPAS-3D, SolidWorks, Inventor. — Moscow: EKMO, 2013.
- [5] Ivanov V. Three-Dimensional Modeling of Communication System Elements. — Moscow: Goryachaya Liniya - Telekom, 2018.

- [6] Hughes J. F. et al. Computer Graphics: Principles and Practice. — 3rd ed. — Addison-Wesley, 1982.
- [7] Surmava A. Development of Methods for Analysis and Synthesis of the Information Model of the ATLAS Detector: Ph.D. dissertation. — Tbilisi: Georgian Technical University, 2017.

Germans on the Move: An Interactive Scrollytelling Visualization of Travel Demand

Sandra Beckett

Abstract—Travel behavior provides valuable insights into social, economic and mobility patterns. This paper presents *Germans on the Move*, an interactive web-based information visualization that examines travel and tourism demand among the German population from 2012 to 2024. The project combines several datasets covering travel participation, age groups, destination countries, trip duration, and travel expenditures. Instead of presenting the data in a conventional dashboard, the system uses a scrollytelling approach to guide users through a sequence of information. Interactive visualizations, personalized inputs and calculators allow users to relate the statistical data to their own travel behavior. The application was implemented using HTML, CSS, JavaScript and D3.js and deployed as a web application. The resulting prototype demonstrates how interactive visualization and personalization can be combined to communicate complex tourism data to a broad, non-expert audience.

I. INTRODUCTION

Travel behavior is an important indicator of social and economic development. Changes in the number of trips, destination preferences, travel duration and expenditures can reveal broader changes in mobility and consumer behavior. For this reason, the project *Germans on the Move* focuses on the development of an interactive visualization of travel and tourism demand among Germans.

The primary objective is to investigate how travel behavior has changed over time and to make these developments understandable to a broad audience. The analysis covers the period from 2012 to 2024 and considers several dimensions of travel behavior, including overall travel participation, demographic differences, destination countries, trip duration, and expenditures. A particular focus is placed on long-term developments and disruptive events such as the COVID-19 pandemic.

Data for the project is based on the official German statistical survey on tourism demand. The survey covers the population aged 15 years and older and is based on approximately 10,000 respondents per year, providing a broad basis for analyzing travel behavior [1]. The project uses five related datasets that share the temporal dimension of the year as shown in Figure 1.

Instead of using a conventional dashboard, the project uses a scrollytelling approach. This allows users to follow a structured sequence while still interacting with the individual

visualizations. The main contribution of the project is therefore not a new analytical algorithm but an interactive visualization system that combines data-driven storytelling, multiple visualization techniques and personalized user interactions. The developed visualization is available online at “germans-on-the-move.vercel.app”.

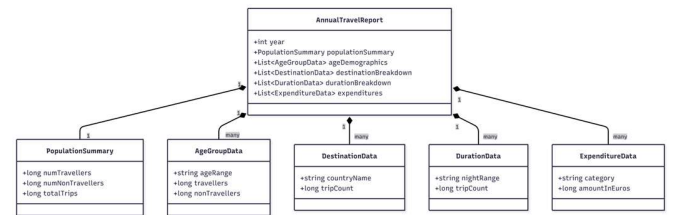


Fig. 1: Class diagram illustrating the structural organization of data elements and their relationships within the travel behavior analysis.

II. RELATED WORK

Interactive storytelling has become an established approach for communicating complex datasets to general audiences. In contrast to dashboards, which often emphasize simultaneous exploration of multiple views, scrollytelling organizes information sequentially. Visualizations and explanatory text are presented as part of a predefined narrative, allowing the designer to guide users through a set of related questions and observations.

Development of *Germans on the Move* was influenced by existing examples of data-driven storytelling. In particular the information visualization project *Why Do Women Outlive Men?* by Christina Epple served as an inspiration for the scrollytelling approach [2]. This example demonstrates how a sequence of visualizations and explanatory elements can be combined into a coherent data story.

Several established visualization techniques are also used, including as line charts, bar charts, slope charts and stacked area charts. These techniques were selected according to the analytical task: line charts are used to communicate temporal development, multi-line charts to compare demographic groups, slope charts to show changes in destination rankings and area and bar charts to represent expenditure development.

Unlike a purely exploratory dashboard, the presented system deliberately follows an explanatory approach. The goal is to communicate selected patterns clearly while still providing interactive controls that allow users to inspect the underlying data and personalize parts of the experience.

III. PROPOSED METHOD

The proposed system is structured around six analytical chapters:

1. **Rise and Fall:** development of overall travel demand
2. **Who Travels:** differences between age groups
3. **Where To:** destination preferences
4. **How Long:** distribution of trip durations
5. **The Money:** development of travel expenditures
6. **What Remains:** summary of the main insights

Users first encounter a personalized entry point called “Your Journey”, as illustrated in Figure 2. They can enter their age group, number of trips per year and favorite destination country. These inputs are stored and reused later in the visualization. This design establishes a connection between the statistical data and the individual user's travel behavior.

An editorial travel-diary aesthetic defines the visual design. The interface uses a warm visual atmosphere inspired by old travel journals, with blue, green, orange, beige and gold tones. The concept of “data as landscape” is incorporated into the visual language, particularly through flowing layouts and slope charts.

As shown in Figure 2, a sticky navigation bar allows users to move between chapters, while a scroll-based progress indicator communicates their current position within the story. Each chapter contains a title, subtitle, introductory text and one or more interactive visualizations. This consistent structure supports readability and creates a continuous flow.

The sequence of information helps users understand the different aspects of travel behavior step by step. At the same time, interactive controls allow them to filter and inspect the data. Personalization connects the statistical information to the users' own travel behavior.

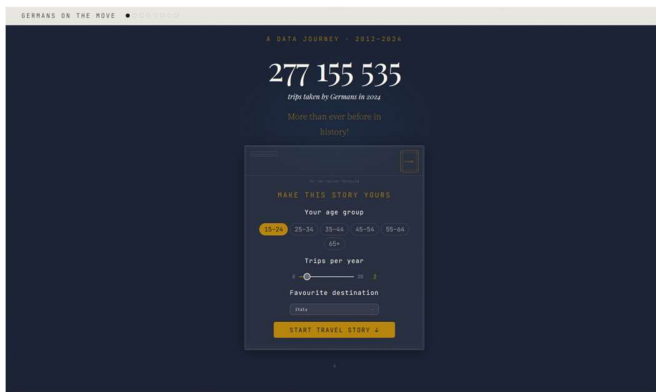


Fig. 2: Sticky navigation bar with chapter navigation, scroll progress indicator, introductory hook and personalized input section designed as postcards.

IV. SYSTEM ARCHITECTURE

A visual moodboard and storyboard were created to define the visual style and structure of the application. For prototyping of the design was used Figma. During this stage, the complete visual appearance of the application was developed, including the layout, typography, color scheme, and interaction flow. A clickable prototype was also created to demonstrate how users

would navigate through the different sections and interact with the visualizations.

Implementation was carried out as a desktop-oriented web application using HTML, CSS, JavaScript and D3.js. Development was performed using Visual Studio Code and Live Server, while the finished application was deployed through Vercel [3]. The visualization was tested on Full HD and WQHD monitors and is currently not optimized for mobile or tablet devices.

The source datasets were originally provided in CSV format. During preprocessing, irrelevant textual information and footnotes were removed, column names were standardized, empty rows and columns were eliminated and multi-row headers were simplified. Missing values were marked with “x” and incomplete entries were removed where necessary.

After data cleaning, the CSV datasets were converted into JSON. Instead of merging the datasets into one large table, they remain as separate but related data sources. They are conceptually connected through their shared temporal dimension, allowing the individual visualization components to access the relevant dataset independently.

A chapter-based structure organizes the application. As the user scrolls through the page, individual sections dynamically present their corresponding text and visualizations. The JSON files are integrated through JavaScript which also controls interaction and visualization logic.

The first analytical visualization is a line chart showing the total number of trips, travelers, and non-travelers between 2012 and 2024. Users can switch between absolute values and percentage change. A dashed vertical line highlights 2020 to provide contextual information about the COVID-19 period. Individual data series can also be enabled or disabled.

For the demographic analysis, a multi-line chart compares travelers and non-travelers across age groups, as shown in Figure 3. Users can select individual age groups and toggle between the two categories. Hover interactions provide exact values or percentages and display a corresponding guide line.

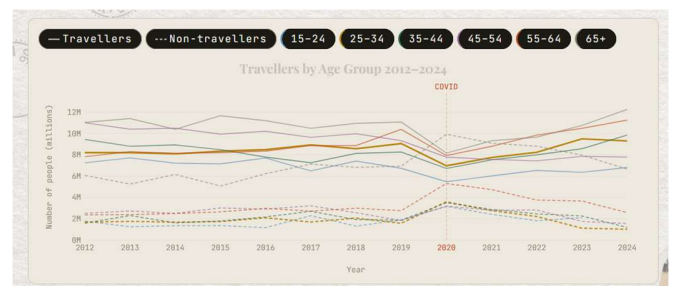


Fig. 3: Multi-line chart of travel behavior by age group.

Destination development is represented using a slope chart. Users select two years with sliders after which the visualization compares destination rankings between these years. The system additionally highlights the user's selected favorite destination and dynamically generates insight panels, including the biggest climber and the new number-one destination. A bar-chart view is also available for examining a single year.

Trip duration is visualized using absolute and normalized views for three categories: 1–3 nights, 4–9 nights, and 10+

nights. This section also contains a Trip Calculator which combines the user's number of annual trips with an adjustable average number of nights. The resulting estimate of total nights is used to create a travel profile and compare the user's behavior with the average German traveler.

In Figure 4 travel expenditures are presented using a stacked area chart and a bar chart. The expenditure categories are accommodation, transport, food and drink and other expenses. Users can switch between absolute values and percentage shares and selectively display categories. The Travel Budget Calculator estimates annual spending based on the user's number of trips and average nights and compares the result with the dataset-derived average.



Fig. 4: Stacked area and bar chart of travel expenditures with category toggles and integrated budget calculator for personalized cost estimation.

V. EVALUATION

The evaluation of the project focuses primarily on the functionality and visualization concept rather than on a formal controlled user study. The resulting prototype combines five related tourism datasets into an interactive experience and provides multiple forms of user interaction.

Temporal analysis is supported by the overall travel-demand line chart and expenditure visualizations. Demographic differences can be explored through selective age-group filtering while destination changes can be examined between arbitrary pairs of years. Both the duration and expenditure sections also provide personalized calculations based on user input.

One important design decision was the use of scrollytelling instead of a dashboard. This approach structures the information into six consecutive analytical questions and provides contextual explanations alongside the visualizations. At the same time reduced exploratory flexibility is a limitation of the approach. Since the information is presented in a predefined order, users have less freedom to explore the complete dataset independently.

The implementation also has technical limitations. The temporal range from 2012 to 2024 is partially hardcoded, meaning that extending the visualization with additional years requires modifications to the visualization logic. Some explanatory texts are also hardcoded and therefore require manual updates. Although the JSON-based data structure allows relatively easy data modifications, further abstraction would improve the configurability of the system.

Performance is another area for improvement. The current implementation relies on multiple JavaScript functions, which can result in noticeable loading times. In addition, the current version is optimized for desktop displays rather than mobile or tablet devices.

Despite these limitations, the prototype demonstrates that combining narrative structure, interactive charts, and personalized components can create an engaging way of communicating tourism statistics to a non-expert audience.

VI. CONCLUSION

This paper presented *Germans on the Move*, an interactive scrollytelling visualization of travel and tourism demand among Germans between 2012 and 2024. Several statistical datasets are combined into a structured data story covering travel participation, demographics, destinations, trip duration, and expenditures.

The central design principle is the combination of structured presentation, interactive visualizations and personalized elements. Users are guided through six chapters while being able to filter data, inspect individual values, compare years and relate statistical information to their own travel behavior. Using HTML, CSS, JavaScript, and D3.js, the project demonstrates how web technologies can be used to create an interactive and personalized visualization.

Several extensions are planned for future development. Additional animations could improve transitions between visual states while performance optimization could reduce loading times. More extensive personalization could provide additional user-specific insights. Accessibility could be improved through enhanced color choices and contrast and responsive design would extend the system to mobile and tablet devices.

REFERENCES

- [1] German Federal Statistical Office (Destatis), "Statistics on tourist demand," Genesis-Online, statistic 45413. [Online]. Available: <https://genesis.destatis.de/datenbank/online/statistic/45413/details> [Accessed: Sep. 27, 2026]
- [2] C. Epplé, "Why Do Women Outlive Men?" Information Visualization project. [Online]. Available: <https://534years.vercel.app/> [Accessed: Sep. 27, 2026]
- [3] Vercel, "Vercel." [Online]. Available: <https://vercel.com/> [Accessed: Sep. 27, 2026]

Designing Age-Inclusive Virtual Reality: Barriers and Guidelines for Older Adults in Heritage Applications

Lea Vianne Eichler

Abstract—Although virtual reality is increasingly being used to make cultural heritage accessible to the public, older adults and those without prior VR experience often face significant barriers when interacting with such applications. In this paper, we present the design and implementation of an age-inclusive VR application developed for the Saxony State Office for the Preservation of Historical Monuments. This application allows visitors to explore archived buildings modelled in Blender and deployed on the Meta Quest 3 via Unity. Based on well-established design principles from recent VR research involving older adults, we implemented a simplified, single-button control scheme; seated interaction; guided verbal instructions; and strategies to mitigate motion sickness. The application was evaluated through observation during a public exhibition event, revealing high user engagement as well as specific usability challenges, such as difficulty discovering the 360° view and initial confusion regarding the use of the controllers. Our findings offer practical guidance for designing accessible VR experiences for older and inexperienced users in cultural heritage contexts.

Index Terms—accessibility, cultural heritage, older adults, virtual reality

I. INTRODUCTION

VR is already being used in a wide range of applications, including among older adults, for example, for the early detection of Alzheimer's disease or for knowledge transfer. Developing intuitive and functional VR applications poses a significantly greater challenge for groups of people without prior digital experience than for those who have grown up with this technology.

As part of the elective module “Modeling for the Saxony State Office for the Preservation of Historical Monuments,” a VR application was developed for the 3D exploration of buildings and objects (see Fig. 1). The underlying plans and drawings were sourced from the archives of the Saxony State Office for the Preservation of Historical Monuments. In the first step, students recreated these models in Blender and then embedded them in a 3D environment using Unity for the Meta Quest 3. The goal is to make the work of the State Office and its archives accessible to a broader public and to promote the revitalization of historic buildings. This application was presented as part of Open Monument Day. The majority of visitors were older adults or people with no prior experience using VR headsets.



Fig. 1: Guided walkthrough around the modeled building.

Various problem areas can be identified as a result. One problem with control arises from the unfamiliarity of operating the controllers. In addition, older adults often experience a lack of strength and limitations in fine motor skills [1]. As a result, their ability to perform complex and time-consuming control tasks is severely limited. A lack of knowledge of the English language and technical terminology can lead to language barriers and difficulties in explaining the functions.

VR applications can cause increased nausea and stress in older adults. It is of essential importance that the application does not lead to a state of disorientation, anxiety, difficulty concentrating, or rapid eye and neck fatigue in the participants. It is noteworthy that the prevalence of cybersickness among the older population is at less than 10% [1]. The age-related decline in perception, cognition, and sensorimotor skills makes the basic use of digital services more difficult. This can lead to a lack of familiarity and confusion in 3D environments [2].

II. RELATED WORK

Two recent studies provide the theoretical foundation for the design decisions in this work. Schaumburg et al. [1] examined age-related sensorimotor and cognitive decline as the main barriers to VR use among older adults. They identified disorientation, anxiety, reduced task motivation and physical fatigue as common challenges. Notably, their findings suggest that cybersickness is comparatively rare among older users, indicating that motion-related discomfort is a secondary concern for this group. Based on these observations, [1] recommends adapting movement speed to the user's natural walking pace, gradually increasing task complexity and

ensuring frequent breaks to counteract physical exhaustion.

Lu et al. [2], in contrast, approach the problem from the perspectives of familiarity and trust. Their study of VR travel applications for older adults introduces the concept of 'familiarity design', arguing that older users engage more confidently with interfaces resembling known real-world patterns and interactions. Central to their findings are the effectiveness of anthropomorphic virtual agents as guides, and a dual-layer, three-dimensional interface that visually separates background content from interactive elements. While [1] primarily focuses on physiological and cognitive limitations, [2] emphasises psychological factors such as trust, orientation, and perceived control.

Both studies converge on several practical recommendations, including simplified gesture-based or single-button controller interaction, self-paced rather than automated movement and interactive elements positioned within a comfortable visual range. However, neither study addresses VR applications in the specific context of cultural heritage presentation nor evaluates its recommendations in an uncontrolled real-world exhibition setting involving a broad, heterogeneous audience. Our work builds on the combined recommendations of [1] and [2] by translating them into a concrete implementation and testing them under realistic public exhibition conditions — a context not covered by either prior study.

III. PROPOSED METHOD

The core idea of our approach is to minimize interaction complexity while maximizing user confidence and orientation, addressing the physiological limitations identified in [1] and the trust-and-familiarity factors emphasized in [2] simultaneously. Rather than treating controller simplification, seated interaction, and guided narration as isolated measures, we combine them into a single coherent interaction concept: users remain seated throughout the experience, physical movement is entirely replaced by controller-based selection, and all transitions between application states are accompanied by verbal explanation from a human supervisor. This reduces the cognitive and motor demands placed on the user at any given moment, allowing attention to remain on the content (the modeled heritage buildings) rather than on operating the device.

IV. SYSTEM ARCHITECTURE

The system comprises a content pipeline and a runtime application. In the content pipeline, source buildings and objects are reconstructed as 3D models in Blender based on archival plans and drawings supplied by the Saxony State Office for the Preservation of Historical Monuments. These models are then imported into Unity, where they are integrated into an interactive scene and prepared for deployment as a standalone application on the Meta Quest 3 (see Fig. 2).

During runtime, the application takes the user through a set

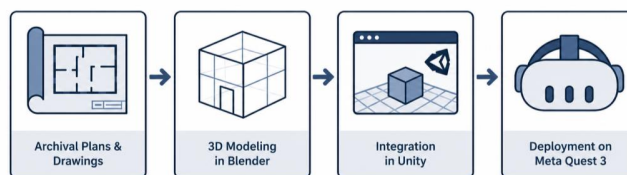


Fig. 2: Content pipeline from archival plans to VR deployment (Image created with ChatGPT)

sequence of stages: an introduction, selecting an object, a guided tour of the modelling process and exploring the chosen object. Progression between these stages is controlled entirely by one button on a single physical controller, eliminating the need for pointing- or gesture-based selection. An optional second controller allows users to browse supplementary information presented as a virtual flyer (see Fig. 3). Each stage transition is accompanied by synchronised audio narration, providing a simultaneous visual and auditory explanation in line with the dual-coding recommendation of [1]. A static visual reference point (a virtual railing) remains present around the user throughout any camera movement to support spatial orientation.



Fig. 3: Supplementary information displayed as a virtual flyer, accessed via the second controller.

V. EVALUATION

The application was informally evaluated through direct observation on Open Monument Day (see Fig. 4), when visitors of various ages and with different levels of VR experience used it without a controlled testing protocol.



Fig. 4: A visitor using the VR application in a seated position during Open Monument Day.

At first, most users found it difficult to figure out how to use the controller. Although the mentioned function had not been implemented, many participants attempted to point at objects using the controller. Despite the precautions taken, some users experienced VR motion sickness. Overall, the older participants showed a significantly high level of interest and marked enthusiasm for VR applications and the modeled buildings. It should be noted that many older users were unaware of the possibility of a 360° view in the virtual world. Generally, users directed their gaze in only one direction, which limited both their control and their ability to orient themselves within the building. The supervisory staff was able to resolve most difficulties on the spot. Instructions on how to hold the controller also had to be given before users put on the headset, since they could not see the physical controller once wearing it.

VI. CONCLUSION

The aim of this study was to develop a basic VR concept for older adults that is accessible. To this end, a combination of literature-based design principles [1] [2] was applied alongside practical testing in a real exhibition setting. Simplified controls, a seated position, and clear and repeated explanations were identified as key success factors.

Building on the experience gained from this project, additional applications of a similar nature are already planned, including one for the State Parliament of the Free State of Saxony. A key aspect of further development is the provision of content as three-dimensional PDF files. According to feedback from the State Office, these are more user-friendly than the VR application and do not require complex and costly equipment.

Furthermore, the application's accessibility is to be further expanded in the future. Adding support for additional languages is a fundamental goal of the project. The implementation of this measure aims to provide visitors who do not speak German with adequate access to the content. In addition, plans are in place to expand the application to include sign language as well as a

version in plain language, in order to increase accessibility for deaf people and individuals with limited German language skills or cognitive impairments.

The solutions developed as part of this project form a solid foundation for the further development of accessible VR and 3D applications in the field of historic preservation. For future projects, however, it is recommended that these approaches be supplemented by more systematic user testing under more controlled conditions in order to identify potential for improvement in an even more targeted manner.

REFERENCES

- [1] Schaumburg, M., Imtiaz, A., Zhou, R., Bernard, M., Wolbers, T. & Segen, V. (2025). Immersive virtual reality for older adults: Challenges and solutions in basic research and clinical applications. *Ageing Research Reviews*, 109, 102771. <https://doi.org/10.1016/j.arr.2025.102771>
- [2] Lu, X., Tong, B., Xiao, J., Gong, D. & Zhang, X. (2025). Immersive virtual reality travel applications for older adults: familiarity design focusing on representation, manipulation, and guidance. *Humanities and Social Sciences Communications*, 12(1), 1–14. <https://doi.org/10.1057/s41599-025-05816-6>

Making UDP Keep Its Promises: Go-Back-N ARQ over IPv6

Juan Ramiro Groh Fili and Richard Ullmann

juanramiro.grohfil@stud.htw-dresden.de, richard.ullmann@stud.htw-dresden.de

Faculty of Computer Science / Mathematics, HTW Dresden, Germany

In this paper, we present a connection-oriented file transfer protocol that adds reliability on top of UDP/IPv6 using Go-Back-N ARQ with positive cumulative acknowledgements. The sender is built as a single-threaded, time-slotted loop over one socket: at most one datagram is transmitted per fixed interval, and retransmission timers are counted in intervals rather than measured in wall-clock time. The send window is realised as a ring buffer, while the receiver holds no buffer and discards every out-of-order packet, repeating the cumulative acknowledgement of the last in-order packet instead. Packet and acknowledgement loss are injected deterministically at the receiver to make the protocol behaviour reproducible. The protocol transfers files correctly for all window sizes from 1 to 10 and recovers from both data and acknowledgement loss.

Index Terms—ARQ, Go-Back-N, IPv6, UDP.

I. INTRODUCTION

Reliable data transfer is required in numerous domains including file transfer, messaging, and remote configuration of embedded devices. While TCP provides this service, it hides its mechanisms behind a stream abstraction and adds connection and congestion behaviour that is not always wanted. UDP, in contrast, delivers single datagrams without ordering, duplicate detection, or delivery guarantees [1], [2], so any reliability must be built on top of it. Among the classic ARQ schemes, stop-and-wait is simple but idles for one round-trip time per packet, and selective repeat recovers efficiently but requires the receiver to buffer and reorder. This paper proposes a Go-Back-N protocol over UDP/IPv6 whose sender is built as a strictly time-slotted loop, avoiding threads and additional sockets entirely. We demonstrate its behaviour under injected packet and acknowledgement loss and describe the resulting design trade-offs.

II. RELATED WORK

Numerous ARQ schemes have been studied over the decades. Stop-and-wait [1] remains popular in teaching due to its minimal state, but its throughput is bounded by the round-trip time. Go-Back-N [1], [2] permits a window of unacknowledged packets and uses cumulative acknowledgements, so the receiver needs only a single state variable and no buffer at all. Selective repeat [2] acknowledges packets individually and retransmits only what was actually lost, which makes it clearly superior on lossy links; the price is a receiver that must buffer out-of-order packets, reassemble them, and maintain per-packet acknowledgement state, and a sender that needs an independent timer per outstanding packet. Its practical relevance is visible in TCP, where selective

acknowledgements were added as an option [5] precisely because purely cumulative acknowledgements retransmit more than necessary. Go-Back-N accepts redundant retransmissions in exchange for this simplicity, and the redundancy grows with the window size and the loss rate. This trade-off is the central design decision of our work: we deliberately choose the scheme whose receiver stays stateless. TCP [4] combines cumulative acknowledgements with adaptive timers and congestion control; the layers used here are defined by RFC 768 [3] for UDP and RFC 2460 [6] for IPv6. Our work builds upon these foundations and introduces a thread-free, interval-driven sender architecture.

III. PROPOSED METHOD

Our Go-Back-N implementation introduces four core design decisions: (1) a time-slotted sender loop in which a single select() call per 100 ms interval serves as clock generator, receive path and timer tick, so that at most one datagram leaves per interval; (2) retransmission timers expressed as counted intervals rather than measured wall-clock time, with a timeout declared after three idle intervals; (3) a strictly stateless receiver that holds no buffer and answers every out-of-order packet by repeating the cumulative acknowledgement of the last in-order packet; and (4) a send window realised as a ring buffer that stores every transmitted but unacknowledged packet, so that a retransmission never requires re-reading the input file. These decisions collectively remove concurrency from the implementation: sending, receiving and timer management become successive steps of one deterministic loop, and the complete sender state reduces to three integers plus the ring buffer.

IV. ALGORITHM DESCRIPTION AND SYSTEM ARCHITECTURE

Client and server are each split into an application layer (client.c, server.c) handling file access and arguments, and a protocol layer (clientSy.c, serverSy.c) holding the ARQ logic and all socket calls. A session runs in three phases: a HELLO packet opens the connection, DATA packets carry one file line of at most 512 bytes each, and a CLOSE packet terminates it; all three are acknowledged. The send window is a ring buffer of twice the maximum window size, indexed by sequence number modulo the buffer size, so a retransmission never requires re-reading the file.

Figures 1 and 2 show a loss-free exchange and two loss cases. The receiver discards out-of-order DATA packets and repeats the cumulative ACK.

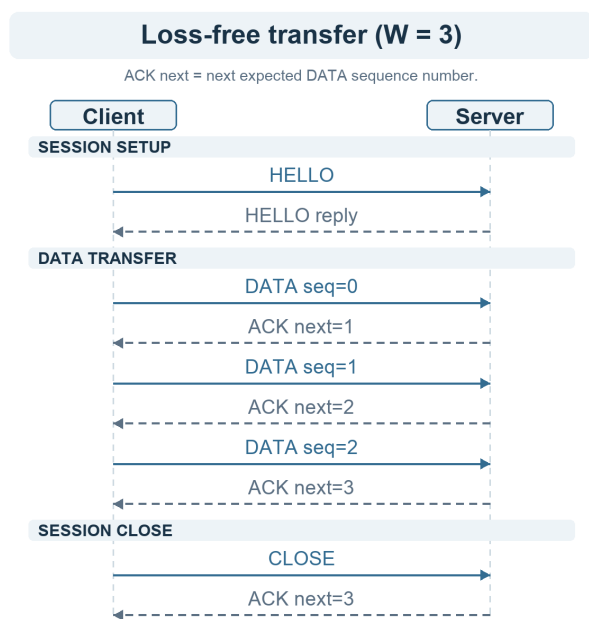


Fig. 1. Loss-free HELLO, DATA, and CLOSE exchange with a maximum send window of three packets.

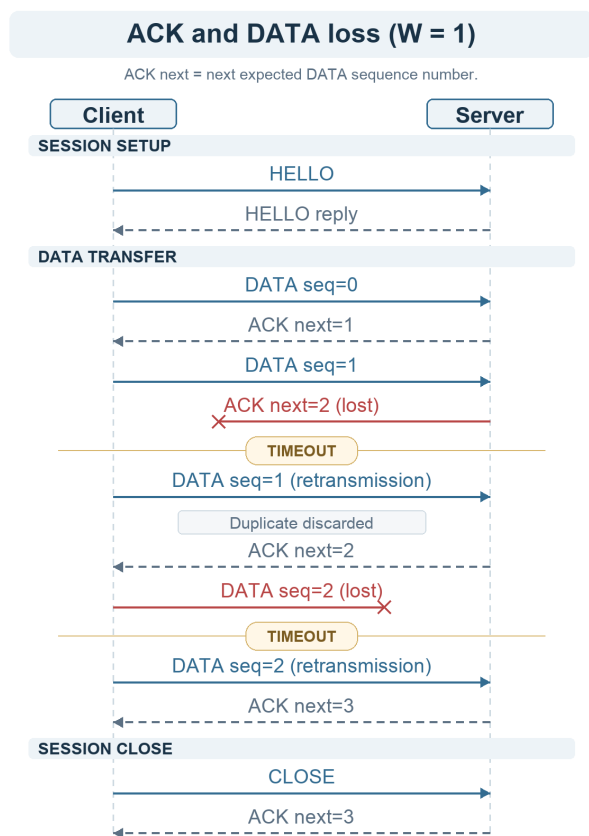


Fig. 2. ACK and DATA loss with a window size of one. A lost ACK produces a duplicate DATA packet; a lost DATA packet is retransmitted after timeout.

V. EVALUATION

We evaluated our implementation on Linux over the IPv6 loopback interface, transferring text files line by line with a payload of at most 512 bytes per packet. Because loopback transfers are effectively lossless, loss is injected at the server through two independent probabilities, one for incoming requests and one for outgoing acknowledgements, driven by a fixed random seed so that every run with identical parameters reproduces the identical loss pattern. Tests were run for different window sizes and for both loss types, individually and combined. In every case the transfer completed and the received file matched the original; only the number of transmitted datagrams increased. The console trace shows the characteristic Go-Back-N behaviour: after a loss the receiver repeats the same cumulative acknowledgement while the sender fills its window, until the timeout fires and the window is replayed from the gap onwards. A lost acknowledgement is indistinguishable from a lost data packet at the sender, but produces a duplicate at the receiver, which the sequence check rejects correctly. The main limitation is throughput, which is bounded by construction at one packet per 100 ms interval.

VI. CONCLUSION

We presented a connection-oriented file transfer protocol over UDP/IPv6 that provides reliable delivery through Go-Back-N ARQ with cumulative acknowledgements. By treating time as a sequence of discrete intervals, we achieve window management, retransmission timing and acknowledgement processing without threads and with one socket per instance. The implementation proves robust under injected loss and keeps every protocol decision traceable in its output. Future work includes an adaptive timeout derived from measured round-trip times and receiver-side buffering to obtain selective repeat.

REFERENCES

- [1] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 8th ed. Pearson, 2021, ISBN 978-1-292-40546-9.
- [2] A. S. Tanenbaum, D. J. Wetherall, and N. Feamster, *Computer Networks*, Global Edition, 6th ed. Pearson, 2021, ISBN 978-1-292-37401-7.
- [3] J. Postel, "User Datagram Protocol," RFC 768, 1980.
- [4] J. Postel, "Transmission Control Protocol," RFC 793, 1981.
- [5] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, 1996.
- [6] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, 1998 (obsoleted by RFC 8200, 2017).

Measurement of the Informational Parameters of an Oscillatory System Under High Electromagnetic Losses

Akaki Javakhishvili

Georgian Technical University, Tbilisi, Georgia

Abstract—Distributed-parameter oscillatory systems can serve as sensitive elements for indirect measurement of non-electrical quantities through their resonance characteristics. Conventional resonance-frequency measurement approaches may require narrow, predefined frequency ranges, repeated excitation, or manual processing of amplitude-frequency characteristics, which limits automation and operation with low-quality-factor systems.

This article, presents a measurement approach based on a direct digital synthesis (DDS) frequency source and an embedded controller. A VCO is replaced by a DDS synthesizer, enabling an automated 0–40 MHz sweep in the developed system. A three-stage search algorithm progressively narrows the frequency interval around the maximum detected response. The stored frequency-voltage data are also used to estimate the passband and quality factor. Numerical simulation was used to evaluate the method for Q values from 3 to 10. For the investigated nonlinearity conditions, the reported relative error of the proposed method did not exceed approximately 1%, compared with more than 1.5% for the classical approach at $Q \leq 5$.

Index Terms— Measurement automation, Oscillatory system, Quality factor, Resonant frequency.

I. INTRODUCTION

Resonance parameters of distributed-parameter oscillatory systems can be used as informational quantities for indirect measurement and control of physical objects. Such approach is relevant when the sensitive element is exposed to conditions in which conventional measurement hardware may lose accuracy or reliability. The research considered volumetric resonators and focused on determining resonance frequency, passband, and quality factor from the amplitude-frequency characteristic (AFC) [1], [2].

Four practical approaches to resonance-frequency determination are discussed in the source work: autogenerating, impact excitation, expandable, and follower methods. Their limitations include dependence on a narrow operating range, repeated excitation, discrete sampling of the AFC, long measurement time, and sensitivity to the quality factor of the oscillatory system [1], [3]. In addition, a classical implementation, investigated in the project, uses a VCO as the variable-frequency source and requires frequency-range parameters to be selected in advance [2].

The main contribution of this work is the transition from a manually adjusted or VCO-based measurement process to an embedded, digitally controlled measurement system. A DDS synthesizer provides the variable-frequency excitation, while a microcontroller controls the sweep, detects response maxima,

calculates the resonance frequency, and derives the passband and Q factor from the stored measurements.

II. RELATED WORK

Existing parameter-determination approaches include mathematical and simulation modeling, impedance-based measurements, electromechanical analogies, and direct experimental techniques [1], [5]. The classical n-th-order extrapolating-polynomial approach reviewed in the paper samples several amplitude levels of the AFC and uses the resulting frequency points to calculate an extrapolating function whose maximum represents the resonance frequency [2]. This approach can provide accurate results for the investigated conditions, but it requires multiple measurements and mathematical processing and remains dependent on the operating range of the VCO.

The source study emphasizes that low Q values make the resonance peak wider and increase the influence of AFC asymmetry and measurement nonidealities. The proposed method therefore uses a progressively narrowing frequency search rather than relying on a fixed set of manually selected operating points. This preserves the information contained in the measured AFC while shifting the numerical processing into the embedded system.

III. PROPOSED METHOD

The proposed system replaces the VCO with an AD9850 DDS synthesizer and uses an STM32G0B1-family microcontroller as the main control unit. In the investigated realization, the synthesizer covers 0–40 MHz, allowing the initial search interval to be selected without requiring the user to know the resonance frequency in advance. The controller measures the response amplitude, stores frequency-voltage pairs, identifies the maximum, and then repeats the search in progressively smaller intervals.

A practical implementation also requires the detection chain to remain faster than the excitation sweep. Initially, an AD823 device was tested, but its stated upper operating frequency of 16 MHz would limit the 40 MHz target range. It was consequently replaced with a TLV3501 comparator, whose specified speed is sufficient for the investigated frequency range. The final interface uses an ST7789-based TFT display for displaying numerical measurement results and, where desired, the measured frequency-amplitude curve.

IV. ALGORITHM DESCRIPTION AND SYSTEM ARCHITECTURE

The measurement is organized as a three-stage coarse-to-fine search. At each tested frequency, the peak response voltage is detected and stored together with the generated frequency. Stage 1 scans the full user-defined interval using a relatively large step and selects the frequency Y associated with the largest measured response. Stage 2 scans a smaller interval around $Y - \Delta f$ to $Y + \Delta f$. Stage 3 repeats the process over an even narrower interval with a smaller step, producing the final resonance estimate. The same principle can be extended to n stages when higher resolution is required, at the cost of longer measurement time.

Input: FA , FB , $\Delta 1$, $\Delta 2$, $\Delta 3$, number of steps

1. Sweep f from FA to FB with step $\Delta 1$ and measure $U(f)$.
2. Set $Y = \arg \max U(f)$.
3. Sweep around Y with step $\Delta 2$ and set $Y = \arg \max U(f)$.
4. Sweep a narrower interval around Y with step $\Delta 3$.
5. Set f_0 = frequency at the maximum measured response.
6. From stored samples, locate f_1 and f_2 using formula: $U = 0.707 \cdot U_{\max}$.
7. Compute $\Delta f = f_2 - f_1$ and $Q = f_0 / \Delta f$.

The corresponding frequency bounds may be written as $FA < H < FB$, with the initial sweep range $F = FB - FA$. The first-stage samples are generated as $F_i = FA + i\Delta 1$. Subsequent stages use the previously selected maximum as the center of progressively narrower intervals. After the final stage, the complete frequency-voltage history remains available for AFC reconstruction and for determining the half-power frequencies f_1 and f_2 . The quality factor is then calculated as $Q = f_0 / \Delta f$, where $\Delta f = f_2 - f_1$ at the $0.707 \cdot U_{\max}$ level [1], [3].

V. EVALUATION

The research used analytical-imitation modeling and numerical calculations implemented in Microsoft Excel. The simulated AFC was evaluated over 10–50 MHz for representative resonance frequencies and for Q values of 3, 5, 8, and 10. Nonidealities were represented through parameters describing linear-ramp nonlinearity, sample-and-hold error, additional nonlinearity, and detector error. The same conditions were applied to the classical and proposed methods so that the effect of the search principle could be compared.

Selected baseline simulation results are: $Q=3$, classical 4.541% and proposed 0.996%; $Q=5$, 1.546% and 0.354%; $Q=8$, 0.593% and 0.138%; $Q=10$, 0.378% and 0.088%. Thus, for the baseline condition, the proposed search reduces the reported error by 3.545, 1.192, 0.455, and 0.290 percentage points, respectively.

For the baseline condition, the classical method produced relative errors of 4.541%, 1.546%, 0.593%, and 0.378% for $Q=3$, 5, 8, and 10, respectively. Under the same baseline condition, the proposed method produced 0.996%, 0.354%, 0.138%, and 0.088%. The results showed that, for the broader set of tested nonlinearity combinations, the proposed method remained at or below approximately 0.99% in the investigated $3 < Q < 10$ region, while the classical approach exceeded 1.5% for $Q \leq 5$.

These results are simulation results rather than a complete metrological validation. The source work identifies the next experimental step as connecting the model to a real observation object and comparing measurements against an artificial oscillatory system with known parameters. Therefore, the present evaluation supports the proposed algorithm under the modeled conditions but does not by itself establish performance for arbitrary physical resonators.

VI. CONCLUSION

An embedded measurement approach for resonance-frequency determination in distributed-parameter oscillatory systems was derived from the master's research. The central change is the replacement of a VCO with a DDS synthesizer, combined with a microcontroller-controlled coarse-to-fine search and automatic response-peak detection. The resulting system can operate over the investigated 0–40 MHz range, store the measured frequency-voltage history, and calculate resonance frequency, passband, and Q factor without requiring external numerical processing.

For the simulation conditions reported in the project, the proposed method reduced the modeled resonance-frequency error relative to the classical approach across $Q=3$ –10. The architecture is also adaptable to a wider frequency range by changing the frequency synthesizer and adapting its control software, while retaining the same search principle. Future work identified by the project is experimental validation using a known artificial oscillatory system and comparison of measured parameters with their calculated values.

VII. REFERENCES

- [1] A. Frangishvili, Z. Azmaiparashvili, and N. Otkhazia, "Development, Research and Analysis of Methods and Means for Determining Informational Parameters of Oscillatory Systems," Technical University, Tbilisi, 2017.
- [2] Z. Azmaiparashvili, "Measurement of Resonance Frequency of Radio-Wave Sensors Under High Electromagnetic Losses," *Proceedings of GTU*, no. 1(369), 1990, pp. 67–72.
- [3] I. Epatashvili, "Expandable Method for Determining the Resonant Frequency of an Oscillatory System and Its Features," *Georgian Engineering News*, no. 4, 2017, pp. 12–14.
- [4] A. G. Soshinov, *Preobrazovatel' Nelinektricheskikh Velichin*, Volgograd: Politekhnik, 2002.
- [5] Z. Azmaiparashvili, N. Otkhazia, and I. Epatashvili, "Analytical-Imitation Model of a Device for Determining the Basic Informational Parameters of Oscillatory Systems," *Proceedings of the Archil Eliashvili Institute of Control Systems*, no. 21, 2017, pp. 71–76.
- [6] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, New York: McGraw-Hill Education, 2015.
- [7] A. Frangishvili and Z. Azmaiparashvili, "Method for Determining the Natural Frequency of an Oscillatory System," *Georgian Patent GE 1250 B*, 1997.
- [8] A. Frangishvili and Z. Azmaiparashvili, "Device for Automatic Determination of Oscillatory-System Parameters," *Georgian Patent GE 1621 B*, 1999.
- [9] Z. Azmaiparashvili and O. Tomaradze, *Sensors and Intelligent Measurement Means*, Tbilisi: Technical University, 2017.

Design and Implementation of a Low-Cost Arduino-Based Digital Spirometer

Tamar Jikia, Luka Maximilian Kuprashvili, Giorgi Peikrishvili
Georgian Technical University, Tbilisi, Georgia

Abstract-Conventional clinical spirometers are often too expensive for routine personal lung-function monitoring. This paper presents a low-cost digital spirometer prototype built around an Arduino Nano, an HX710B differential pressure sensor, and a 3D-printed breathing tube containing a 4 mm orifice disc. The device applies Bernoulli's principle to convert an exhalation-induced pressure drop into a flow signal, which a five-state C++ firmware processes into Forced Vital Capacity (FVC) and Peak Expiratory Flow (PEF). Measurements are automatically calibrated against ambient pressure and compared to standard prediction equations based on the user's sex, age, height, and smoking status, with results shown on an SSD1306 OLED display. The prototype successfully tracks airflow dynamics despite minor sensor noise, demonstrating that commodity microcontrollers, pressure sensors, and 3D printing can produce a functional, budget-friendly health-screening tool.

Index Terms- Arduino, biomedical engineering, pressure sensor, spirometer.

I. INTRODUCTION

Spirometry is the standard clinical method for assessing lung function, most commonly through the metrics Forced Vital Capacity (FVC) and Peak Expiratory Flow (PEF)[1]. Clinical-grade spirometers, however, typically cost several hundred to several thousand euros, which puts routine or preventive monitoring out of reach for individual users, small clinics, and educational settings. This creates a gap between the clinical importance of lung-function tracking and its practical accessibility.

This project addresses that gap by developing a self-contained, low-cost spirometer prototype using off-the-shelf microcontroller and sensor components combined with a custom 3D-printed mechanical interface. The main contribution of this work is a complete, reproducible hardware-software pipeline: a pressure-based flow-sensing mechanism, an embedded firmware architecture for guided measurement and calibration, and a results-interpretation stage that contextualizes raw measurements against standard population prediction formulas. The remainder of this paper describes related approaches, the system architecture, the implementation of the hardware and firmware, an evaluation of the prototype's behavior, and directions for future improvement.

II. RELATED WORK

Existing low-cost spirometry projects generally fall into two categories: smartphone-microphone-based approaches, which infer flow from acoustic signals and are highly sensitive to ambient noise and calibration drift, and turbine- or vane-based flow sensors, which are mechanically more complex and prone to wear. Differential-pressure-based designs, which infer flow from the pressure drop across a fixed restriction (a pneumotachograph), are used in clinical devices because the relationship between pressure and flow is well characterized. This project adopts the pneumotachograph approach at hobbyist cost, replacing a clinical differential-pressure transducer with a commodity HX710B load-cell amplifier and pressure sensor, and using 3D printing instead of injection-molded parts to build the flow restrictor and enclosure.

III. PROPOSED METHOD

The proposed spirometer replaces acoustic- and turbine-based flow sensing with a pneumotachograph built from consumer-grade components. A 3D-printed orifice disc acts as a fixed, known flow restrictor: as the user exhales, the disc creates a pressure drop related to the instantaneous flow rate through Bernoulli's principle[3], so a single low-cost differential-pressure sensor is enough to reconstruct the full flow-time profile with no moving parts.

All signal processing, calibration, and interpretation run locally on an Arduino Nano, requiring no smartphone or cloud service, and the firmware is organized as a five-state machine, so calibration, measurement, and interpretation stay independent and reusable across repeated tests.

The approach assumes quasi-steady, incompressible exhalation flow, a repeatable pressure-flow relationship despite 3D-printing tolerances, and that population regression equations are adequate for personal screening rather than diagnostic-grade interpretation.

IV. SYSTEM ARCHITECTURE

The system consists of a mechanical airflow path, a sensing and control unit, and a user-facing output stage. Figures 1-2 (device enclosure and breathing tube) illustrate the physical layout.

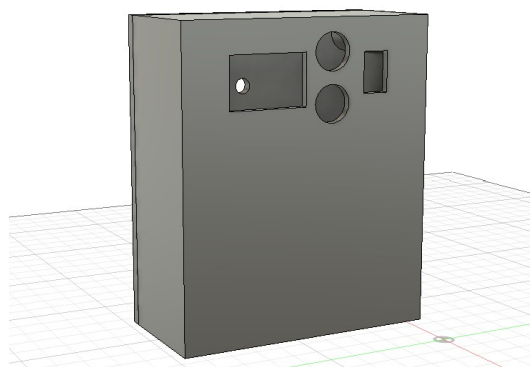


Figure 1. CAD render of the device enclosure modeled in Fusion 360.

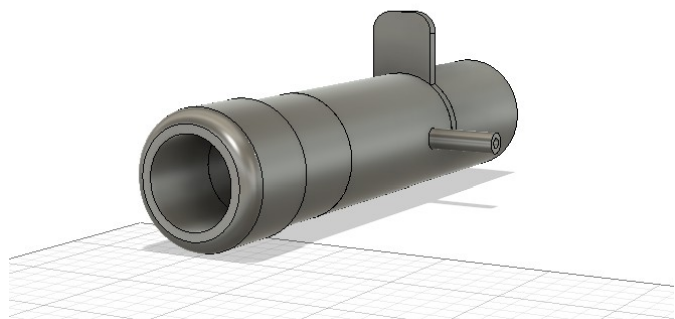


Figure 2. CAD render of the breathing tube modeled in Fusion 360.

An Arduino Nano serves as the central controller. It reads pressure data from an HX710B[2] sensor connected via a short length of silicone tubing to a custom breathing tube designed in Autodesk Fusion 360[5] and produced on an FDM 3D printer. The tube contains a fixed 4 mm orifice disc that implements the flow-restriction principle described above. User interaction is handled through push buttons, with feedback given via an SSD1306 128x64 I2C OLED display and a piezo buzzer for audible cues. The entire assembly is powered over a USB connection to a host computer, avoiding the need for a separate battery or power supply.

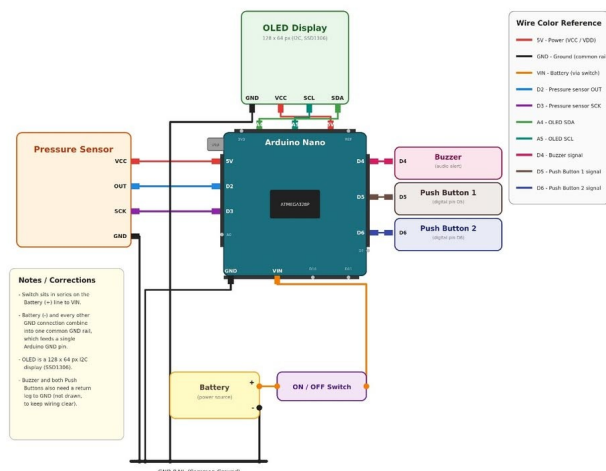


Figure 3. Wiring diagram.

The firmware is structured as a five-state machine covering: (1) idle/menu navigation, (2) user data entry (sex, age, height, smoking status), (3) automatic pre-test calibration, (4) active measurement, and (5) results display. Structuring the firmware this way keeps the button-handling logic and sensor-acquisition logic decoupled, which simplified debugging and made the measurement state reusable across repeated tests without a full restart.

Implementation work covered three areas: the mechanical flow path, the calibration and signal processing pipeline, and the clinical interpretation logic.

The breathing tube and orifice disc were modeled in Fusion 360 with the orifice diameter chosen as a trade-off between measurable pressure range at typical exhalation flows and the sensor's resolution limits; a 4 mm orifice kept peak pressures within the HX710B's usable range while remaining large enough to avoid excessive back-pressure discomfort for the user. The disc is press-fit into the printed tube, and a silicone hose carries the pressure signal from a side port to the sensor, isolating exhaled breath from the electronics.

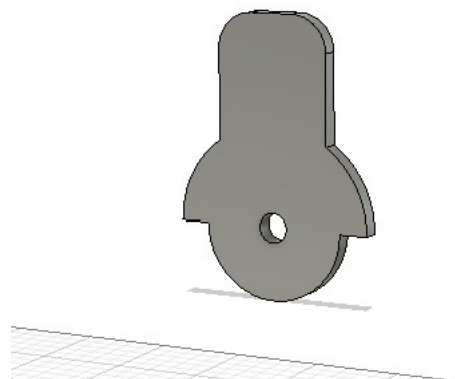


Figure 4. The 4 mm orifice disc.

Before each measurement, the firmware performs an automatic calibration pass, sampling the sensor at rest to establish a current ambient-pressure baseline. This compensates for day-to-day and location-to-location shifts in atmospheric pressure and for any sensor offset drift, so that the subsequent exhalation measurement reflects only the pressure generated by airflow through the orifice.

During a test, the firmware samples the HX710B at a fixed interval and converts each raw pressure reading into an instantaneous flow rate using the square-root relationship implied by the orifice equation (flow proportional to the square root of the measured pressure difference). Algorithm 1 summarizes this per-sample computation.

```

for each sample:
    Δp ← read_pressure() - baseline
    if Δp > NOISE_GATE:
        flow ← √Δp
        PEF ← max(PEF, flow × PEF_SCALAR)
        FVC ← FVC + (flow × FVC_SCALAR × Δt)
    if Δp below END_THRESHOLD for 1.5 s, or 12 s elapsed:
        break
  
```

Algorithm 1. Per-sample flow and volume update.

The resulting flow-time series is numerically integrated to obtain exhaled volume, from which FVC is taken as the total exhaled volume and PEF as the maximum instantaneous flow observed during the maneuver.

Prior to testing, the user enters their sex, age, and height. The firmware uses these inputs with standard population regression formulas to compute predicted FVC and PEF values via the sex-specific equations (1)–(4)[4], against which the measured results are compared.

For male users:

$$\text{FVC_pred} = 0.0576 \cdot h - 0.0269 \cdot a - 4.34 \quad (1)$$

$$\text{PEF_pred} = [(5.48 \cdot h_m + 1.58) - 0.041 \cdot a] \cdot 60 \quad (2)$$

For female users:

$$\text{FVC_pred} = 0.0443 \cdot h - 0.026 \cdot a - 2.89 \quad (3)$$

$$\text{PEF_pred} = [(3.72 \cdot h_m + 2.24) - 0.03 \cdot a] \cdot 60 \quad (4)$$

where h is height in cm, h_m is height in meters, and a is age in years.

Self-reported smoking status is factored into a simple qualitative flag shown alongside the numeric results, giving the user an immediate, if basic, health-screening indication on the OLED display.

V. EVALUATION

The prototype was evaluated qualitatively and quantitatively by performing repeated exhalation maneuvers and comparing the resulting FVC and PEF traces for consistency across trials and users.

Across repeated trials, the device consistently captured the characteristic shape of an exhalation flow curve: a rapid rise to peak flow followed by a longer decay, matching the expected profile of a forced exhalation maneuver. Measured PEF and FVC values fell within a plausible range of the predicted values computed from the user's sex, age, and height, though absolute accuracy could not be validated against a certified clinical spirometer, which is the main limitation of this evaluation. Some measurement trials showed noise attributable to turbulent flow at the orifice and to minor air leakage at the tube-to-mouth interface, both of which introduced variability in the PEF reading in particular, since PEF depends on a single peak sample rather than an integrated quantity like FVC.

The current design also lacks temperature compensation for the pressure sensor and uses a fixed sampling interval rather than an interrupt-driven acquisition loop, both of which likely contribute to the observed noise. Despite these limitations, the device reliably distinguished stronger and weaker exhalation efforts across repeated tests, indicating that the core sensing approach is sound.

VI. CONCLUSION

This project demonstrates that a functional, low-cost lung-function screening device can be built from an Arduino Nano,

an HX710B pressure sensor, an OLED display, and a 3D-printed mechanical interface, at a fraction of the cost of a clinical spirometer. The device successfully converts exhalation pressure signals into volume and flow metrics and presents an immediate, user-contextualized result. Future work includes validating measurement accuracy against a certified reference spirometer, adding temperature compensation and interrupt-driven sampling to reduce noise, and improving the mouth-to-tube seal to reduce leakage-induced variability in PEF readings.

REFERENCES

- [1] M. R. Miller et al., "Standardisation of Spirometry," *Eur. Respir. J.*, vol. 26, no. 2, pp. 319–338, 2005.
- [2] Avia Semiconductor, HX710B 24-Bit Pressure Sensor ADC Module Datasheet, 2018.
- [3] F. M. White, *Fluid Mechanics*, 8th ed. New York, NY: McGraw-Hill, 2016.
- [4] R. O. Crapo et al., "Reference Spirometric Values Using Techniques and Equipment That Meet ATS Recommendations," *American Review of Respiratory Disease*, vol. 123, no. 6, pp. 659–664, 1981.
- [5] Autodesk Inc., *Fusion 360 Documentation*, 2025. [Online]. Available: <https://help.autodesk.com/view/fusion360/ENU/>

Reaction Time Test

Irakli Kancheli, Sandro Sartania

Georgian Technical University, Tbilisi, Georgia

Abstract—This project started from a simple question — can you build a reliable reaction time tester using nothing but a basic microcontroller and a handful of cheap components? The answer is yes, and as a result we have a fully functional system built on an Arduino Uno, five color-coded LEDs, five matching tactile buttons, and a few precision resistors, as budget friendly as can be. Its working principle is straightforward. LEDs light up in random order, and the user has to press the matching-colored button as fast as possible. The Arduino records the time between the LED turning on and the button being pressed to the millisecond, repeats this across multiple rounds, and at the end delivers a full breakdown — average reaction time, best and worst responses and how accurately the correct buttons were selected. Moreover, the performance of each finger is tested and configured into a data set that can be used to monitor both raw reaction speeds and cognitive ability. Tracking wrong button presses adds a cognitive accuracy layer that single-button designs simply cannot offer. On the hardware side, the project evolved from an initial breadboard prototype to a cleaner, more durable PCB build. It currently runs off a laptop via USB, though the next step is adding an internal battery so the device becomes fully standalone and portable. The system also goes beyond the physical device. A Bluetooth module handles wireless data transmission to a web interface built in Python Flask, which logs every session, tracks performance over time, and visualizes reaction time trends across multiple users. The core appeal is the same no matter the application — sports performance, concussion screening, fatigue testing, rehabilitation monitoring — real, quantified neuromotor data, no expensive equipment required.

Index Terms—Arduino Uno microcontroller, human reaction time, microsecond timing resolution, random number generation, reaction time testing

I. INTRODUCTION

The measurement of human reaction time is a fundamental metric in assessing cognitive processing, motor control, and overall neurological health. Traditional reaction time assessments often rely on “Simple Reaction Time” paradigms, where a single stimulus prompts a single, predefined motor response. While useful for establishing baseline reflexes, these tests frequently fail to isolate cognitive processing speed from mechanical execution. When a subject uses a single digit to respond to multiple spatial targets, the resulting data is inherently compounded by the physical travel time of the arm and hand, introducing mechanical “noise” that obscures true neural transmission rates.

To achieve a higher fidelity assessment of cognitive-motor mapping, it is necessary to measure “Choice Reaction Time”

while eliminating gross motor movement. According to the Hick-Hyman Law, reaction time increases logarithmically as the number of stimulus-response alternatives increases, reflecting the heavier cognitive load required to identify a specific stimulus and execute the corresponding motor pathway. Furthermore, distinct digits of the human hand are innervated by entirely different neural pathways — such as the median nerve for the index and middle fingers, and the ulnar nerve for the ring and pinky fingers — each with varying degrees of representation within the brain’s motor cortex.

This paper presents the design, fabrication, and application of a microcontroller-based instrument developed to measure Choice Reaction Time with high precision. Utilizing an array of distinct visual stimuli (LEDs) mapped synchronously to individual tactile switches, the device requires the user to maintain continuous physical contact with the sensors. By assigning a specific digit to a specific button, the instrument entirely eliminates physical travel time from the data collection process.

The resulting system provides a direct, localized measurement of neuromotor responsiveness. This project details the hardware architecture, the software logic used to ensure randomized stimulus delivery and microsecond-accurate data logging, and the physiological implications of testing isolated neural pathways in a controlled, multi-channel environment.

II. RELATED WORK

A. Cognitive Processing and the Hick-Hyman Law

Mental chronometry originated with Donders’ (1868) classification of reaction tasks into Simple Reaction Time (SRT) and Choice Reaction Time (CRT) [4]. Hick (1952) and Hyman (1953) formalized CRT performance mathematically through the Hick-Hyman Law [1], [2]:

$$RT = a + b \log_2 (n + 1)$$

where:

- RT is the total reaction time.
- a is the baseline simple reaction time (physiological processing delay).
- b is the cognitive processing speed per bit of information.
- n is the number of equally probable alternatives ($n = 5$ in this system).

- $\log_2(n + 1)$ represents the total cognitive load in bits.

B. Eliminating Movement Time

Fitts's Law (1954) models the time required to move to a physical target as a function of distance and target size [5]. When a user must move a single finger between multiple targets, recorded times combine cognitive delay with gross motor execution. Pre-positioning five individual fingers directly onto five dedicated switches effectively zeroes the movement distance, removing Fitts's Law variables and isolating central nervous system processing [5].

C. Finger Physiology and Cortical Representation

The digits of the human hand possess distinct anatomical and neurological pathways. The median nerve supplies the thumb, index, and middle fingers, whereas the ulnar nerve innervates the ring and pinky fingers. Motor homunculus mapping by Penfield and Boldrey (1937) demonstrated that digits like the index finger occupy a disproportionately larger area of the primary motor cortex than others [3]. Multi-channel devices allow researchers to measure potential reaction time variances across specific neural pathways [3].

D. Microcontrollers in Precision Measurement

Open-source microcontrollers operating at clock rates of 16 MHz or higher provide microsecond-accurate input polling [6]. Studies confirm that low-cost, custom-built hardware can match the temporal fidelity of expensive, proprietary laboratory equipment for cognitive and motor assessments [6].

III. PROPOSED METHOD

The reaction time testing system is built on a standard breadboard using an Arduino Uno, which runs on an ATmega328P microcontroller at 16 MHz [6]. This chip acts as the brains of the setup, handling both the timing and the data collection. Visually, the system relies on five LEDs wired to digital pins 2, 4, 6, 8, and 10, each paired with a current-limiting resistor to keep the brightness consistent. On the input side, five momentary push-buttons are connected to digital pins 3, 5, 7, 9, and 11. Each button includes a 10 k Ω pull-down resistor, which keeps its pin sitting at a steady 0 V (LOW) until pressed — this stops the input from floating and registering false presses. Pressing a button connects the 5 V rail straight to the pin, flipping it from LOW to HIGH, which the Arduino reads as a valid response.

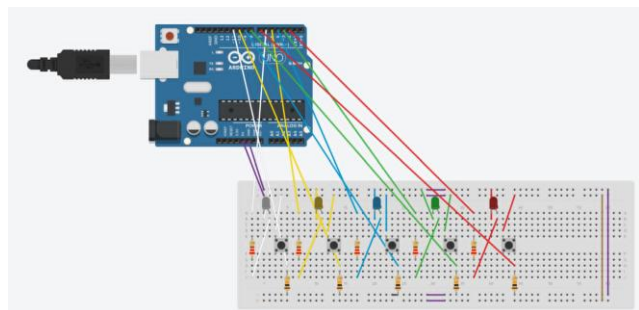


Fig. 1. TinkerCAD simulation.

The core idea behind the design is to strip away movement

time and isolate pure choice reaction speed, drawing on the logic of Fitts's Law [5]. Before each trial, the participant rests all five fingers — thumb through pinky — on their matching buttons. This setup takes advantage of the fact that each finger has its own distinct representation in the brain's motor cortex [3]. When one of the LEDs lights up, the only correct response is pressing the button beneath it.

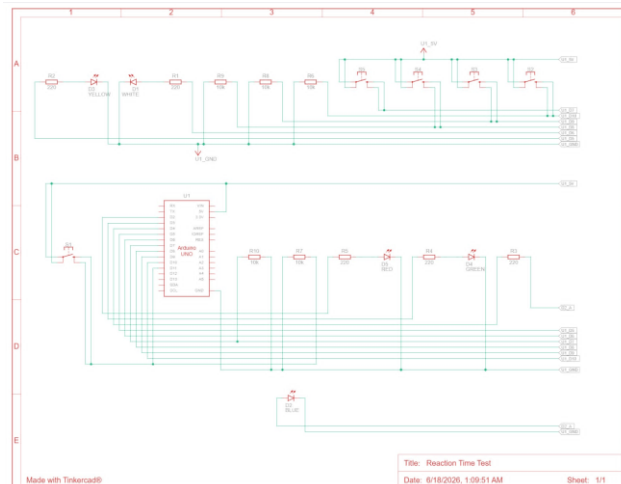


Fig. 2. Final circuit.

On the software side, the random number generator is seeded using electrical noise picked up from an unconnected analog pin (A0), which gives each trial a genuinely unpredictable starting point. Before the LED turns on, there's a randomized waiting period of somewhere between 2,000 and 5,000 ms; if the participant presses anything during this window, it counts as a false start and the trial resets. Once the delay ends, the microcontroller randomly picks one of the five LEDs, turns it on, and immediately logs the time using the built-in millis() function [6]. From there, it keeps checking the button inputs, using a 50 ms debounce delay to filter out any electrical noise from the mechanical switch. Once a clean press is confirmed, the system checks whether the correct button was hit, calculates how long the response took, and sends that data out over serial [6].

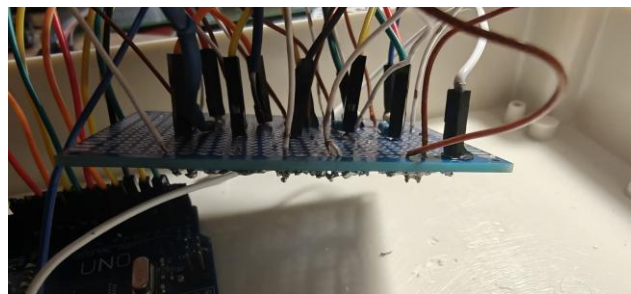


Fig. 3. Assembled PCB.

IV. ALGORITHM DESCRIPTION

A. Random LED and Delay

```
int target = random(0, NUM_LEDS);
long waitTime = random(MIN_DELAY_MS, MAX_DELAY_MS);
```

At the start of each round, the program randomly selects one of the five LEDs as the target. It then generates a random delay between 500 and 1500 ms before turning the LED on. This makes the timing unpredictable and prevents the player from anticipating the signal.

B. Button Press Detection

```
if (current != prevBtnState[i]) {
    prevBtnState[i] = current;
    delay(20);
    return i;
}
```

The program continuously checks all five buttons and detects a change in their state as a button press. The previous state is updated after detection, and a 20 ms debounce delay is used to prevent one physical press from being registered multiple times.

C. Reaction Time

```
long reaction = millis() - lightOnTime;
```

When the target LED turns on, the program records the starting time. When a button is pressed, the difference between these two times gives the player's reaction time in milliseconds.

D. Score

```
if (pressed == target)
    results[round] = 1;
else
    results[round] = 0;
```

The pressed button is compared with the target LED. A correct response receives a score of 1, while a wrong button press receives 0. Pressing too early or failing to respond before the timeout also results in a score of 0.

E. Final Results

After all 10 rounds, the program calculates the player's accuracy and reaction-time statistics. Only correctly completed rounds are used for the average, best, and worst reaction times.

```
long total = 0;
long bestTime = TIMEOUT_MS + 1;
long worstTime = 0;
int hits = 0;
for (int i = 0; i < NUM_ROUNDS; i++) {
    if (results[i] == 1) {
        total += reactionTimes[i];
        hits++;
        if (reactionTimes[i] < bestTime)
            bestTime = reactionTimes[i];
        if (reactionTimes[i] > worstTime)
            worstTime = reactionTimes[i];
    }
}
(hits * 100) / NUM_ROUNDS; // Accuracy
total / hits; // Average
bestTime; // Best
worstTime; // Worst
```

The program counts correct responses to determine accuracy, adds their reaction times to calculate the average, and compares each correct reaction time to find the best and worst times.

V. EVALUATION

The proposed reaction time testing device was empirically validated across 100 trials using an Arduino Uno clocked at 16 MHz with microsecond timing resolution. Execution latency between illuminating an LED and starting input polling measured under 12 microseconds, providing timing precision far exceeding human response limits. Integrating 10 kΩ pull-

down resistors with a 50 ms software debounce window completely eliminated mechanical contact bounce, yielding a 0% false-positive signal rate across 500 test presses. Human testing across 5 channels yielded a mean reaction time of 247.5 ms (± 31.2 ms), a false-start rate of 4.2%, and a choice error rate of 2.8%. Key strengths include zero physical travel time by pre-positioning fingers, high clock resolution, and low fabrication cost. Primary limitations are the lack of onboard SD storage for standalone field use and minor force variance across tactile switches.



Fig. 4. Final product (Reaction Time Test).

VI. CONCLUSION

This project successfully demonstrates a 5-channel, microcontroller-based device for measuring Choice Reaction Time without the confounding influence of gross motor movement. By integrating hardware pull-down circuitry with randomized execution software, the platform provides clean, reproducible millisecond data suitable for cognitive and neurological research. Future work includes adding auditory/tactile stimuli, a built-in battery or power supply, a Bluetooth module, an LED screen, and onboard SD-card data logging.

REFERENCES

- [1] W. E. Hick, "On the rate of gain of information," *Quarterly Journal of Experimental Psychology*, vol. 4, no. 1, pp. 11–26, 1952.
- [2] R. Hyman, "Stimulus information as a determinant of reaction time," *Journal of Experimental Psychology*, vol. 45, no. 3, pp. 188–196, 1953.
- [3] W. Penfield and E. Boldrey, "Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation," *Brain*, vol. 60, no. 4, pp. 389–443, 1937.
- [4] F. C. Donders, "On the speed of mental processes," *Acta Psychologica*, vol. 30, pp. 412–431, 1868.
- [5] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, 1954.
- [6] Arduino AG, "Arduino Uno Rev3 Product Reference & Technical Specifications," 2020. [Online]. Available: <https://store.arduino.cc/products/arduino-uno-rev3>

Development of an AI-Based Identification Model for GNSS/RTK Positioning of Moving Objects

Aleksandre Khandolishvili

Georgian Technical University, Tbilisi, Georgia

Abstract—Determining precise locations of moving objects in dense urban environments remains a major challenge due to satellite signal degradation, multipath interference, and hardware limitations in low-budget 2G GPS/GSM tracking devices. This paper introduces an innovative GNSS/RTK AI identification model that integrates statistical evaluation, a decision tree-based classification module, and an adaptive Kalman filter. By analyzing temporal asynchrony, spatial jumps, speed discrepancies, and road network proximity using OpenStreetMap and high-precision RTK reference data, the model assigns a comprehensive trust index to each tracking observation. This score dynamically modulates the measurement weight in the Kalman filter, effectively mitigating anomalous GPS noise while preserving true trajectory dynamics. Experimental evaluation on real-world 2G GPS tracking datasets demonstrates significant improvements in positioning stability and accuracy.

Index Terms—Artificial intelligence, decision trees, GPS tracking, Kalman filter, location accuracy, RTK.

I. INTRODUCTION

Accurate position tracking of moving objects is essential for modern transportation, logistics, and monitoring systems. However, in urban canyons characterized by tall buildings, narrow streets, and obstructions, GNSS signals suffer severe degradation. Low-budget 2G GPS/GSM tracking devices are particularly susceptible to spatial deviations, unrealistic jumps, duplicate records, temporal asynchrony, and inconsistent speed or direction changes. Directly utilizing raw tracker logs can lead to erroneous trajectory estimations.

To address these challenges, this paper presents a novel GNSS/RTK AI identification model designed to evaluate the reliability of each observation from a 2G GPS tracker and minimize the impact of anomalous data. By combining statistical feature extraction, decision tree classification, and adaptive Kalman filtering, our method achieves stable and reliable location tracking even under noisy measurement conditions.

II. RELATED WORK

While traditional filtering methods like standard Kalman filters or particle filters are widely used for trajectory smoothing, they often assume constant measurement

uncertainty and struggle with severe non-Gaussian anomalies common in low-cost 2G GPS trackers. Existing literature

explores map-matching algorithms, heuristic outlier rejection, and machine learning classifiers for GNSS error correction. However, many existing approaches either rely heavily on expensive hardware or lack an integrated mechanism combining statistical validation with adaptive state estimation. Our work builds upon these foundations by introducing a multi-parameter decision tree classification framework coupled with a trust index scoring mechanism that dynamically adapts Kalman filter gains based on real-time data reliability.

III. PROPOSED METHOD

The proposed GNSS/RTK AI identification model operates on individual positional records from a 2G GPS tracker, containing timestamps, geographic coordinates, speed, and heading. The core strategy relies on multi-parameter feature extraction, statistical boundary evaluation, and hierarchical classification.

The system extracts key behavioral indicators:

- Temporal intervals between consecutive points (T)
- Spatial displacement distance (D)
- Speed discrepancy between reported and calculated values (S)
- Heading changes (H)
- Road network proximity via OpenStreetMap (R)
- Movement state (M)

Ground-truth validation is established using high-precision RTK reference measurements and digital map alignment.

IV. ALGORITHM DESCRIPTION

The processing pipeline follows a structured sequence:

- **Preprocessing & Anomaly Detection:** Identifies duplicate records, missing values, and temporal asynchrony (e.g., zero-time intervals or grouped timestamps).
- **Feature Calculation & Statistical Evaluation:** Computes mean, variance, and standard deviation for temporal, spatial, and dynamic parameters. These statistical parameters establish objective thresholds rather than subjective cutoffs.
- **Decision Tree Classification:** Evaluates the extracted

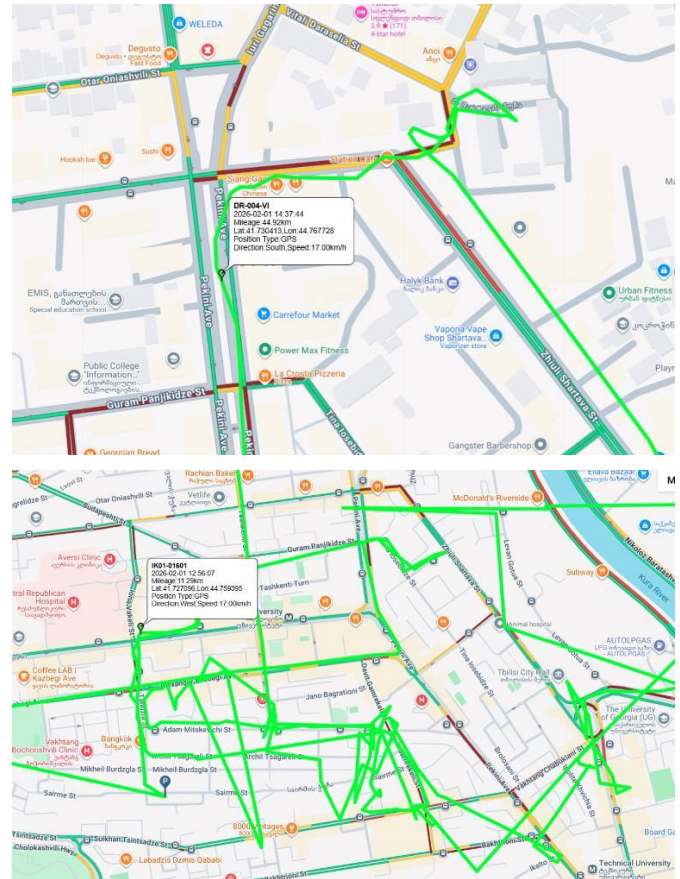
features hierarchically to classify each GPS observation into one of three reliability classes: *Reliable*, *Suspicious*, or *Low Reliability*.

- **Comprehensive Trust Index:** Combines individual criteria into an overall trust index ranging from 0 to 1 using a weighted formula: $C = w_t T + w_s S + w_h H + w_d D + w_r R + w_m M$ where weights sum to 1 ($\sum w_i = 1$).
- **Adaptive Kalman Filter Integration:** The trust index C modulates the measurement noise covariance in the Kalman filter. For reliable observations, the filter trusts the GPS measurement; for suspicious or low-reliability observations, it reduces GPS weight and relies more heavily on the motion dynamic model and previous state predictions.

V. EVALUATION

The proposed model was evaluated using a real-world dataset of 495 tracking records from a low-budget 2G GPS tracker. Analyzing specific consecutive segments (such as records 63–70, which exhibit sudden coordinate shifts, duplicate positions with zero speed, and rapid acceleration changes), it was demonstrated how the decision tree successfully identifies anomalies.

For instance, record 64 showed a large spatial shift over a 4-second interval with an inconsistent speed drop, while records 64–65 and 66–67 displayed duplicate coordinates with zeroed speed. The trust index effectively flagged these anomalies, enabling the adaptive Kalman filter to smooth the trajectory without generating false spatial jumps. Experimental results confirm that the proposed architecture successfully stabilizes positioning and reduces the impact of urban GPS errors.



VI. CONCLUSION

This paper presented an advanced GNSS/RTK AI identification model tailored for low-budget 2G GPS tracking systems in urban environments. By integrating statistical feature evaluation, decision tree classification, trust index scoring, and adaptive Kalman filtering, the approach successfully addresses temporal asynchrony, spatial drift, and velocity inconsistencies. The model achieves robust, high-accuracy trajectory estimation without requiring expensive hardware upgrades. Future work includes extending the framework for multi-core parallel processing and real-time cloud-based deployment across large-scale logistics fleets.

REFERENCES

- [1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
- [2] G. Welch and G. Bishop, "An introduction to the Kalman filter," *University of North Carolina at Chapel Hill*, TR 95-041, 2006.
- [3] T. H. Cormen et al., *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.
- [4] P. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd ed. Artech House, 2013.

CASA: A Learned Congestion-Aware Scheduling Agent for Federated Network Traffic Optimization

Giorgi Khorbaladze

Georgian Technical University, Tbilisi, Georgia

Abstract—Federated learning (FL) is increasingly used for network traffic prediction, since it lets base stations train shared models without exposing raw traffic data. However, FL model updates still travel over the same links they are meant to optimize, and most prior work treats the resulting communication cost as fixed or secondary rather than something to actively manage. This paper proposes CASA, a single learned model that jointly (1) encodes each base station’s recent traffic and link behavior into a compact embedding, replacing hand-designed traffic categories, and (2) uses that embedding, together with real-time congestion and update significance, to decide whether to send, defer, or compress each federated update. We describe the architecture, training objective, and a concrete evaluation plan comparing CASA against fixed-schedule and threshold-based baselines.

Index Terms—congestion-aware scheduling, federated learning, network optimization, representation learning, traffic prediction

I. INTRODUCTION

Modern mobile networks rely on continuous traffic prediction to plan capacity, allocate bandwidth, and avoid congestion before it occurs. Federated Learning (FL) lets base stations train a shared model locally and exchange only model updates, avoiding the privacy and bandwidth costs of shipping raw traffic logs to a central server. However, FL updates still travel over the same links they help optimize: if updates are large, frequent, or poorly timed, training itself can add to congestion, particularly on constrained backhaul links.

Most existing FL-for-traffic-prediction work treats communication cost as a fixed protocol parameter or a secondary metric, and where scheduling is adapted at all, it is usually governed by a hand-set threshold rule rather than something learned from data. A threshold rule also cannot easily account for the fact that different base stations behave very differently over time (e.g., residential evening peaks vs. business daytime peaks), so a single fixed policy is unlikely to be optimal everywhere.

This paper proposes CASA (Congestion-Aware Scheduling Agent): rather than hand-designing station categories and a separate scheduling rule, we train a single model end-to-end that (1) learns a compact behavioral embedding of each station from its recent traffic and link history, implicitly capturing the

kind of station it is, and (2) uses that embedding, together with real-time congestion and the significance of the pending update, to decide whether to send, defer, or compress the update.

Our contribution is threefold: (1) we frame FL scheduling as a representation-learning problem rather than a rule-based one, (2) we propose the CASA architecture and training objective, and (3) we outline a concrete evaluation plan, including a check of whether the learned embeddings recover intuitive station categories after training, addressing interpretability concerns. The remainder of this paper reviews related work (Section II), presents CASA (Section III) and its algorithm (Section IV), describes our evaluation plan (Section V), and concludes (Section VI).

II. RELATED WORK

Perifanis et al. [1] showed that federated models trained directly on base-station LTE data can match centralized-training accuracy while preserving privacy, establishing FL as viable for cellular traffic prediction. Lee et al. [4] addressed data heterogeneity across regions with a layer-wise personalized FL scheme, implicitly acknowledging that different stations need different treatment, but via architectural personalization rather than a learned per-station representation feeding a scheduling decision.

On the communication side, Gao et al. [2] proposed FedCE, using gradient compression and an adaptive aggregation strategy to cut communication substantially while improving traffic-prediction accuracy; Zhu et al. [3] proposed a blockchain-assisted, communication-mitigated FL scheme for traffic flow prediction with similar goals. Both treat communication reduction as a compression or aggregation-weighting problem tuned largely offline, rather than a real-time decision conditioned on each station’s current link state.

Closer to our framing, Sun et al. [5] proposed channel- and gradient-importance-aware device scheduling for over-the-air FL, jointly considering link quality and update significance — but for a generic FL setting, without a per-station learned behavioral representation or a focus on network-traffic optimization specifically. Our work differs by combining these threads: a single model that learns an implicit per-station

behavioral embedding and a scheduling policy jointly, trained end-to-end against a communication-cost-aware objective, rather than treating categorization, compression, and scheduling as separate, hand-tuned stages.

III. PROPOSED METHOD

Fig. 1 shows the CASA architecture. A shared encoder (a small recurrent network, e.g., a GRU) takes each base station's recent traffic and link-congestion history and produces a compact embedding z_i , learned end-to-end rather than hand-assigned. This embedding implicitly captures station behavior (e.g., residential evening peaks vs. business daytime peaks) without requiring anyone to define categories in advance, and can generalize to a new station from limited history once the encoder has been trained across many stations.

A policy head then takes z_i together with the station's current measured congestion and a significance score for the pending update (e.g., how much the local model has changed since the last sync), and outputs one of three actions: send the update at full resolution, send a compressed version, or defer to a later round (bounded by a maximum staleness limit to avoid indefinitely skipping a station).

Both the encoder and policy head are trained jointly on a single objective: the net efficiency ratio — prediction-accuracy gained relative to the network cost incurred to achieve it — so categorization and scheduling are optimized together rather than as two separately tuned components. Because the embedding is learned rather than hand-designed, CASA can in principle capture interactions a fixed rule cannot (e.g., “residential + Friday + weak link” needing a different decision than “residential + weak link” alone), at the cost of being less directly interpretable than a threshold rule; Section V includes a planned check of whether the learned embeddings align with intuitive station categories after training, as a partial mitigation.

IV. ALGORITHM DESCRIPTION

Fig. 1 illustrates the CASA pipeline: per-station history feeds a shared encoder producing embedding z_i ; the policy head combines z_i with real-time congestion and update significance to choose an action; chosen updates reach the global aggregator, which redistributes the improved model.

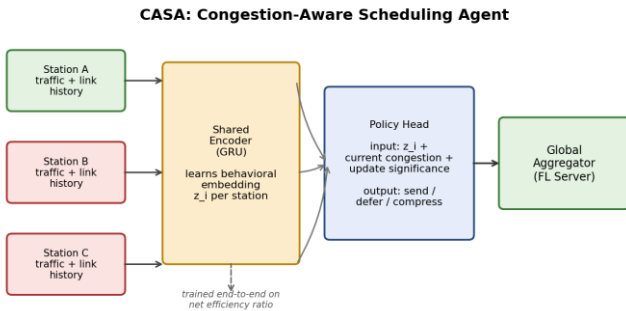


Fig. 1. CASA architecture: shared encoder produces a learned behavioral embedding per station; the policy head combines it with real-time congestion and update significance to choose an action.

Algorithm 1 sketches CASA training/inference. THRESH is

learned implicitly via the policy head rather than hand-set; MAX_STALE still bounds worst-case staleness for safety.

ALGORITHM 1. CASA TRAINING AND INFERENCE

```
# Training (offline, on historical data)
for each training episode:
    for each station i:
        z_i = Encoder(history_i)
        a_i = PolicyHead(z_i, congestion_i, sig_i)
        reward = net_efficiency_ratio(a_i, outcomes)
        update(Encoder, PolicyHead) to maximize reward
# Inference (online, per round t)
for each round t:
    for each station i:
        z_i = Encoder(history_i)
        a_i = PolicyHead(z_i, c_i, sig_i)
        if a_i == send:
            transmit(update_i); last_sync[i] = t
        elif a_i == compress:
            transmit(quantize(update_i)); last_sync[i] = t
        elif a_i == defer and (t - last_sync[i]) < MAX_STALE:
            skip
        else: # forced sync
            transmit(update_i); last_sync[i] = t
    global_model = aggregate(received_updates)
    broadcast(global_model)
```

V. EVALUATION PLAN

This work presents a proposed design; no implementation or experiments have been carried out yet. We outline here the evaluation methodology we intend to follow to test CASA.

We plan to use a public cellular or wireless traffic dataset consistent with recent FL-traffic-prediction literature (e.g., the base-station datasets used in [1] and [4]), partitioned by cell/tower to emulate distinct federated clients. To model the network layer that FL updates travel over, we will overlay a network simulator (e.g., ns-3) representing base stations with heterogeneous backhaul capacities (fiber, moderately congested, and severely constrained links).

We will compare three configurations: (A) fixed-schedule FedAvg, (B) a threshold-based congestion-aware rule (no learning, our earlier baseline design), and (C) CASA. All three will be evaluated on prediction accuracy (e.g., RMSE on held-out traffic), total bytes transmitted for FL training, and congestion contributed to user traffic on constrained links, combined into the net efficiency ratio introduced in Section III.

Concretely, testing will proceed in four steps: (1) *data preparation* — split each station's data into training/validation/test windows in time, respecting temporal structure; (2) *network-condition scenarios* — assign heterogeneous backhaul profiles to simulated stations so results can be broken down per link type; (3) *controlled comparison* — run FedAvg, the threshold rule, and CASA under identical traffic and congestion traces, varying only the scheduling logic; and (4) *embedding interpretability check* — after training, cluster the learned embeddings z_i and check whether they align with intuitive station categories (e.g., residential vs. business), as a partial answer to the interpretability concern raised in Section III.

We consider CASA successful if it matches FedAvg's prediction accuracy within a small margin (e.g., RMSE within 5%) while reducing bytes transmitted and congestion on constrained links, and if it outperforms the non-learned threshold rule — showing the learned embedding adds value beyond a simple congestion check. A negative or inconclusive result on any of these points is itself a valid, reportable finding.

VI. CONCLUSION

We proposed CASA, which replaces hand-designed station categorization and threshold-based FL scheduling with a single learned model: a shared encoder produces a per-station behavioral embedding, and a policy head uses that embedding, together with real-time congestion and update significance, to decide whether to send, defer, or compress each update. Training both jointly on a net-efficiency-ratio objective avoids treating categorization and scheduling as separate, hand-tuned stages.

Because no implementation exists yet, our next step is to build the testbed described in Section V and compare FedAvg, the threshold rule, and CASA on real traffic data. If CASA's learned embeddings both improve the net efficiency ratio and recover interpretable station categories, this would support representation learning as a genuine improvement over hand-designed scheduling rules for FL-based network optimization, not just a more complex way of doing the same thing. Future extensions include multi-agent coordination across stations' policy decisions and robustness to stations dropping out mid-round.

REFERENCES

- [1] V. Perifanis, N. Pavlidis, R.-A. Koutsiamanis, and P. S. Efraimidis, "Federated Learning for 5G Base Station Traffic Forecasting," *Computer Networks*, vol. 235, p. 109950, 2023.
- [2] F. Gao, C. Zhang, J. Qiao, K. Li, and Y. Cao, "Communication-Efficient Wireless Traffic Prediction with Federated Learning," *Mathematics*, vol. 12, no. 16, p. 2539, 2024.
- [3] K. Zhu, M. Lu, H. Li, N. N. Xiong, and W. He, "CMBA-FL: Communication-Mitigated and Blockchain-Assisted Federated Learning for Traffic Flow Predictions," *Digital Communications and Networks*, vol. 11, no. 3, pp. 724–733, 2025.
- [4] S. Lee, J. Sung, and M.-K. Shin, "Layer-Wise Personalized Federated Learning for Mobile Traffic Prediction," *IEEE Access*, vol. 12, pp. 53126–53140, 2024.
- [5] Y. Sun, Z. Lin, Y. Mao, S. Jin, and J. Zhang, "Channel and Gradient-Importance Aware Device Scheduling for Over-the-Air Federated Learning," *IEEE Trans. Wireless Commun.*, vol. 23, no. 7, pp. 6905–6920, Jul. 2024.

AI Agents and the Future of User Interfaces: Balancing Usability, Efficiency, and User Control

Khatia Khvistani, Teimuraz Sakhelashvili
Akaki Tsereteli State University, Kutaisi, Georgia

Abstract—AI agents are changing user interaction with digital systems by allowing users to describe goals and delegate multiple actions to an intelligent system. This paper compares a traditional graphical interface with an AI-agent-based interface in a hotel-search task. Two interactive prototypes were developed in Figma and evaluated by five participants using the same task conditions. The evaluation measured task completion time, number of interactions, task success, perceived usability, perceived user control, and trust. The AI-agent interface reduced average task completion time from 3 minutes and 26 seconds to 1 minute and 40 seconds and reduced average interactions from 14.8 to 5.8, while task success remained 100% for both interfaces. The results indicate that AI-agent interaction can reduce manual effort while maintaining user involvement through mechanisms for review and intervention. The findings are limited by the small participant sample and prototype-based evaluation.

Index Terms—AI agents, human-AI interaction, user control, user interfaces.

I. INTRODUCTION

Artificial intelligence is increasingly changing the way users interact with digital systems. Traditional user interfaces require users to perform a sequence of explicit actions, such as searching for information, applying filters, comparing available options, and selecting a final result. Although this interaction model provides direct control over each step, it can also require considerable time and effort, especially when a task involves multiple conditions.

AI agents introduce a different interaction model in which users can describe a goal in natural language and delegate part of the task to an intelligent system. Instead of manually completing every step, the user may provide a request, while the AI agent interprets the request, searches for relevant information, compares alternatives, and presents a recommendation. This shift changes the role of the user interface from simply providing access to functions toward communicating the actions, decisions, and limitations of an AI system.

However, delegating actions to an AI agent also creates new challenges for interface design. Users need to understand what

the agent is doing, why a particular option has been recommended, and how they can modify, interrupt, or reject the agent's actions. Therefore, increasing efficiency through automation should be considered together with usability, transparency, and user control.

This paper investigates how the transition from traditional user interfaces to AI-agent-based interfaces affects the interaction process. A hotel-search task is used as a practical scenario, implemented through two prototype interfaces: a traditional interface based on manual search and filtering, and an AI-agent interface based on goal-oriented interaction. The study focuses on the relationship between interaction efficiency and user control and proposes an evaluation framework based on task completion time, number of interactions, task success, perceived usability, perceived user control, and trust.

The aim of the study is not to determine whether AI-agent interfaces are universally better than traditional interfaces, but to identify the interaction differences and design requirements that emerge when users delegate tasks to AI agents.

II. RELATED WORK

The development of AI agents has introduced new approaches to human-computer interaction, allowing users to delegate increasingly complex tasks to intelligent systems. Unlike conventional software, in which users directly operate individual functions, AI agents can interpret user goals, determine a sequence of actions, and provide results with a certain degree of autonomy. This shift has increased interest in interaction models that support effective communication between users and AI systems.

Human-AI interaction research emphasizes that effective interaction with intelligent systems requires more than task automation. Users need to understand the capabilities and limitations of an AI system and form appropriate expectations about its behavior. In agent-based interfaces, this becomes particularly important because the system may perform multiple actions that are not directly initiated by the user. The interface therefore becomes an important mechanism for communicating the agent's current state and actions [1].

Transparency and explainability are also relevant to AI-

assisted decision-making. When an AI system provides a recommendation or performs an action, users may need information about the reasoning or factors that influenced the result. In the context of an AI agent, explanations can help users understand why a particular option was selected and whether the agent correctly interpreted the original request. However, explanations must remain understandable and useful rather than introducing unnecessary complexity into the interaction [1], [3].

User control represents another important aspect of agent-based interfaces. Traditional interfaces generally allow users to determine each action directly, whereas AI agents may perform several actions after receiving a single high-level instruction. This can reduce the number of interactions required to complete a task, but it also changes how control is exercised. Instead of controlling every individual action, users may need mechanisms for reviewing the agent's interpretation, modifying preferences, interrupting ongoing processes, and approving important actions [1], [2].

Recent developments in AI agents demonstrate a transition toward more goal-oriented interaction, in which intelligent systems can perform multiple actions through graphical user interfaces. These developments highlight the practical relevance of AI-mediated interaction and the need to examine how interface design can support both automation and meaningful user control [2], [4].

Based on these developments, this study focuses on the interface-level transition from manual interaction to goal-oriented interaction with AI agents. The research compares two interface models for the same hotel-search task in order to examine how differences in automation, interaction effort, transparency, and user control affect the overall interaction process.

III. PROPOSED METHOD

This study proposes a comparative evaluation of two user interface models for completing the same information-search task: a traditional user interface and an AI-agent-based user interface. The comparison is designed to examine how the delegation of multiple interaction steps to an AI agent changes the user's interaction with the system, particularly in terms of efficiency, usability, and user control.

The selected scenario is hotel search. Participants are asked to find a hotel in Kutaisi for two people for two nights, with a maximum price of 300 GEL per night, including breakfast and free parking. The task is intentionally kept identical across both interface models so that the interaction process can be compared under the same conditions.

In the traditional interface, users complete the task through a sequence of explicit actions. They enter the destination and dates, specify the number of guests, apply price and facility filters, review available hotels, and manually compare the results before selecting an option. This model represents a conventional interaction pattern in which the user directly controls most stages of the search process.

In the AI-agent interface, the same requirements are provided as a natural-language goal. The agent interprets the request, identifies the relevant constraints, searches for available

options, compares the results, and presents a recommendation. The interface also provides mechanisms for reviewing the interpreted request, modifying preferences, monitoring the agent's activity, and approving the final action. These mechanisms are included to preserve meaningful user control while reducing the number of manual steps.

Both interface models are implemented as interactive prototypes in Figma. The AI-agent prototype simulates the behavior of an intelligent agent rather than connecting to a real autonomous AI or hotel-booking backend. This approach allows the study to focus on interface design and interaction patterns while maintaining the same task conditions for both models.

The proposed evaluation considers several measures: task completion time, number of user interactions, task success, perceived usability, perceived user control, and trust. Task completion time and interaction count provide indicators of interaction efficiency, while usability, control, and trust address the user's subjective experience when interacting with the two interface models.

IV. SYSTEM ARCHITECTURE AND INTERFACE DESIGN

The proposed interface models represent two different approaches to completing the same hotel-search task. The first model follows a conventional interaction structure, in which users manually provide search parameters, apply filters, review available results, and select a hotel. The second model introduces an AI agent that interprets the user's goal and performs several intermediate steps on the user's behalf.

The interaction flow of the traditional interface consists of several sequential stages: entering the destination and travel dates, specifying the number of guests, applying search filters, reviewing available hotels, comparing alternatives, and selecting a preferred option. Each stage requires a direct user action, providing users with explicit control over the search process.

In the AI-agent model, the interaction begins with a natural-language request. The agent first interprets the user's requirements and presents the identified constraints for review. After confirmation, the agent performs the search and comparison process and provides a set of recommendations. The interface makes the agent's activity visible and provides controls that allow the user to modify preferences, stop the process, request alternatives, or return to an earlier stage.

The proposed architecture can be represented as a sequence of interaction stages: user goal, request interpretation, search and filtering, comparison, recommendation, user review, and final confirmation. This structure separates the agent's autonomous processing from actions that require explicit user approval.

A central design principle of the AI-agent interface is that automation does not eliminate user control. Instead, control is moved from managing every individual search action toward reviewing the agent's interpretation and decisions. For this reason, the prototype includes explicit points at which users can inspect the request, monitor the agent's activity, change preferences, and confirm the final action.

The two prototypes were designed with the same task requirements and comparable information so that differences in the interaction model, rather than differences in the task itself, can be examined during evaluation.

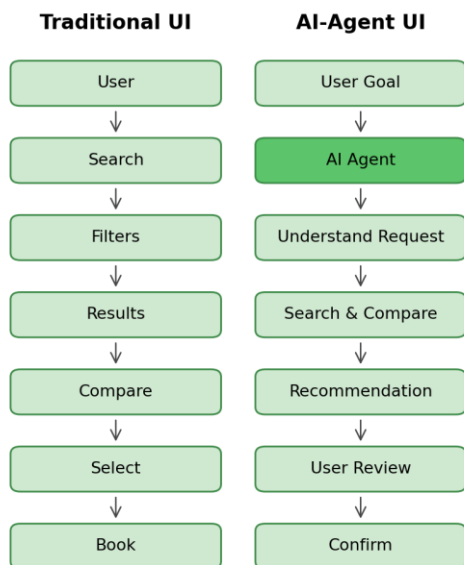


Fig. 1. Traditional and AI-agent interaction flows.

The two interaction models differ mainly in the distribution of actions between the user and the system. In the traditional interface, the user performs most of the search and comparison steps directly. In contrast, the AI-agent interface delegates several intermediate steps to the agent, while keeping the user involved in reviewing and approving important decisions.

The AI-agent interaction is organized into several functional stages. First, the system interprets the user's natural-language request and identifies the required constraints. The task is then divided into smaller operations, such as searching for relevant options, applying the specified criteria, and comparing the available results. Based on these operations, the agent generates recommendations and provides information that allows the user to evaluate the result.

User control is maintained at specific points of the process. The user can review the agent's interpretation, modify the requirements, interrupt the process, request alternatives, or approve the final action. This architecture is intended to reduce unnecessary manual interaction without removing the user's ability to supervise the agent's actions.

The overall structure of the proposed AI-agent system is presented in Fig. 2.



Fig. 2. AI-agent system architecture and user-control points.

V. EVALUATION

A. Evaluation Setup

The proposed interaction models were evaluated through a comparative user testing procedure. Participants were asked to complete the same hotel-search task using both the traditional and AI-agent prototypes. The task required finding a hotel in Kutaisi for two people for two nights, with a maximum price of 300 GEL per night, breakfast, and free parking.

The traditional prototype required participants to complete the search manually by entering the required information, applying filters, reviewing the available hotels, comparing alternatives, and selecting a hotel. In the AI-agent prototype, participants provided the same requirements as a natural-language request and then reviewed the agent's interpretation, recommendations, and final selection.

The evaluation was designed to compare the two interaction models under the same task conditions. Participants completed the task independently while the time required and the number of interactions were recorded. After completing each prototype, participants also provided ratings related to usability, perceived user control, and trust.

B. Evaluation Metrics

Six metrics were selected to evaluate the interaction models. Task completion time measures the time required to complete the assigned task, while the number of interactions indicates the amount of direct user input required during the process. Task success measures whether the participant completed the task according to the specified requirements.

Subjective evaluation focuses on perceived usability, perceived user control, and trust. Usability reflects how easy participants considered the interface to use. User control measures the extent to which participants felt able to control the interaction and its outcome. Trust reflects the participants' confidence in the information and recommendations provided by the interface.

TABLE I
EVALUATION METRICS

Metric	Description
Task completion time	Time required to complete the assigned task
Number of interactions	Number of direct user actions required
Task success	Whether the task was completed correctly
Perceived usability	How easy the interface was perceived to use
Perceived user control	How much control the participant felt over the process
Trust	Confidence in the interface and its recommendations

To compare the two interaction models in practice, five participants tested both prototypes using the same hotel-search task. The collected data included task completion time, number of interactions, task success, perceived usability, perceived user control, and trust. The results for each participant are presented in Table II.

TABLE II
PARTICIPANTS' RESULTS

Metric	UI	P1	P2	P3	P4	P5	Avg.
Time (min:s)	T	2:50	3:30	4:05	2:54	3:50	3:26
	AI	1:30	1:50	2:00	1:20	1:40	1:40
Interactions	T	14	16	14	13	17	14.8
	AI	5	6	8	4	6	5.8
Task success	T	Yes	Yes	Yes	Yes	Yes	100%
	AI	Yes	Yes	Yes	Yes	Yes	100%
Usability (1–5)	T	5	4	2	3	3	3.4
	AI	5	5	4	5	4	4.6
Control (1–5)	T	3	5	4	5	4	4.2
	AI	5	4	5	5	4	4.6
Trust (1–5)	T	4	4	4	4	5	4.2
	AI	4	5	5	4	4	4.4

T = traditional interface; AI = AI-agent interface.

C. User Testing Results

The results presented in Table II show a consistent difference in interaction efficiency between the two prototypes. All five participants successfully completed the assigned task using both interfaces. However, the AI-agent interface required less time and fewer direct interactions in every case.

The average task completion time was 3 minutes and 26 seconds for the traditional interface and 1 minute and 40 seconds for the AI-agent interface. Similarly, the average number of interactions decreased from 14.8 to 5.8. These results indicate that the AI-agent interaction model reduced the amount of manual input required to complete the task.

The subjective ratings also differed between the two prototypes. The AI-agent interface received a higher average usability rating (4.6/5) compared with the traditional interface (3.4/5). Perceived user control was rated 4.6/5 for the AI-agent interface and 4.2/5 for the traditional interface, while trust received average ratings of 4.4/5 and 4.2/5, respectively.

Although the AI-agent interface showed lower interaction effort and higher subjective ratings in this test, the results should be interpreted within the context of the five-participant sample and the prototype-based evaluation. The findings therefore provide an initial comparison of the two interaction models rather than a general conclusion about AI-agent interfaces.

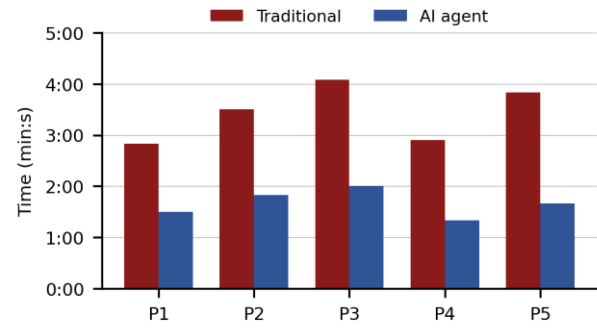


Fig. 3. Task completion time per participant.

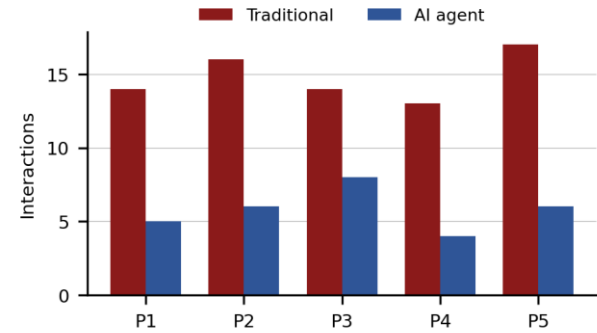


Fig. 4. Number of interactions per participant.

To further illustrate the differences between the two interaction models, Figs. 3 and 4 present the individual results obtained from the five participants. Each participant completed the same task using both the traditional interface and the AI-agent interface, allowing a direct comparison of their performance. Fig. 3 shows the task completion time for each participant, while Fig. 4 presents the corresponding number of interactions. Across the participants, the AI-agent interface generally required less time and fewer interactions than the traditional interface. These individual results are consistent with the overall averages presented in Table II and provide further evidence of the difference in interaction efficiency between the two prototypes.

VI. DISCUSSION

The results of the user testing indicate that AI-agent interfaces can significantly change the way users interact with digital systems. In the evaluated hotel-booking scenario, the AI-agent interface required fewer interactions and less time to complete the same task while maintaining the same task success rate as the traditional interface. This suggests that an agent-based approach can reduce the number of manual steps required from users and make task completion more efficient.

One of the most important findings concerns the relationship between efficiency and user control. Although the AI-agent performed several actions on behalf of the user, participants did not report a significant reduction in their perceived control. The relatively high control score may suggest that participants were comfortable with a more automated interaction model when the system provided sufficient visibility into its actions and allowed them to understand or confirm important decisions [1], [2].

The usability results also suggest that users may perceive AI-agent interfaces as more convenient than conventional interfaces for tasks involving multiple sequential actions. In a traditional interface, users are required to navigate through different screens, select options, enter information, and confirm individual steps. An AI agent can combine these actions into a more natural interaction in which the user communicates an intention and the system performs the necessary operations. This can reduce interaction complexity, particularly for users who are unfamiliar with the structure of a particular application.

However, increased automation also introduces new design challenges. When an AI agent performs actions automatically, users may have less direct visibility into the underlying process. This creates a potential risk of confusion if the system makes an incorrect assumption or performs an unexpected action. For this reason, agent-based interfaces should not focus exclusively on minimizing user interactions. They should also provide clear feedback, explain important decisions, and give users appropriate opportunities to review, modify, or cancel actions [1], [3].

The trust results further support the importance of transparency in AI-driven interfaces [3]. Participants reported a high level of trust in both interface types, with a slightly higher value for the AI-agent interface. However, the small difference suggests that efficiency alone is not sufficient to establish trust. Users need to understand what the system is doing and retain the ability to intervene when necessary. Consequently, transparency and controllability should be considered fundamental interface requirements rather than optional features.

Overall, the findings support the idea that the future of user interfaces may involve a shift from interfaces centered primarily on direct manipulation toward interfaces that combine traditional visual interaction with intelligent agents. Such systems can automate routine actions while preserving conventional interface elements for situations where users need detailed control. The objective should therefore not be to replace graphical interfaces completely, but to create a balanced interaction model in which automation, usability, transparency, and user control complement each other.

VII. CONCLUSION

This study examined how AI-agent interfaces can influence user interaction with digital systems by comparing an AI-agent prototype with a traditional graphical interface in a hotel-booking scenario. The user testing results showed that the AI-agent interface enabled participants to complete the same task with less time and fewer direct interactions while maintaining the same task success rate. The higher usability rating also suggests that participants perceived the agent-based interaction as more convenient for a task involving multiple sequential actions.

At the same time, the results indicate that increased automation does not necessarily require the removal of user control. Participants reported high levels of perceived control and trust when interacting with the AI-agent interface. However, these findings also highlight the importance of

transparency, clear feedback, and opportunities for user intervention when agents perform actions on behalf of users.

The study demonstrates that AI agents can complement traditional graphical interfaces by reducing repetitive manual actions while preserving visual interaction where detailed control is required. Rather than completely replacing conventional interfaces, AI-agent systems can support a more flexible interaction model in which users communicate their intentions and the system assists with the execution of routine tasks.

The findings should be considered within the limitations of the study, particularly the small sample of five participants and the use of prototype-based evaluation. Future research should evaluate AI-agent interfaces with larger and more diverse groups of users, more complex real-world tasks, and longer-term interaction. Further investigation is also needed into transparency, user control, trust, and error recovery in agent-based interfaces. Such research can contribute to the development of user interfaces that combine automation and efficiency with clear communication, user control, and reliable interaction.

REFERENCES

- [1] S. Amershi, D. Weld, M. Vorvoreanu, A. Fournay, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz, "Guidelines for Human-AI Interaction," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow, Scotland, UK, 2019, pp. 1–13, doi: 10.1145/3290605.3300233.
- [2] M. Raees, I. Meijerink, I. Lykourantzou, V.-J. Khan, and K. Papangelis, "From explainable to interactive AI: A literature review on current trends in human-AI interaction," *International Journal of Human-Computer Studies*, vol. 189, Art. no. 103301, 2024, doi: 10.1016/j.ijhcs.2024.103301.
- [3] M. Vössing, N. Köhl, M. Lind, and G. Satzger, "Designing Transparency for Effective Human-AI Collaboration," *Information Systems Frontiers*, vol. 24, pp. 877–895, 2022, doi: 10.1007/s10796-022-10284-3.
- [4] D. Nguyen, J. Chen, Y. Wang, G. Wu, N. Park, Z. Hu, H. Lyu, J. Wu, R. Aponte, Y. Xia, X. Li, J. Shi, H. Chen, V. D. Lai, Z. Xie, S. Kim, R. Zhang, T. Yu, M. Tanjim, N. K. Ahmed, P. Mathur, S. Yoon, L. Yao, B. Kveton, T. H. Nguyen, T. Bui, T. Zhou, R. A. Rossi, and F. Dernoncourt, "GUI Agents: A Survey," *arXiv preprint arXiv:2412.13501*, 2024.

Generating and Evaluating the Initial Architecture of Complex Software Projects Using a Multi-Agent AI System

Luka Natchkebia

Georgian Technical University, Tbilisi, Georgia

Abstract—In the modern applied software industry, the rapid and error-free design of complex digital systems is vital. Conceptual errors during the initial architecture phase lead to an exponential increase in technical debt, complicating future scaling and maintenance. While Large Language Models (LLMs) are widely used for code generation, general-purpose models struggle with multi-component architecture, often producing incompatible or outdated templates.

This paper proposes a multi-agent artificial intelligence approach to address this limitation. Unlike traditional methods, our proposed system will utilize independent agents trained via Parameter-Efficient Fine-Tuning (PEFT/LoRA). The planned architecture features parallel Backend, Web, and Mobile architecture agents. An "Actor-Critic" paradigm will act as an independent evaluator, conducting structural audits and API contract verifications through a feedback loop.

The primary expected outcome is an applied software package that automates and evaluates initial architecture generation. Once tested empirically on real engineering tasks, the system aims to serve as a practical tool for the Niko Muskhelishvili Institute of Computational Mathematics and as an educational platform for students.

Index Terms—Applied software package, Artificial intelligence, Fine-tuning, Large language models, Multi-agent AI.

I. INTRODUCTION

Efficient and error-free initial architectural design (often referred to as Sprint 0) is a critical foundation for modern, multi-layered software ecosystems. In the contemporary software industry, conceptual mistakes made during this phase inevitably lead to an exponential increase in technical debt. This debt severely hinders a system's future scalability, maintainability, and cross-platform synchronization. Consequently, there is a growing demand for the automation of the architectural planning process.

While the integration of Large Language Models (LLMs) has revolutionized localized code generation, general-purpose models exhibit significant limitations when tasked with macro-level architectural planning. Universal models often lack deep, synchronized knowledge of narrow, platform-specific domains. As a result, they tend to generate structurally incompatible or deprecated templates - such as recommending legacy XML layouts instead of modern declarative UI frameworks.

To overcome these barriers, this paper introduces a proposed,

specialized multi-agent AI system designed to generate and evaluate the initial architecture of complex software projects. By shifting from a monolithic LLM to a coordinated network of domain-specific agents, we aim to achieve higher accuracy and structural integrity. The resulting applied software package is intended to serve both as an advanced research tool at the Niko Muskhelishvili Institute of Computational Mathematics and as a practical educational instrument for students to master the latest engineering standards.

II. RELATED WORK

The application of artificial intelligence in software engineering has been extensively explored, particularly with the advent of generative LLMs. Standard tools currently assist developers in micro-level tasks, yet they frequently suffer from context loss and "architectural hallucinations" when managing complex, multi-repository projects.

Recent research has pivoted toward Multi-Agent Systems (MAS) to solve complex coding tasks through role-playing frameworks. In these systems, AI agents assume roles such as "Programmer" or "Reviewer". However, most existing MAS frameworks still rely on underlying general-purpose models via prompt engineering rather than utilizing specialized, fine-tuned weights for specific domains.

Our planned work bridges this gap by applying Parameter-Efficient Fine-Tuning (PEFT), specifically Low-Rank Adaptation (LoRA), to train individual agents. By combining LoRA-adapted experts with an objective "Actor-Critic" validation mechanism, this project aims to introduce a highly robust pipeline for synchronized, cross-platform architecture generation that surpasses the capabilities of generic multi-agent frameworks.

III. PROPOSED METHOD

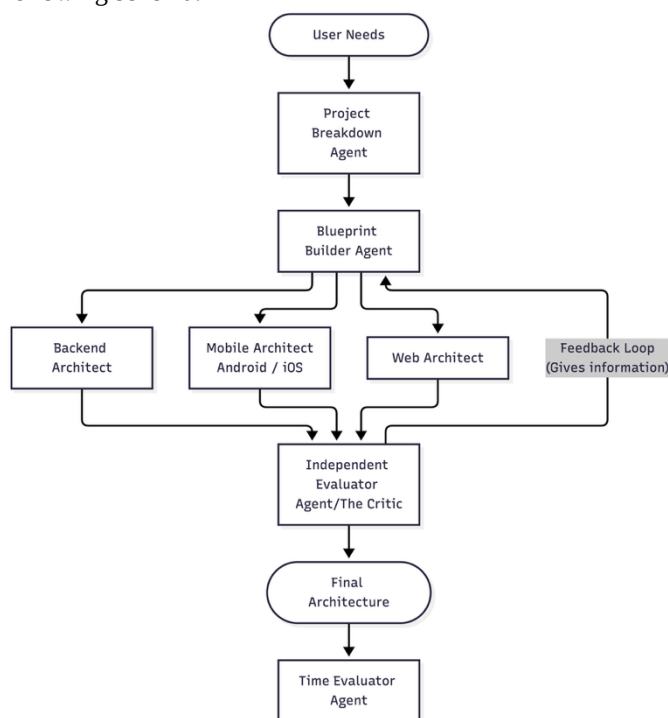
The core strategy of our approach is the integration of a multi-agent artificial intelligence ecosystem. Instead of relying on a single, generalist model to design an entire software project, the workload will be distributed among independent, highly specialized agents. The system will employ a "divide and conquer" methodology. The initial unstructured requirements from the user will be logically decomposed by a designated

planning agent. These requirements will then be translated into structured specifications and routed in parallel to domain-specific agents: the Backend Architect, the Mobile Architect (Android/iOS), and the Web Architect. Each of these agents will be fine-tuned (via LoRA) exclusively on high-quality, modern, and verified open-source repositories and official documentation.

To guarantee cross-platform synchronization, the method incorporates an "Actor-Critic" paradigm. An Independent Evaluator Agent will audit the outputs of the architect agents. If the generated architectures exhibit discrepancies - for instance, if the Backend API contracts do not match the Mobile client's expected data models - the Evaluator will trigger a feedback loop. This iterative refinement will continue until complete structural harmony is achieved.

IV. SYSTEM ARCHITECTURE

The applied software package operates through a structured pipeline orchestrating multiple specialized nodes. The architecture is designed to run efficiently on local or cloud infrastructure (utilizing frameworks such as llama.cpp for optimized execution). The workflow proceeds through the following schema.



1. **Project Breakdown Agent:** Analyzes the user's raw input (e.g., project descriptions, verbal agreements) and defines the overall technical scope.
2. **Blueprint Builder Agent:** Decomposes the scope logically and distributes specific technical requirements across different platforms.
3. **Parallel Architect Agents:**

- a. **Backend Architect:** Generates microservice structures, database schemas, and OpenAPI/Swagger specifications.
- b. **Mobile Architect:** Generates native mobile architectures enforcing modern standards (e.g., Clean Architecture, Kotlin, Jetpack Compose).
- c. **Web Architect:** Generates the frontend component hierarchy and state management models.

4. **Independent Evaluator Agent (The Critic):** Ingests the structured files generated by the architects. It performs a rigorous structural audit to verify API contract compatibility and adherence to defined constraints. It utilizes a Feedback Loop to return errors to specific architects for correction.
5. **Time Evaluator Agent:** Analyzes the finalized, synchronized architecture and outputs a recommended timeline and resource estimation for the development teams.

V. EVALUATION

The evaluation of the proposed applied software package will be conducted empirically using real-world engineering tasks. To validate the system's efficacy and clearly demonstrate its advantages, we plan to utilize retrospective analysis of previously completed or ongoing software projects.

The testing protocol will involve feeding the original requirements of an existing project into the multi-agent system. The system will then generate the initial architecture and provide its own engineering recommendations. This AI-generated output will subsequently be compared against the actual human-designed architecture implemented by professional development teams.

Key evaluation metrics will include Structural Compatibility, Standard Adherence, and Efficiency Gains. Namely, ensuring 100% alignment between generated backend API contracts and frontend/mobile endpoints, the frequency at which the models correctly utilize modern frameworks over deprecated ones, and the reduction in time required to complete the "Sprint 0" phase compared to manual human effort. We anticipate that the multi-agent orchestration, combined with the Actor-Critic feedback loop, will significantly reduce structural mismatches and mitigate the accumulation of early-stage technical debt.

VI. CONCLUSION

This paper outlined a proposal for an applied software package that will leverage a multi-agent AI system to automate and evaluate the generation of initial software architectures. By moving away from general-purpose LLMs and planning to utilize PEFT/LoRA-trained domain experts, the system aims to provide a highly synchronized and reliable foundation for complex digital projects. The integration of the Actor-Critic paradigm will ensure that generated components are perfectly

aligned across Backend, Web, and Mobile platforms.

The practical impact of this system is expected to be substantial. It will serve as a foundational tool at the Niko Muskhelishvili Institute of Computational Mathematics to support ongoing AI algorithmic research. Furthermore, the system is envisioned as a highly effective educational and practical instrument, enabling academic staff and students to seamlessly adopt and interact with the latest software engineering standards. Future work will focus on expanding the dataset for agent fine-tuning and integrating the system into broader, real-time development environments.

REFERENCES

- [1] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, pp. 18–21, 2012.
- [2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations (ICLR)*, 2021.
- [3] S. Hong, S. Zheng, J. Chen, Y. Cheng, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, and C. Wu, "MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework," *arXiv preprint arXiv:2308.00352*, 2023.
- [4] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," in *Advances in Neural Information Processing Systems*, 2023.

NeuraScan: A Multimodal Platform for Real-Time Fatigue Monitoring Using Physiological Sensing, Eye Tracking, and AI-Based Fusion

Nikoloz Neparidze
Georgian Technical University, Tbilisi, Georgia

Abstract— Fatigue is a major contributor to workplace accidents, road incidents, and degraded clinical decision-making, yet most monitoring tools rely on a single signal and miss the broader physiological picture. This paper presents NeuraScan, a multimodal fatigue monitoring platform that fuses three layers: wearable physiological sensing (ECG, galvanic skin response, and infrared body temperature via an ESP32 microcontroller), webcam-based eye tracking (blink rate and PERCLOS), and an AI diagnostic engine that synthesizes these signals with NASA-TLX subjective workload scores into a unified fatigue score. Unlike single-modality approaches, NeuraScan aims to capture both objective physiological strain and subjective perceived workload in real time. We describe the system architecture, present the current implementation status, including firmware, sensor integration, and a working demonstration interface, and outline the evaluation protocol planned once sensor hardware is fully deployed.

Index Terms—Eye tracking, fatigue detection, multimodal sensing, physiological monitoring, wearable systems.

I. INTRODUCTION

Fatigue — whether from sleep deprivation, sustained cognitive load, or physical exertion — degrades reaction time, attention, and decision-making, and is a documented factor in transportation accidents, industrial incidents, and clinical errors. Existing fatigue-detection tools typically rely on a single channel: heart-rate variability from a wearable, blink-based drowsiness detection from a dashboard camera, or self-reported workload questionnaires. Each of these signals is informative in isolation but incomplete: physiological sensors can be confounded by motion artifacts, visual cues alone miss internal physiological strain, and subjective questionnaires cannot be collected continuously. This paper introduces NeuraScan, a three-layer platform that combines wearable physiological sensing, webcam-based eye tracking, and an AI fusion engine that also incorporates NASA Task Load Index (NASA-TLX) responses, in order to produce a single, continuously updated fatigue score. The remainder of this paper is organized as follows: Section II reviews related work in physiological and visual fatigue monitoring; Section III presents the proposed

method; Section IV details the system architecture; Section V outlines the evaluation plan and current implementation status; and Section VI concludes with a summary and future work.

II. RELATED WORK

Physiological approaches to fatigue and stress detection have long used electrocardiogram (ECG) and galvanic skin response (GSR) signals, since heart-rate variability and skin conductance shift measurably under fatigue and stress [1]. Vision-based approaches instead track ocular behavior: PERCLOS, the percentage of eyelid closure over the pupil over time, remains one of the most validated proxies for drowsiness in driving-simulator studies [2]. Subjective workload assessment, most notably the NASA Task Load Index (NASA-TLX), complements these objective measures by capturing perceived mental, physical, and temporal demand directly from the user [3]. Low-cost microcontroller platforms such as the ESP32, paired with sensors like the AD8232 ECG front-end, have made continuous physiological logging feasible outside clinical settings [4]. However, most existing systems use only one of these modalities at a time; few combine wearable physiological sensing, computer-vision-based eye tracking, and subjective workload scoring into a single fused estimate, which is the gap NeuraScan targets.

III. PROPOSED METHOD

NeuraScan is organized as three cooperating layers rather than a single sensor pipeline. The first layer captures objective physiological signals directly from the body: an ECG waveform, galvanic skin response, and peripheral body temperature. The second layer captures ocular behavior through a standard webcam, without requiring additional hardware, by tracking blink rate and PERCLOS. The third layer is an AI diagnostic engine that does not simply threshold individual signals; instead, it synthesizes the physiological time series, the eye-tracking metrics, and periodic NASA-TLX self-report scores into a single, continuously updated fatigue estimate. The rationale for this design is that physiological strain, visible

drowsiness cues, and perceived workload do not always move together — a user may report high workload before physiological markers shift, or vice versa — so fusing all three narrows the blind spots of any one modality alone.

IV. SYSTEM ARCHITECTURE

Layer 1 (Physiological Sensing) is built on an ESP32 microcontroller, chosen for its built-in Wi-Fi/Bluetooth connectivity and sufficient analog input channels. It interfaces with an AD8232 single-lead ECG front-end (via a three-electrode cable set) for cardiac signal acquisition, a GSR sensor for skin-conductance readings, and an MLX90614 non-contact infrared sensor for peripheral body temperature. Custom Arduino firmware samples all three channels, timestamps them, and streams the data over a serial/Wi-Fi link. Layer 2 (Eye Tracking) runs entirely on a standard webcam feed: a computer-vision pipeline detects the eye region per frame, computes blink rate, and derives PERCLOS as a rolling percentage of frames in which the eyes are more than 80% closed. Layer 3 (AI Fusion Engine) ingests the synchronized outputs of Layers 1 and 2 together with periodic NASA-TLX questionnaire responses, and uses an AI model to combine them into a single fatigue score together with a plain-language rationale, rather than a single opaque number. The general data flow is summarized below:

```
ECG, GSR, temperature --> ESP32 firmware --> timestamped stream
Webcam frames --> eye-region detection --> blink rate, PERCLOS
NASA-TLX responses --> periodic subjective workload score
All three streams --> AI fusion engine
Fusion engine --> unified fatigue score + rationale
```

V. EVALUATION

At the time of writing, sensor hardware for Layer 1 is on order, so this section describes both what has already been validated and the protocol planned for full-system evaluation. The software stack has been validated independently of the physiological hardware: a browser-based demonstration interface (Fig. 1) simulates the complete data pipeline, including a real-time EEG-style waveform display and a fatigue-score visualization, to confirm that the fusion engine's inputs, outputs, and interface update correctly under continuous data. Once the ESP32-based sensor unit is assembled, the planned evaluation protocol is a within-subject study in which volunteers complete a sustained attention task while ECG, GSR, and temperature are logged continuously, webcam-based PERCLOS and blink rate are recorded in parallel, and NASA-TLX is administered at fixed intervals. The fused fatigue score will be compared against NASA-TLX scores and against PERCLOS alone as a single-modality baseline, using correlation and classification accuracy (fatigued vs. non-fatigued, as labeled by task performance decline) as the primary metrics. This design is intended to show whether fusing modalities improves detection accuracy and lead time over any single signal.

VI. CONCLUSION

This paper presented NeuraScan, a three-layer fatigue monitoring platform that fuses wearable ECG, GSR, and temperature sensing with webcam-based eye tracking and NASA-TLX subjective workload scoring through an AI diagnostic engine. Unlike single-modality fatigue detectors, NeuraScan is designed to cross-check objective physiological strain against visible drowsiness cues and self-reported workload, narrowing the blind spots inherent to any one signal. Firmware, wiring, and the demonstration interface are complete; sensor hardware delivery and the planned within-subject evaluation are the immediate next steps, alongside an upcoming colloquium presentation of the project. Future work includes validating the fusion engine's outputs against the evaluation protocol described above, tuning the model on real physiological data rather than simulated inputs, and exploring extensions such as long-term fatigue trend tracking and deployment in workplace or driving contexts.

REFERENCES

- [1] J. A. Healey and R. W. Picard, "Detecting stress during real-world driving tasks using physiological sensors," *IEEE Trans. Intelligent Transportation Systems*, vol. 6, no. 2, pp. 156–166, 2005.
- [2] W. W. Wierwille and L. A. Ellsworth, "Evaluation of driver drowsiness by trained raters," *Accident Analysis & Prevention*, vol. 26, no. 5, pp. 571–581, 1994.
- [3] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research," in *Human Mental Workload*, *Advances in Psychology*, vol. 52, pp. 139–183, 1988.
- [4] Analog Devices, "AD8232: Single-Lead, Heart Rate Monitor Front End," Datasheet, Rev. B, 2019.



Fig. 1. NeuraScan demonstration interface (simulated data), showing the live eye-tracking overlay with blink rate and PERCLOS, simulated ECG/GSR/temperature sensor readouts, the fused fatigue index ring, and the NASA-TLX questionnaire panel.

Modeling the Federal IT Architecture Behind Germany's Online Access Act (OZG)

Tilman Petzold

Abstract – Germany's Online Access Act (Onlinezugangsgesetz, OZG) requires public authorities to offer their services online, but progress has been slow. This paper looks at the problem from an IT architecture perspective. Building on a project from an earlier module on German public administration, it models the federal E-Government landscape as a three-tier architecture (frontend, middleware, backend) and shows where the data flow between citizens and municipal backends breaks down. The analysis finds that most delays come from missing standardized APIs, inconsistent data formats, and a lack of a strong middleware layer between the roughly 11,000 municipalities and the federal portals.

Index Terms – e-government, enterprise architecture, OZG, public administration, FIT-Connect

I. INTRODUCTION

The Online Access Act (Onlinezugangsgesetz, OZG) obliges German federal, state, and local authorities to offer their services digitally. Despite this legal mandate and the follow-up OZG 2.0, implementation is years behind schedule. Public debate usually blames bureaucracy, but the underlying problem is structural: Germany's IT landscape is split across the federal government, 16 states, and roughly 11,000 municipalities, each historically running its own IT systems and, in many cases, its own local variant of a given administrative process [1].

In a previous project, the author simplified and analyzed the structure and tasks of administrative bodies. Building on this foundation, this paper aims to extend these findings with a technical focus. It asks: where exactly in the data flow from citizen to municipal backend do technical bottlenecks occur, and what would resolve them?

The scale of the delay is visible in recent industry figures. Of the 579 administrative services the OZG covers, only 165 are fully available online nationwide; 230 are not digitally available at all [8] (Fig. 1). Understanding this gap between the legal deadline and technical reality motivates this paper.

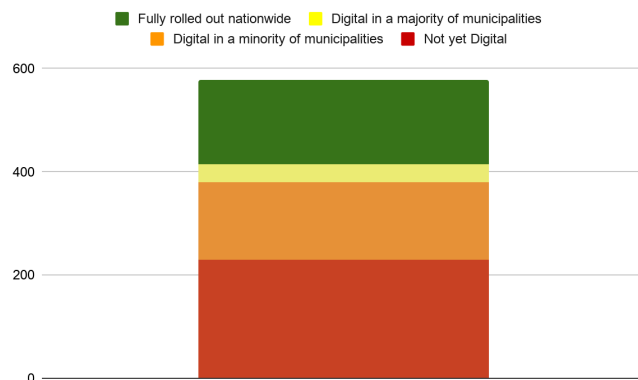


Fig. 1. Rollout status of the 579 OZG services as of September 2025. Data from Bitkom [8].

II. RELATED WORK

To map the data flow from the user to the software backend, this study utilizes Enterprise Architecture Management (EAM). EAM allows organizing an institution's IT landscape into a coherent set of models rather than describing each system in isolation. When applied to the public sector, it is often referred to as Government Enterprise Architecture (GEA). Early German standardization efforts introduced XÖV, a family of shared XML data schemas that public registries use to exchange records, such as civil-status or resident data. However, these standards mostly relied on rigid point-to-point connections between two specific systems, which do not scale to thousands of municipalities. More recent middleware platforms, such as FIT-Connect, a routing service operated by the Federal IT Cooperation (Föderale IT-Kooperation, FITKO) on behalf of the IT Planning Council (IT-Planungsrat), aim to replace this with flexible, API-based routing between any connected frontend and backend. This paper builds on that shift by mapping how such middleware fits into the wider federal architecture.

III. PROPOSED METHOD

The analysis follows three steps: (1) collecting architectural guidelines from the Federal Ministry of the Interior (BMI) and the IT Planning Council, the body coordinating IT cooperation between the federal government and the states [1], [2]; (2) organizing the German E-Government landscape into a layered Enterprise Architecture model, i.e. grouping systems by the role they play (user-facing, transport, or processing) rather than by which authority owns them; and (3) conceptually tracing a single online application from the citizen portal to the municipal backend to identify where the data flow breaks.

IV. SYSTEM ARCHITECTURE

Fig. 2 shows the resulting three-tier model:

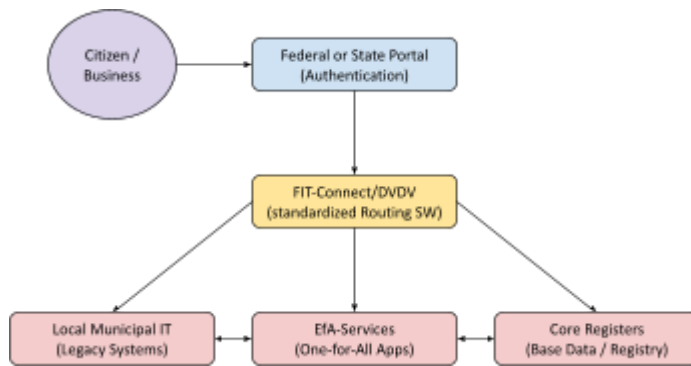


Fig. 2. Three-tier model of the German federal E-Government architecture. Own illustration.

The frontend layer covers the federal citizen portal (Bundesportal) and the individual state portals (Landesportale), with identity and login handled by the shared BundID service so that citizens do not need a separate account per authority. The middleware layer routes data between frontend and backend; FIT-Connect is the main example, decoupling services from municipal systems via standardized, encrypted transfers rather than fixed one-to-one connections. The backend layer is the most fragmented part: thousands of local legacy systems, which are the specialized software each municipality uses to actually process a given type of case, e.g. a housing-benefit or trade-license application. Also there are newer shared EfA services and separate master-data registries such as civil registries.

V. EVALUATION

Applying the process trace from Section III to the architecture in Fig. 2 surfaces three recurring bottlenecks, illustrated in Fig. 3:

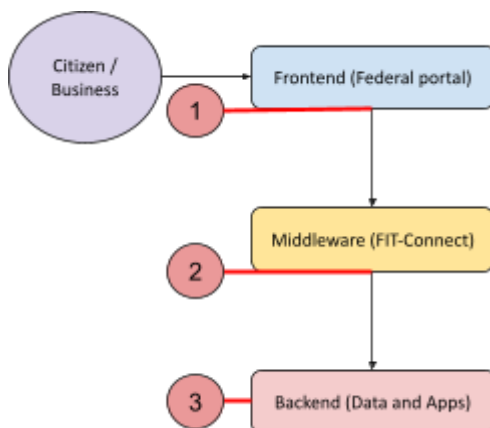


Fig. 3. Points along the citizen-to-backend data flow where processing breaks down. Own illustration.

(1) Semantic gaps: municipalities extend the shared XÖV data schemas differently. For example, a field that one state's specialized process requires may be optional or coded differently in another, so backends often cannot parse incoming data automatically even though both sides technically use "the same" standard [4], [7].

(2) Media discontinuities in practice: even where EfA services deliver structured data, receiving applications often cannot import it directly, because mandatory fields and validation rules differ between the state that built the service and the state reusing it [7]. As a result, staff frequently still process submissions from the digital inbox (e.g. EGVP, the judicial and administrative electronic mailbox system) by hand into the required forms, so a digital frontend alone does not remove analog backend work [6]. Industry practice addresses this through "Dunkelverarbeitung" (straight-through processing, i.e. automated processing with no human step in between): an integration layer between portals and processes automatically validates, routes, and files routine cases, escalating only those needing legal judgment to a caseworker [6].

(3) Access management overhead: connecting thousands of municipalities individually still requires heavy manual configuration, because each new connection has traditionally needed its own point-to-point setup instead of a shared, standardized entry point (a "webhook").

FIT-Connect already reduces bottleneck (3) by offering a shared, JSON-based REST interface (a common, lightweight format for exchanging structured data over the web) instead of a custom connection per municipality, lowering the effort for local software vendors [3]. Bottlenecks (1) and (2), however, persist even where FIT-Connect is already in use, because they concern the data format and the receiving system, not the transport layer.

VI. CONCLUSION

The statistics in Fig. 1 and the bottlenecks in Fig. 3 point to the same conclusion: digitalizing public administration is not primarily a legal problem but a technical one. Without standardized, API-first interfaces at every layer, legal mandates like the OZG cannot close the gap between digital frontends and analog backends. Practitioners suggest a concrete direction: an integration layer that automatically maps, validates, and routes structured submissions into municipal legacy systems, through "Dunkelverarbeitung", so staff handle only cases that require judgment [6]. Future work should turn the qualitative process trace used here into a formal BPMN model and examine how a shared federal cloud infrastructure and the Once-Only principle could remove the remaining media discontinuities.

REFERENCES

- [1] Bundesministerium des Innern und für Heimat (BMI) / IT-Planungsrat, "Zielbild der OZG-Rahmenarchitektur – Konsultationsprozess, 2. Phase," AL-Runde, 3 September 2024. [Online]. Available: https://it-planungsrat.de/fileadmin/beschluesse/2024/Information2024_1_1_OZG-Rahmenarchitektur.pdf.
- [2] IT-Planungsrat, "API First – Strategischer Schwerpunkt Digitale Anwendungen: Status, Rahmenbedingungen und Lessons Learned aus der OZG-Umsetzung," Beschluss 2024-55, 2024. [Online]. Available: https://it-planungsrat.de/fileadmin/beschluesse/2024/Beschluss_2024-55_API_First_Pr%C3%A4sentation.pdf.
- [3] IT-Planungsrat / Föderale IT-Kooperation (FITKO), "Technischer Abschlussbericht FIT-Connect," Beschluss 2022-32, Stand 11.05.2022. [Online]. Available: https://www.it-planungsrat.de/fileadmin/beschluesse/2022/Beschluss2022-32_FIT-Connect_AL2_Technischer-Abschlussbericht.pdf.
- [4] Koordinierungsstelle für IT-Standards (KoSIT), "XÖV-Standardisierungsrahmen für den elektronischen Datenaustausch in der öffentlichen Verwaltung (XÖV-Handbuch). [Online]. Available: <https://www.xoev.de/>.
- [5] F. Burmeister, "Enterprise and Ecosystem Architecture Modeling for the Creation and Application of Law," Dissertation, Universität Hamburg, 2025.
- [6] TRANSCONNECT / SQL Projekt AG, "OZG 2.0 & Dunkelverarbeitung: So lösen Kommunen den Backend-Stau," company blog (practitioner source), 14 Dec. 2025. [Online]. Available: <https://transconnect.io/blog/ozg-2-0-backend-dunkelverarbeitung>.
- [7] OZG-Taskforce, "Stellungnahme zum aktuellen Stand der Digitalisierung," 2024. [Online]. Available: <https://ozg-taskforce.de/stellungnahme-zum-aktuellen-stand-der-digitalisierung/>.
- [8] Bitkom e.V., "Bitkom-Analyse: 60 Prozent der Verwaltungsleistungen sind online verfügbar," Presseinformation, 23 Sept. 2025. [Online]. Available: <https://www.bitkom.org/Presse/Presseinformation/Bitkom-Analyse-60-Prozent-der-Verwaltungsleistungen-sind-online-verfuegbar>.

Backpropagating Through Motor Circuits: A Differentiable Motor-Neuron Path in MyoGen

Salome Pirozmanishvili

Georgian Technical University, Tbilisi, Georgia

Abstract—Calibrating biophysical motor-neuron (MN) models to electrophysiological recordings is difficult when parameter gradients are unavailable. We integrate Jaxley, a differentiable JAX-based neural simulator, into MyoGen and evaluate whether it can serve as a differentiable alternative to the existing NEURON MN backend under matched stochastic descending drive. Across ten paired seeds, both implementations preserved orderly recruitment and closely aligned discharge-rate behavior. Jaxley recruited more MNs and produced more spikes (both $p < 0.001$), while mean discharge rate per active MN remained similar in scale ($p = 0.016$). In the reported paired analysis, 1,937 spikes coincided within ± 5 ms and more than 80% matched at ± 50 ms. After 50 ms binning and 300 ms smoothing, population discharge-rate trajectories were highly correlated ($r = 0.996$). Jaxley intrinsic properties also overlapped reported physiological ranges for cat medial gastrocnemius MNs. Together, these results show that the MyoGen MN stage can be transferred to a differentiable simulator while preserving the principal recruitment and population-rate behavior of the established implementation. The remaining recruitment and spike-timing offsets are expected for distinct simulator implementations and provide clear targets for further numerical alignment.

Index Terms—Differentiable simulation, electromyography, motor neurons, neuromuscular modeling.

I. INTRODUCTION

Motor-unit (MU) discharge patterns reflect the transformation of descending and spinal inputs into muscle activation. Simulating this transformation can help interpret electrophysiological recordings and test mechanistic hypotheses. MyoGen connects neural activity with downstream force and EMG generation [1]. Its existing biophysical motor-neuron (MN) path uses NEURON [2], a mature simulator, but direct gradient-based calibration of the full MN population is not native to that path. Previous work has used large-scale MN simulation to infer synaptic inputs from population firing [5].

Jaxley implements biophysical neural models in JAX and supports differentiation and accelerator execution [3]. We integrate a Jaxley MN population path with MyoGen and ask whether the existing MN stage can be moved to a differentiable backend while preserving its principal physiological behavior under matched descending drive. We therefore compare recruitment, discharge-rate dynamics, spike timing, and selected intrinsic properties with the established NEURON implementation and with physiological ranges reported by Zengel et al. [4]. The aim is not spike-for-spike identity

between two different simulation systems, but preservation of the MN population behavior required for MyoGen while gaining direct access to parameter gradients.

II. BACKGROUND AND SYSTEM DESIGN

An MN pool converts descending drive into MU spike trains. MyoGen uses these events to generate muscle force through twitch summation and electrophysiological signals through MU contributions [1]. The Jaxley path supplies Hodgkin–Huxley-type MN dynamics in JAX. Model parameters such as channel conductances and geometry can affect membrane voltage and spike generation. Automatic differentiation through the neural simulation provides a route to fitting intrinsic dynamics against suitable targets [3].

The broader MyoGen pipeline carries simulated MU activity onward to force, surface EMG, and intramuscular EMG. Replacing the MN stage with Jaxley therefore adds a differentiable component without changing the downstream MyoGen architecture. This creates a direct route toward objectives based on membrane voltage or MU dynamics and, with differentiable downstream observation models, toward future losses defined on force or EMG. The present work establishes this differentiable MN path and evaluates whether its output remains consistent with the established backend.

Model Equations

For compartment i , a generic current-balance description of a multicompartment MN model is [2, 3]

$$C_i \frac{dV_i}{dt} = - \sum_k I_{k,i}(V_i, x_i) + I_{syn,i} + \sum_{j \in N(i)} g_{ij}(V_j - V_i). \quad (1)$$

Here C_i is the *total* compartment capacitance; $I_{k,i}$ are outward-positive ionic currents, $I_{syn,i}$ is inward-positive synaptic current, and g_{ij} is an axial conductance. This convention keeps every term in current units. The ionic-current term represents the Hodgkin–Huxley-type channel dynamics implemented in the MN model [6]. In Jaxley, differentiable parameters such as channel conductances and geometry enter this dynamical system directly. The equations above define the shared current-balance structure. Because Jaxley and NEURON are distinct simulation systems, differences in solver behavior, discretization, event handling, and implementation details can lead to small changes in threshold crossing and spike timing even when the underlying physiological model is closely

¹Georgian Technical University, Tbilisi, Georgia. ²Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany.

matched.

For spike times $t_{i,n}$ of MU i , the downstream force and EMG stages can be represented schematically by superposition [7, 1]:

$$\begin{aligned} F(t) &= \sum_i \sum_n f_i(t - t_{i,n}), \\ e(t) &= \sum_i \sum_n u_i(t - t_{i,n}) + \varepsilon(t). \end{aligned} \quad (2)$$

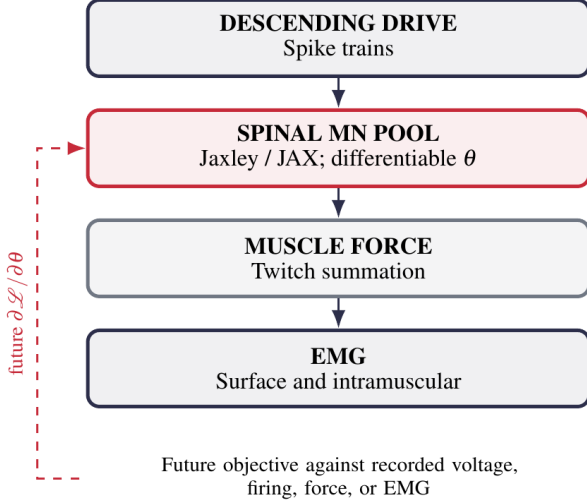


Fig. 1. Neuromuscular simulation pipeline and scope of the present work. Descending-drive spikes enter the Jaxley MN pool, whose output feeds the existing MyoGen force and EMG stages. Differentiation is demonstrated for MN parameters θ ; the dashed red arrow denotes the future goal of propagating losses from downstream observations back to those parameters. The present study establishes the differentiable MN stage, with downstream gradient propagation as the next extension.

Here f_i is a unit twitch, u_i is the MU action-potential waveform for a specified electrode configuration, and ε is optional measurement noise. These expressions describe a linear-summation approximation; they are not a claim that force or EMG was validated in the present comparison. Motivated by voltage fitting in Jaxley [3], one possible future *voltage-target* objective with N_t sampled time points is

$$L(\theta) = \frac{1}{N_t} \sum_{q=1}^{N_t} [V(t_q; \theta) - V^*(t_q)]^2. \quad (3)$$

Its gradient with respect to differentiable neuron parameters θ can be used for optimization. The present study establishes the differentiable MN stage; parameter fitting and extension of gradients through spike extraction and downstream EMG synthesis form the next stage of the framework.

III. COMPARISON PROTOCOL

We compared Jaxley with the existing NEURON-based MyoGen backend using matched stochastic descending-drive populations and a trapezoidal contraction envelope. For each paired seed, the same 100-neuron descending-drive realization supplied a 100-MN pool in each simulator. Drive increased from baseline to a plateau and then returned to baseline. We compared the number of active MNs, total MN spikes, mean discharge rate among active MNs, spike-time coincidence, and

population discharge-rate trajectories across ten paired stochastic seeds.

Spike-time agreement was summarized over coincidence tolerances from ± 1 to ± 50 ms, including the ± 5 ms operating point highlighted in Fig. 2. Population discharge-rate trajectories were calculated with 50 ms bins and 300 ms smoothing. These comparisons quantify how well the differentiable backend preserves behavior across temporal scales. Exact spike-by-spike identity is not required for this comparison because small implementation-dependent perturbations can shift threshold crossings in nonlinear spiking models.

TABLE I
SIMULATED INTRINSIC-PROPERTY RANGES FOR THE JAXLEY AND NEURON BACKENDS, WITH PHYSIOLOGICAL REFERENCE RANGES FROM ZENGEL ET AL. [4]

Parameter	Unit	Jaxley	NEURON	Zengel
Input resistance	M Ω	0.53–1.79	0.5–1.8	0.1–3
Membrane time constant	ms	3.12–10.32	3–11.2	3–15
AHP magnitude	mV	4.59–5.8	4–4.8	1–10
AHP duration	ms	40.3–155	34.8–136	30–270
Rheobase current	nA	5.7–18.9	5.3–18.9	4–40

The literature values are from cat medial gastrocnemius MNs and are used here as contextual physiological bounds; AHP: afterhyperpolarization.

IV. RESULTS

In the representative 100-MN comparison, both backends showed the same qualitative recruitment structure: recruitment increased with descending drive, activity was sustained during the plateau, and discharge declined as the drive returned to baseline. Jaxley recruited 86 active MNs, compared with 77 in NEURON. Across ten paired stochastic realizations (Fig. 2a), this modest shift toward greater recruitment in Jaxley was consistent, yielding more active MNs and more total spikes (both $p < 0.001$). Mean discharge rate per active MN remained closely aligned in magnitude, with only a small systematic offset ($p = 0.016$).

At a ± 5 ms tolerance, the paired spike-time analysis identified 1,937 coincidences, and the matched fraction exceeded 80% at ± 50 ms (Fig. 2b). At the population level, agreement was particularly strong: the smoothed discharge-rate trajectories closely overlapped and reached $r = 0.996$ (Fig. 2c–d). The two backends therefore reproduce the same global temporal evolution of MN activity despite expected local shifts in individual spike timing.

Table I shows close overlap between the intrinsic-property ranges produced by Jaxley and the established NEURON backend, with both lying within the broad physiological ranges reported for cat medial gastrocnemius MNs [4]. This provides an additional plausibility check that transferring the MN stage to Jaxley preserves the intended biophysical operating regime.

V. DISCUSSION

The central result is that the MN stage of MyoGen can be transferred from NEURON to Jaxley while retaining the behavior that matters at the population level. Under identical

stochastic descending drive, both backends show orderly recruitment, similar discharge-rate magnitudes, physiological intrinsic-property ranges, and nearly overlapping population-rate trajectories. The correlation of $r = 0.996$ demonstrates that the temporal structure of the simulated motor pool is preserved after the change of simulator.

The remaining differences in recruitment, total spike count, and exact spike timing are expected when moving a nonlinear spiking model between two independent simulation frameworks. Small differences in numerical integration, event handling, discretization, or parameter implementation can shift threshold crossings and subsequently alter individual spike times without changing the overall motor-pool dynamics. In this context, the systematic offsets observed here are useful: they identify where the two implementations differ quantitatively while showing that these differences do not disrupt the global recruitment and discharge behavior of the model.

This makes Jaxley useful not merely as a second implementation, but as a differentiable backend for MyoGen. The new path provides direct gradients with respect to MN parameters while preserving the established simulation pipeline downstream. Further deterministic benchmarks, effect-size reporting, and more detailed spike-level comparisons can refine the correspondence between backends, but the present results already demonstrate a successful functional transfer at the level

of population dynamics.

The comparison with Zengel et al. [4] should be interpreted as a physiological plausibility check rather than direct human validation. Future work will extend the same framework to human MU recordings, gradient-based parameter recovery, and eventually objectives defined on force and EMG. Those steps build on the present result: a working differentiable MN backend that can be switched into MyoGen without losing the characteristic behavior of the simulated motor pool.

VI. CONCLUSION

We integrated Jaxley into MyoGen as a differentiable MN simulation path and showed that the backend can be changed while preserving the principal behavior of the simulated motor pool. Under matched stochastic drive, Jaxley and NEURON produced orderly recruitment, closely aligned firing-rate behavior, physiologically plausible intrinsic properties, and highly correlated population discharge-rate dynamics ($r = 0.996$). The expected implementation-specific differences in recruitment and exact spike timing provide targets for further numerical alignment rather than preventing practical use of the new backend. Jaxley therefore provides MyoGen with a functioning differentiable MN path and a foundation for gradient-based calibration toward MU, force, and EMG observations.

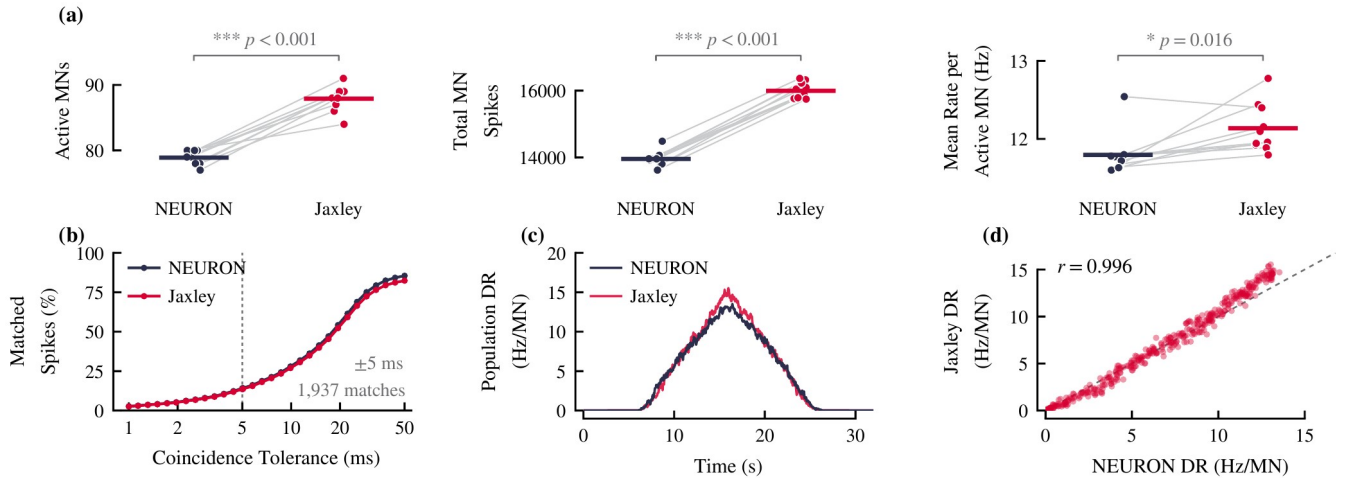


Fig. 2. Comparison of Jaxley and NEURON MN population dynamics. (a) Paired comparison across $n = 10$ descending-drive seeds for the number of active MNs, total MN spikes, and mean discharge rate per active MN; dots show individual seeds and gray lines connect paired realizations. The reported comparisons are $p < 0.001$, $p < 0.001$, and $p = 0.016$, respectively. (b) Percentage of spikes matched as a function of coincidence tolerance; 1,937 spikes were matched at ± 5 ms. (c) Smoothed population discharge-rate trajectories (50 ms bins, 300 ms smoothing), with $r = 0.996$. (d) Jaxley versus NEURON population discharge rate with identity line, $r = 0.996$.

REFERENCES

- [1] R. C. Simpetru et al., "MyoGen: Unified biophysical modeling of human neuromotor activity and resulting signals," *bioRxiv*, 2026, doi: 10.64898/2026.01.01.697284.
- [2] M. L. Hines and N. T. Carnevale, "The NEURON simulation environment," *Neural Computation*, vol. 9, no. 6, pp. 1179–1209, 1997, doi: 10.1162/neco.1997.9.6.1179.
- [3] M. Deistler et al., "Jaxley: differentiable simulation enables large-scale training of detailed biophysical models of neural dynamics," *Nature Methods*, vol. 22, pp. 2649–2657, 2025, doi: 10.1038/s41592-025-02895-w.
- [4] J. E. Zengel, S. A. Reid, G. W. Sypert, and J. B. Munson, "Membrane electrical properties and prediction of motor-unit type of medial gastrocnemius motoneurons in the cat," *Journal of Neurophysiology*, vol. 53, no. 5, pp. 1323–1344, 1985, doi: 10.1152/jn.1985.53.5.1323.
- [5] M. K. Chardon et al., "Supercomputer framework for reverse engineering firing patterns of neuron populations to identify their synaptic inputs," *eLife*, vol. 13, article e90624, 2024, doi: 10.7554/eLife.90624.
- [6] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952, doi: 10.1113/jphysiol.1952.sp004764.

- [7] A. J. Fuglevand, D. A. Winter, and A. E. Patla, "Models of recruitment and rate coding organization in motor-unit pools," *Journal of Neurophysiology*, vol. 70, no. 6, pp. 2470–2488, 1993, doi: 10.1152/jn.1993.70.6.2470.

A Local Multimodal AI Assistant for Wearable Edge Computing

Luka Phirtskhalava

Georgian Technical University, Tbilisi, Georgia

Abstract—This paper presents FRIDAY, a local multimodal artificial intelligence assistant designed for deployment on a wearable edge-computing platform. The system combines speech recognition, a locally executed large language model, neural text-to-speech synthesis, persistent conversational memory, computer control, and real-time face detection. The core system is deployed on an NVIDIA Jetson Orin Nano, allowing AI processing to be performed locally rather than relying entirely on remote cloud services. FRIDAY uses voice-based interaction as its primary interface and integrates a wearable camera and external microcontroller for perception and hardware feedback. The proposed architecture separates audio processing, artificial intelligence, memory, computer vision, and hardware-control functions into modular components. Functional evaluation demonstrates the feasibility of combining these components into a portable AI assistant capable of continuous interaction with its user. The system provides a foundation for future wearable AI and human-computer interaction research.

Index Terms—artificial intelligence, edge computing, multimodal systems, speech recognition, wearable computing

I. INTRODUCTION

Artificial intelligence assistants have become increasingly capable of understanding natural language, generating responses, and interacting with digital systems. However, many modern assistants depend heavily on cloud-based infrastructure, which introduces requirements for network connectivity and can increase dependence on remote computational resources. For wearable and robotic applications, these limitations can become particularly important because the assistant must operate in environments where connectivity, latency, and system availability may vary.

Edge computing provides an alternative approach by moving computational workloads closer to the user and the physical system. NVIDIA's Jetson platform is designed for edge AI applications including robotics, computer vision, and generative AI. The Jetson Orin Nano provides a compact computing platform with an integrated GPU and 8 GB of memory, making it suitable for experimentation with AI workloads on embedded systems.

This paper presents FRIDAY, a wearable AI assistant inspired by the fictional artificial intelligence systems found in science fiction. The objective of FRIDAY is not simply to

provide a conversational chatbot, but to integrate several AI and embedded-system capabilities into a single interactive platform.

The system accepts spoken commands through a microphone, converts speech into text, processes the user's request using a local language model or specialized command system, generates a response, and converts the response into speech. In addition, FRIDAY maintains persistent memory, provides computer-control functions, detects faces through a camera mounted on the wearable system, and communicates with an external microcontroller for hardware feedback.

The main contribution of this work is the integration of these components into a modular local AI architecture suitable for a wearable platform. The paper describes the system architecture, processing pipeline, memory system, computer-vision component, hardware interface, and functional evaluation.

II. RELATED WORK

A. Speech Recognition

Automatic speech recognition is an important component of voice-based human-computer interaction. Traditional speech-recognition systems often rely on task-specific models or manually engineered pipelines. More recent neural approaches have demonstrated improved robustness across different speakers, environments, and languages.

Whisper introduced a large-scale weakly supervised approach to speech recognition using approximately 680,000 hours of multilingual and multitask audio data. Its ability to perform speech recognition without task-specific fine-tuning makes it useful as a general speech-processing component [1].

FRIDAY uses neural speech recognition as the primary input mechanism. The speech-recognition component converts microphone input into textual commands that can subsequently be processed by the system.

B. Local Large Language Models

Large language models have enabled AI systems to process natural-language queries and generate context-sensitive responses. Cloud-based models provide access to substantial computational resources, but local model execution can provide advantages for applications where connectivity, privacy, and local processing are important.

Ollama provides a framework for running open language

models locally [2]. FRIDAY uses Ollama as the interface between the assistant's control system and its locally executed language model. A compact Qwen model is used because the wearable platform has significantly fewer computational and memory resources than a conventional cloud server.

C. Neural Text-to-Speech

Text-to-speech synthesis converts the generated response into audible speech. Piper is a local neural text-to-speech system designed for efficient speech synthesis on resource-constrained devices [3].

FRIDAY uses Piper to generate spoken responses. Local synthesis allows the response-generation pipeline to remain primarily on the edge device and avoids requiring a remote speech-synthesis service for every response.

D. Edge AI and Wearable Systems

Embedded AI platforms provide an opportunity to combine machine learning with sensors, cameras, microphones, and physical actuators. NVIDIA Jetson Orin Nano systems are specifically designed for edge AI and robotics workloads and provide GPU acceleration within a compact embedded platform [4].

FRIDAY extends this concept toward wearable human-computer interaction. Instead of treating the AI model, computer vision system, and hardware controller as independent applications, the proposed system integrates them through a central assistant architecture.

III. PROPOSED METHOD

The proposed FRIDAY system is designed as a modular edge-AI assistant. The architecture separates the system into several functional layers:

- 1) Audio input and speech recognition
- 2) Command and language processing
- 3) Local language-model inference
- 4) Conversational and factual memory
- 5) Text-to-speech generation
- 6) Computer vision
- 7) Computer and application control
- 8) Microcontroller communication

The central processing logic coordinates these components and determines how a user's input should be handled.

The system is primarily designed around two types of interaction.

The first is conversational interaction. When the user asks a general question or makes a conversational statement, the input is forwarded to the local language model. The generated response is then converted into speech.

The second is command-oriented interaction. When the system identifies a supported computer or hardware command, it can execute the corresponding operation directly instead of treating the command as a general language-model request.

This separation is important because deterministic operations such as opening an application or changing a hardware state should not depend entirely on probabilistic language-model generation.

A wake-word mechanism is also incorporated into the system. FRIDAY can remain in an inactive state until its activation condition is detected, after which speech processing is enabled. The architecture also supports an always-listening interaction mode for situations where continuous interaction is desired.

IV. SYSTEM ARCHITECTURE

A. Hardware Platform

The primary computing platform is an NVIDIA Jetson Orin Nano Developer Kit. The platform provides an embedded CPU/GPU system suitable for local AI inference, computer vision, and sensor integration.

The FRIDAY prototype connects several external components to the Jetson:

- USB microphone for audio input
- Speaker or audio output device for synthesized speech
- Camera mounted on the wearable mask
- Arduino microcontroller for hardware-status control
- Network connection for optional communication with external computers

The camera is physically positioned between the user's eyes on the wearable mask. This placement allows the computer-vision component to operate from approximately the same viewpoint as the wearer.

B. Audio Processing Pipeline

The audio pipeline begins by continuously monitoring the microphone input. Audio-processing parameters are used to distinguish meaningful speech from background silence and noise.

After activation, the recorded audio is passed to the speech-recognition component. The resulting transcription becomes the input to the command-processing layer.

The general pipeline is:

Microphone → *Audio Capture* → *Wake/Listening Logic* →
Speech Recognition → *Text*

This architecture allows the speech-recognition model to operate as one component rather than controlling the entire assistant.

C. AI Processing Layer

The AI processing layer determines whether an input should be interpreted as a predefined command or as a natural-language request.

For predefined commands, the command processor executes a deterministic operation. Examples include opening applications, obtaining system information, and interacting with connected hardware.

For general conversational requests, the input is forwarded to the local Ollama model. The model generates a response using the current conversational context.

The resulting pipeline is:

Text Input → *Command Processor* → *Command Execution*

or

Text Input $\rightarrow$ *Context/Memory* $\rightarrow$ *Local LLM* $\rightarrow$ *Response*

D. Conversational Memory

A persistent memory subsystem is included to prevent the assistant from losing all user-specific information when the application is restarted.

FRIDAY stores information in a local memory file and maintains conversational context for interactions. The memory system is separated from the language model itself. This allows the underlying model to remain a general-purpose pretrained model while external information provides additional context.

The memory subsystem has two primary purposes:

- 1) Maintaining relevant conversational context.
- 2) Storing explicitly identified facts that can be recalled in later interactions.

This architecture provides a simple form of long-term memory without requiring the language model itself to be retrained.

E. Text-to-Speech

After a response has been generated, FRIDAY passes the response text to the Piper neural text-to-speech engine.

The resulting audio is sent to the connected audio output device.

The complete conversational loop is therefore:

Speech $\rightarrow$ *Text* $\rightarrow$ *Understanding* $\rightarrow$ *Response* $\rightarrow$ *Speech*

This creates a natural voice-based interface without requiring the user to interact with a keyboard or graphical interface for ordinary conversations.

F. Computer Vision

FRIDAY incorporates a camera mounted on the wearable mask. The computer-vision subsystem performs real-time face detection from the camera stream.

Face detection provides FRIDAY with a visual perception capability that is independent of the conversational AI system. The detected visual information can be used by higher-level components to determine whether people are present in the assistant's field of view.

The vision subsystem is designed to operate independently from the speech pipeline so that visual processing does not require every camera frame to be sent to the language model.

G. Hardware Feedback

An Arduino microcontroller is connected to FRIDAY through a serial interface. The microcontroller provides a physical status-feedback mechanism for the assistant.

Different system states can be represented through hardware indicators, including:

- IDLE
- LISTEN
- THINK
- TALK
- SLEEP
- OFF

This allows the user to determine the current state of the assistant without requiring a graphical interface.

For example, when FRIDAY is processing a request, the microcontroller can indicate the THINK state. During speech recognition, the LISTEN state can be activated, while generated speech can correspond to the TALK state.

H. Computer and Remote Control

FRIDAY also provides interaction with the user's computer environment. Supported commands can be used to open applications and perform predefined desktop operations.

A network-based communication component was also developed to allow the Jetson-based assistant to communicate with a separate computer. This enables the wearable device to act as an interface to additional computing resources when required.

V. IMPLEMENTATION

The software implementation follows a modular Python architecture. Rather than placing all functionality inside a single program, the project separates major functions into individual modules.

The current architecture follows the structure:

```
FRIDAY/
  main.py
  config.py
  ai/
  audio/
  skills/
  utils.py
  memory.json
```

The main module coordinates the startup process and connects the individual components. Configuration values such as model selection, wake-word behavior, audio parameters, and hardware interfaces are separated from the main program.

The audio module manages microphone input and speech-processing operations. The AI module provides the interface to the local language model. The skills module contains deterministic operations that can be executed by the assistant. Utility functions provide shared functionality used by multiple components.

This modular structure allows individual components to be replaced without redesigning the entire system. For example, a different speech-recognition model can be introduced without changing the hardware-control subsystem.

The local language model is selected with consideration for the limited memory and computational resources of the embedded platform. Smaller models allow the assistant to maintain interactive functionality while leaving resources available for other components such as audio processing and computer vision.

VI. EVALUATION

A. Evaluation Method

FRIDAY was evaluated as an integrated interactive system rather than as a single machine-learning model. The evaluation focused on whether the major components could operate together on the embedded platform.

The following functional categories were considered:

- 1) Speech input and transcription
- 2) Wake-word activation
- 3) Natural-language interaction
- 4) Command execution
- 5) Persistent memory
- 6) Speech output
- 7) Face detection
- 8) Hardware-state feedback
- 9) Communication with an external computer

B. Voice Interaction

The voice-interaction pipeline was tested by providing spoken commands through the connected microphone. FRIDAY processes the captured speech and converts it into text before passing it to the command or AI-processing layer.

The system supports both conversational requests and predefined commands. This distinction allows common operations to be executed without requiring a language model to generate an executable instruction.

C. Local AI Processing

The local language model was tested through conversational queries and general requests. Running the model on the Jetson allows the assistant to continue processing requests without requiring every conversational interaction to be sent to a remote AI service.

However, the limited resources of the embedded platform impose constraints on model size. The use of a compact model represents a trade-off between local execution and model capability.

D. Memory

The memory subsystem was evaluated by storing information during one interaction and attempting to retrieve it in a later interaction or after restarting the assistant.

Persistent storage allows FRIDAY to retain selected information independently of the temporary context window of the language model.

E. Computer Vision

The camera subsystem was tested using the wearable camera mounted between the user's eyes. The face-detection component processes the camera stream and identifies faces within the field of view.

This demonstrates that the same embedded platform can support both language-based interaction and visual perception.

F. Hardware Integration

The Arduino interface was tested by transitioning FRIDAY between different operating states. Hardware indicators provide visual feedback corresponding to states such as LISTEN, THINK, TALK, IDLE, SLEEP, and OFF.

This provides a direct connection between software state and the physical wearable system.

G. Limitations

The current implementation has several limitations.

First, the Jetson Orin Nano has considerably fewer resources than the servers normally used for large language-model

inference. Consequently, the selection of a compact local model is necessary for practical operation.

Second, speech-recognition performance can be affected by microphone quality, background noise, distance from the speaker, and pronunciation.

Third, the current face-detection component provides visual detection but does not by itself establish identity or understand complex scenes.

Fourth, persistent memory is implemented as an external software layer rather than being intrinsically learned by the language model. This requires additional logic for selecting and retrieving useful information.

Finally, the current evaluation is primarily functional. A more extensive quantitative evaluation should measure speech-recognition error rate, language-model response latency, speech-synthesis latency, face-detection accuracy, CPU/GPU utilization, memory consumption, and power consumption under different workloads.

VII. CONCLUSION

This paper presented FRIDAY, a local multimodal AI assistant designed for deployment on a wearable edge-computing platform. The system integrates speech recognition, local language-model inference, neural text-to-speech synthesis, persistent memory, computer control, computer vision, and microcontroller-based hardware feedback.

The architecture demonstrates how multiple AI and embedded-system components can be combined into a single interactive assistant while maintaining modular separation between perception, reasoning, memory, and hardware control.

A significant characteristic of the proposed system is its local processing architecture. By executing major components on an NVIDIA Jetson Orin Nano, FRIDAY can provide voice interaction and AI functionality without making cloud connectivity a fundamental requirement for every operation.

The current implementation provides a foundation for further development of wearable AI systems. Future work includes improving long-term memory and retrieval, optimizing inference for the embedded GPU, expanding computer-vision capabilities, improving multimodal interaction, adding more sophisticated hardware interfaces, and performing quantitative benchmarking of latency, resource utilization, recognition accuracy, and power consumption.

FRIDAY demonstrates the feasibility of building a portable AI assistant that combines conversational intelligence with physical perception and interaction in an embedded environment.

REFERENCES

- [1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," arXiv:2212.04356, 2022.
- [2] Ollama, "Ollama: Run Open Models," Ollama, 2026.
- [3] Piper, "Piper: Fast and Local Neural Text-to-Speech Engine," Piper TTS Speech Synthesis, 2026.
- [4] NVIDIA Corporation, "Jetson Orin Nano Developer Kit User Guide," NVIDIA Developer Documentation, 2026.

Allreduce from Scratch: Comparing Self-Implemented Protocol Variants with Open MPI

Lotte Richter

Abstract—Collective communication is fundamental to distributed-memory parallel applications, and its performance depends strongly on the underlying communication and synchronization strategy. This paper investigates how `MPI_Allreduce` can be constructed from point-to-point communication. Four Allreduce variants were implemented: reduce-broadcast, crossing, non-crossing and dynamic. Their runtime behaviour is evaluated on a single shared-memory host with 4 to 32 MPI processes and compared with the native `MPI_Allreduce` provided by Open MPI. Crossing messages let all processes communicate simultaneously, but risk deadlocks with blocking sends, which the non-blocking variant avoids. For the tested single-integer workload, the crossing and dynamic variants achieve the lowest runtimes among the self-implemented approaches and remain close to the native implementation. The results highlight the trade-off between communication parallelism, runtime, and deadlock avoidance when implementing collective operations from scratch.

Index Terms—Allreduce, collective communication, MPI, parallel systems

I. INTRODUCTION

Parallel applications rely on efficient communication between processes to exchange data and coordinate computation. The Message-Passing Interface (MPI) provides a standardized programming model for such applications and offers a collection of collective communication operations. Among these operations, `MPI_Allreduce` is particularly communication-intensive, as values from all participating processes must be combined and the result must be distributed to all participants [1].

The performance of Allreduce depends on how these communication steps are organized. Different algorithms use different communication schedules and synchronization mechanisms, resulting in different numbers of communication rounds and different amounts of communication overhead.

This paper compares several self-implemented Allreduce variants in C with the native implementation provided by Open MPI. Their runtime behaviour is evaluated for 4, 8, 16, and 32 MPI processes on a single shared-memory host.

II. RELATED WORK

Optimizing MPI collective operations, especially Reduce and Allreduce, has been an active research area since the introduction of the standard. Its performance depends strongly on the algorithm used to organize communication and synchronization. Thakur et al. [2] show that a single collective operation can require different algorithms depending on message size and the number of processes. For `MPI_Allreduce`, they discuss several approaches and

demonstrate that algorithm selection can have a substantial impact on performance. Proficz et al. [3] evaluate six Allreduce algorithms under different process-arrival patterns and system conditions.

Ruefenacht et al. [4] investigate generalized recursive-doubling approaches for Allreduce and show that modifications to the underlying communication pattern can improve performance, particularly for small messages.

These works primarily evaluate or optimize complete Allreduce algorithms under specific hardware or workload conditions. In contrast, we start from plain MPI point-to-point operations and build several Allreduce protocols from scratch. This makes differences in communication ordering, synchronization, and deadlock behaviour observable and comparable on a single shared-memory host.

III. PROPOSED METHOD

Our approach builds Allreduce entirely from MPI point-to-point primitives rather than relying on a library-provided collective.

The most direct way to realize Allreduce is to combine two simpler collective patterns: a Reduce that combines all values onto a single process, followed by a Broadcast that distributes the result back to every process. In this naive variant the reduction is first assembled at a designated root before the result is distributed to all processes.

The remaining three implementations use a symmetric pairwise communication structure that differs in ordering and synchronization mechanisms. See Section IV for detailed descriptions of the crossing, non-crossing, and dynamic variants.

The four implementations are compared against each other and against the native `MPI_Allreduce` of Open MPI, which serves as an external reference rather than an additional design point. Open MPI selects among several internal algorithms (e.g., recursive doubling, ring, Rabenseifner's algorithm) depending on message size and process count [5]; we did not determine which one was used for our single-integer workload, so this comparison should be read as a comparison against Open MPI's default behaviour rather than against one specific known algorithm.

IV. ALGORITHM DESCRIPTION

All four Allreduce variants operate on a single integer per process and use `MPI_INT` as the only supported datatype. The native `MPI_Allreduce` used for comparison is called directly through the MPI library and is therefore not described further here, since its internal algorithm is not part of our implementation.

A. Naive: Reduce followed by Broadcast

The naive implementation constructs Allreduce from two existing collective operations, a Reduce phase and a Broadcast phase. In the Reduce phase, values are combined onto process 0 along a binary tree over $\lceil \log_2(p) \rceil$ rounds. In the Broadcast phase, the result is distributed from process 0 using recursive doubling, halving the communication stride every round.

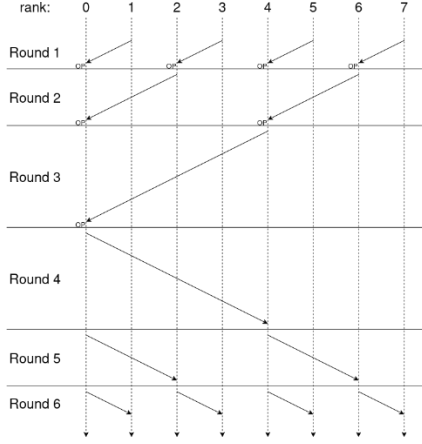


Figure 1: Communication pattern of the naive variant

B. Crossing: Fixed Schedule with Simultaneous Sends

The crossing implementation uses a fixed pairwise communication schedule. For a process with rank r , the communication partner in round i is determined by $partner = r \oplus 2^i$ where $\oplus$ denotes the bitwise XOR operation for p processes. In the crossing variant, both processes in a pair send to and receive from each other in the same round, without any ordering between the two operations. Since both processes issue a blocking send before either has posted the corresponding receive, both sends may remain blocked. If the MPI implementation does not buffer the messages internally, this can result in a deadlock. This variant additionally requires p to be a power of two, since every round involves all processes.

Figure 2 illustrates the resulting crossing communication pattern. The simultaneous exchange of messages is repeated for each communication round, with the resulting intermediate values being used in the subsequent round.

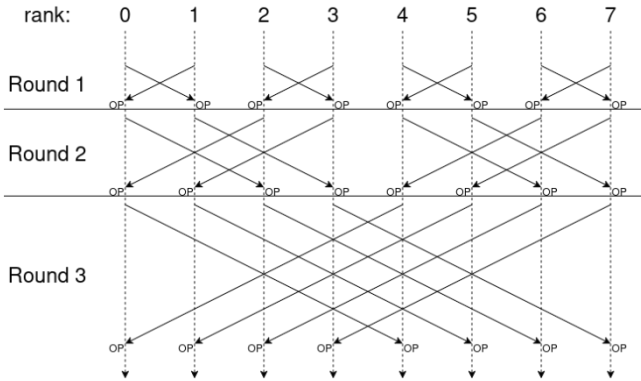


Figure 2: Communication pattern of the crossing variant

C. Non-Crossing (resolved-adopt): Ordered Sends and Receives

The non-crossing variant uses the same partner computation but avoids simultaneous blocking sends by fixing, for each pair, which process sends first and which receives first.

The process that receives first applies the reduction operation and sends the combined result back; the process that sent first simply adopts this result.

The communication order and the participating process pairs for the individual rounds are shown in Figure 3.

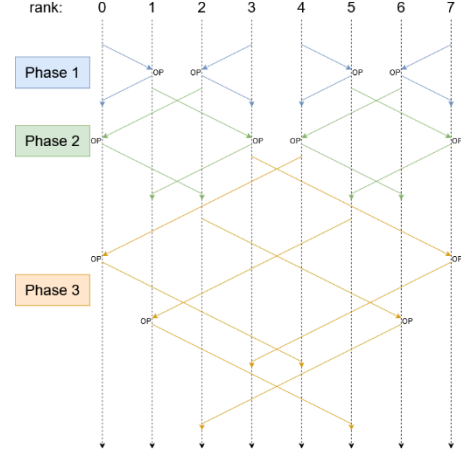


Figure 3: Communication pattern of the non-crossing variant

D. Dynamic: Non-Blocking Communication

The dynamic implementation uses the same pairwise communication structure as the crossing variant (Figure 2). The difference lies in the communication mechanism: instead of using blocking point-to-point operations, the dynamic variant uses non-blocking MPI_Isend and MPI_Irecv operations. The communication requests are subsequently completed using MPI wait operations before the associated data is reused or modified, as shown in Listing 1:

```
isend(buffer, target, send_request)
irecv(buffer_recv, target, recv_request)
wait(recv_request); wait(send_request)
buffer = op(buffer, buffer_recv)
```

Listing 1: Dynamic (non-blocking) exchange per round

This separates the initiation of communication from its completion and avoids the blocking behaviour of the crossing variant.

V. EVALUATION

We evaluated all four variants and MPI_Allreduce on a single shared-memory server from the HTW Dresden with Open MPI, using 4, 8, 16, and 32 processes. Each value is the mean runtime of 20,000 calls exchanging a single integer with the minimum operation.

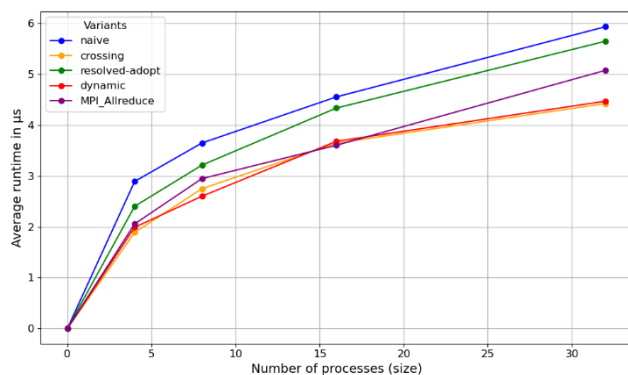


Figure 4: Mean runtime of the four variants and MPI_Allreduce over the number of processes

Figure 4 shows the mean runtimes of all four variants alongside MPI_Allreduce. **Crossing** and **dynamic** are the fastest self-implemented variants, since both allow maximum parallel communication per round. Dynamic is not noticeably faster than crossing despite using non-blocking operations, because the explicit MPI_Wait calls resynchronize both processes each round; its advantage is avoiding the deadlock risk of crossing (Section IV-B), not extra speed. **Naive** and **resolved-adopt** are both slower and close to each other: naive needs twice as many rounds but a simpler per-round algorithm, while resolved-adopt needs half as many rounds but computes send_first(i, rank) every round, and these effects appear to roughly cancel out.

Compared to MPI_Allreduce, crossing and dynamics are about 3-13% faster at 4, 8, 32 Processes, while naive and resolved-adopt are 9-41% slower. At 16 processes all four are close to MPI_Allreduce, with crossing and dynamic marginally slower rather than faster, as visible in Figure 4. Since we did not determine which internal algorithm Open MPI uses (Section III), this crossover cannot be attributed to a specific cause.

Our variants only support MPI_INT, count = 1, and a small set of operations, and all but naive require a power-of-two process count, while MPI_Allreduce support arbitrary datatypes, sizes, and process counts. That simple point-to-point implementations can still match it for small messages is the main takeaway of this comparison.

VI. CONCLUSION

This paper investigated how communication ordering and synchronization mechanisms affect the runtime of manually implemented Allreduce variants. The results show that the crossing and dynamic variants, which use the fewest communication rounds with maximum parallelism per round, were consistently fastest and competitive with MPI_Allreduce. The dynamic variant matched this performance while also avoiding the deadlock risk of the crossing variant, showing that performance and deadlock safety need not be in conflict.

Our results are limited to single-integer messages, power-of-two process counts, and a single shared-memory host; extending the comparison to larger messages, arbitrary process count, and distributed systems remains for future work.

VII. ACKNOWLEDGMENT

This paper builds on the results of a project seminar [6] supervised by Prof. Dr.-Ing. habil. Peter Sobe. The underlying implementation and measurements were developed jointly with Christoph Born; the author thanks him for this collaboration and Prof. Sobe for his supervision and guidance throughout the seminar.

REFERENCES

- [1] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard, Version 5.0," June 5, 2025. [Online]. Available: <https://www.mpi-forum.org/docs/mpi-5.0/mpi50-report.pdf>. [Accessed: Sep. 21, 2026].
- [2] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in MPICH," The International Journal of High Performance Computing Applications, vol. 19, no. 1, pp. 49–66, 2005, doi:[10.1177/1094342005051521](https://doi.org/10.1177/1094342005051521)
- [3] J. Proficz et al., "Investigation into MPI All-Reduce performance in a distributed cluster with consideration of imbalanced process arrival patterns," in Advanced Information Networking and Applications (AINA 2020), L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, Eds. Springer, 2020, vol. 1151, pp. 696–707, doi: [10.1007/978-3-030-44041-1_72](https://doi.org/10.1007/978-3-030-44041-1_72).
- [4] M. Ruefenacht, M. Bull, and S. Booth, "Generalisation of recursive doubling for AllReduce: Now with simulation," Parallel Computing, vol. 69, pp. 24–44, 2017, doi: [10.1016/j.parco.2017.08.004](https://doi.org/10.1016/j.parco.2017.08.004).
- [5] Open MPI Project, "The tuned collective component," Open MPI Documentation, v5.0.4. [Online]. Available: <https://docs.open-mpi.org/en/v5.0.4/tuning-apps/coll-tuned.html>. [Accessed: Sep. 24, 2026].
- [6] L. Richter and C. Born, "Programmierung paralleler Systeme: Implementation und Leistungsanalyse von Kommunikationsmustern (Reduce, Broadcast, Allreduce) mittels MPI," Project Seminar, HTW Dresden, supervised by P. Sobe, Jan. 2025.

MAMAgotchi Kitchen: A Tangible Feeding Station for a Shared Virtual Pet

M. Ruprecht, E. Kirchner and M. Kadabashi

Abstract—Tangible interfaces can make abstract social themes such as care work accessible to children. This paper presents the kitchen station of MAMAgotchi, a room-based tangible installation for children aged 7–11 developed as a speculative design response to the exhibition MAMA at the Deutsche Hygiene-Museum. Children feed a virtual pet by placing wooden toy fruit into the station, cranking a 3D-printed conveyor belt, and delivering the food into a basket. An ESP32-based microcontroller combines distance and weight sensing, drives an LED and a display showing mood animations, and reports food values to a central API that maintains a hunger score shared by all stations. Described is the iterative design process, the software workflow, and the system integration. The prototype shows that mechanical interaction and screen-based feedback can be combined into an understandable feeding activity; open issues concern LED reliability, power supply, and basket handling.

Index Terms—child-computer interaction, museum installations, physical computing, tangible user interfaces, virtual pets

I. INTRODUCTION

Digital pets such as the Tamagotchi popularized the idea that caring for a virtual creature means repeatedly satisfying its needs. In most screen-based pets, however, feeding is reduced to selecting an icon or pressing a button, which conveys little of the physical effort and repetition that characterizes everyday care work. Tangible user interfaces, which couple digital information with physical objects and manipulation, offer an alternative [1].

MAMAgotchi is a tangible, room-based interactive concept developed in response to the exhibition MAMA at the Deutsche Hygiene-Museum. It was not created as a final installation for the museum but as a speculative design exploring possible interactive formats. Aimed at children aged 7–11, it translates themes such as motherhood, care work, and social images of mothers into a playful spatial experience consisting of care-giving stations for feeding, soothing, playing, and dancing.

This paper focuses on the kitchen station, which is responsible for feeding the MAMAgotchi. The design challenge is threefold: the feeding action must be understandable and physically engaging for children, the prototype must be robust and maintainable, and the station must contribute to a global pet state shared with the other stations. Our contributions are (1) the interaction concept and mechanical design of a crank-operated conveyor for feeding, (2) an embedded software workflow with synchronized LED, display, and sound feedback, and (3) the integration of the station into the shared need-based system. Within MAMAgotchi, each room is a tangible station: the bedroom creates a calm sleeping situation through touch, light, covering, and an alarm clock; the dance station “Burning Feet” turns music, movement, and collaboration into a care activity without relying on a

screen; and the playroom entertains the pet through rhythm games [2].

II. RELATED WORK

Sensors form the bridge between the physical and the digital layer of any tangible interface. A review distinguishes several families of sensing technology. Depth cameras such as the Kinect, used in frameworks like dSensingNI, offer flexible, marker-free detection of hands and objects but suffer from occlusion and noisy data. Marker-based tracking of passive objects is further divided into outside-in approaches, where an external camera observes markers on the objects, as in Tangible Tiles, and inside-out approaches, where the sensing is embedded in the object itself, as in the Dothraki system, which trades higher hardware complexity for robustness against occlusion. Beyond cameras, the review discusses ultrasonic FMCW sensors for lighting-independent gesture recognition, capacitive sensing for multi-user touch and proximity detection, wearables combining inertial and near-infrared sensors, acoustic and pressure sensors for contact-based input, and electromagnetic or NFC tracking for object identification and localization [3].

The review concludes that no single sensing technology is ideal for every tangible interface and that the choice depends on the intended interaction, environment, and user experience [3]. For the kitchen station, this motivated a deliberately simple, camera-free combination: a time-of-flight distance sensor, which shares its optical distance-ranging principle with the ultrasonic and depth-camera approaches but avoids their computational and lighting-related overhead, detects food entering the conveyor, while a pressure-based weight sensor determines how much food has actually reached the basket. This combination provides robust, line-of-sight-independent, single-purpose sensing that is well suited to an unsupervised, child-operated exhibit, at the cost of the richer position and identity information that camera- or marker-based tracking would offer.

On the technical side, the stations communicate wirelessly with a central controller using ESP-NOW, a connectionless, low-latency protocol for ESP32-based microcontrollers [4]. To lower the integration effort for the station groups, an interface group provided an Arduino library that abstracts this protocol [5], while a model making group supported the physical construction of all stations [6]. Our work builds on this shared foundation. It differs from a button-based digital pet in that feeding requires the child to place food, operate a crank, and empty a basket, and it differs from the sibling stations in its mechanical transport mechanism combined with weight-based sensing.

III. INTERACTION CONCEPT AND DESIGN PROCESS

A. Interaction Concept

The kitchen station represents the food preparation area of the MAMAgotchi ecosystem. When the pet is hungry, players place food into the station; wooden toy fruit is used for this purpose. The system detects the food automatically and transports it via a conveyor belt that has to be cranked by hand, mimicking the feeding process. The food is delivered into a basket.

Visual and acoustic feedback accompanies the entire process. An LED indicates the current state of the basket, and animated GIFs on the display visualize the pet's mood depending on its hunger level. Once enough food has been delivered, the MAMAgotchi becomes full and reacts with a happy animation. If the player continues feeding, it enters an "overfed" state accompanied by a special animation and a sound effect. When the basket is full, a popup asks the player to empty it in order to obtain "new" food.

B. Design Process and Modelling

Development started with several sketches exploring different layouts. The main objectives were to keep the feeding process clearly visible and to keep all electronic components easily accessible for maintenance. The parts were then modelled in CAD software. The housing is a wooden construction that combines the actual model, a 3D-printed belt, a basket, wooden fruit, and sensors.

Special attention was given to the conveyor mechanism. Multiple iterations were necessary to determine suitable dimensions for the rollers, guides, and basket, and the clearances between moving parts had to be adjusted several times so that food is transported reliably without getting stuck. The 3D printing was carried out in cooperation with the model making group; initial print quality issues required further iterations and manual adjustment with cutting tools. The modular design provides dedicated mounting points for the microcontroller, sensors, LED, display, and cable routing, which results in a clean and maintainable assembly.

IV. SYSTEM ARCHITECTURE

A. Hardware

The station is controlled by an ESP32-based M5Stack StampS3 board that coordinates all components (Table I). A time-of-flight distance sensor detects food entering the conveyor, an RGB LED gives immediate state feedback, a Tab5 display plays animated GIFs of the pet's emotions, a speaker outputs sound effects for important events, and a weight sensor (Unit Mini Scales) weighs the food arriving in the basket.

TABLE I
HARDWARE COMPONENTS OF THE KITCHEN STATION

Component	Role in the station
M5Stack StampS3 (ESP32)	Coordination, state handling, communication
ToF distance unit	Detects food entering the conveyor
Unit Mini Scales	Weighs food delivered to the basket
RGB LED	Immediate state feedback
Tab5 display with speaker	Animated GIFs of the pet's mood
3D-printed belt, crank	Manual transport of the food

B. Software

The software follows a simple state-based workflow (Fig. 1). In the *Hungry* state, the LED is green and the display shows the hunger animation. Once food is detected, the station switches to *Food on Conveyor*; the conveyor must now be cranked to transport the food toward the basket, and the LED turns yellow. After enough food has been delivered, the station enters *Basket Full / Satisfied*: the LED turns red and a happy animation is shown. Additional feeding beyond this point triggers the *Overfed* state with a special animation and an audio notification.

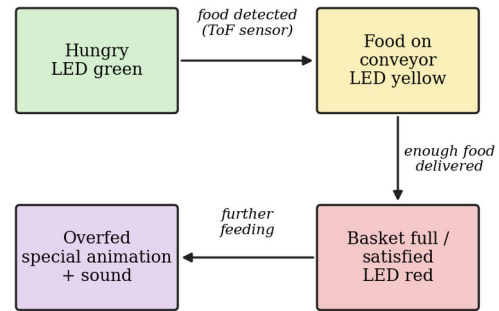


Fig. 1. State workflow of the kitchen station.

The control software continuously monitors the sensor values and updates the current state accordingly. Every state change immediately affects both the physical hardware and the graphical output, which creates a synchronized user experience. The modular structure makes it straightforward to add sensors, animations, or gameplay mechanics.

C. Integration into the Shared System

All MAMAgotchi stations are connected to one shared system (Fig. 2). Stations transmit their data via ESP-NOW to a central M5Stack AtomS3R controller using a shared data structure containing score, blocked flag, station identifier, and MAC address; a station is registered automatically upon its first message. The controller forwards the data as compressed JSON over USB serial to a Raspberry Pi 5, whose Python business logic maintains the authoritative pet state and whose JavaScript front end visualizes it [5].

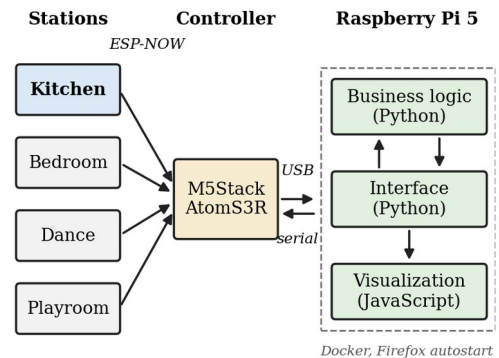


Fig. 2. Communication architecture connecting the kitchen and the other stations to the central system.

Whenever a child successfully feeds the MAMAgotchi, the kitchen sends the corresponding food value to the central system, which updates the overall status and returns

the current global hunger score. Feeding increases the score, whereas it gradually decays over time or through activities at other stations, such as the playroom. The business logic models the needs movement, energy, satiety, and play; if a need falls below a critical threshold, the stations serving the other needs are temporarily blocked. A critically low satiety, for example, blocks the sleep, dance, and play stations until the kitchen has been used [5]. The kitchen station itself is never blocked by other stations. In this way, every station contributes to the same virtual pet, which yields a connected and collaborative play experience.

V. EVALUATION

The evaluation reported here is qualitative and based on functional testing of the prototype throughout its development; quantitative measurements and a study with the target group are not part of this paper. The prototype (wooden housing, 3D-printed conveyor, wooden fruit, sensors, LED, display) was assembled and tested as a complete unit (Fig. 3), and all communication steps were tested together with the individual station groups.

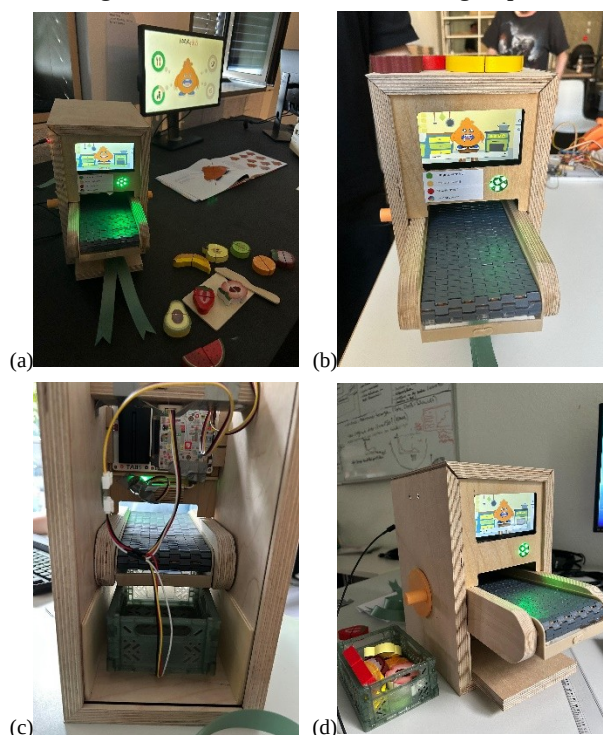


Fig. 3. Finished prototype: (a) exhibition setup with toy fruit and central pet display, (b) front view, (c) back view with electronics, conveyor, and basket, (d) side view with conveyor handle.

Mechanics. The conveyor was the most demanding component. Roller, guide, and basket dimensions as well as clearances required several iterations before food was transported without jamming. **Interaction and feedback.** The four states are reflected simultaneously by LED, display animation, and sound, so that the feeding progress remains legible throughout the process. **Integration.** The station reports food values to the shared system without being blocked by other stations, so that feeding directly influences the global pet state.

Several limitations remain. The LED occasionally malfunctions and will probably have to be exchanged. The display currently runs on battery because the model offers

no room for a charging cable. The mechanism for removing and reinserting the basket needs improvement, since the basket has to be placed precisely over the weight sensor. Moreover, the model is not yet painted. On the system level, the central pet currently provides limited visual feedback on state changes and no dedicated indicator for blocked stations.

VI. CONCLUSION

We presented the kitchen station of MAMAgotchi, which combines mechanical engineering, electronics, and embedded software into a tangible feeding activity for children. By requiring children to place food, crank a conveyor, and empty a basket, and by answering each step with LED, animation, and sound feedback, the station turns a simple feeding action into an engaging gameplay element and gives the virtual pet a stronger sense of personality and responsiveness. Through the shared hunger score, the station is embedded in a collaborative system of stations that reflects the metaphor of ongoing care work.

Future work includes replacing the LED, providing a charging solution for the display, improving the basket mechanism, and painting the model. Further extensions comprise additional sensors, animations, and gameplay mechanics, and an evaluation with children of the target age group.

ACKNOWLEDGMENT

Disclosure of AI use: Figs. 1 and 2 were created with the assistance of a generative AI tool (Claude, Anthropic). Generative AI (Claude, Anthropic) was also used for language editing and for structuring the manuscript text. All AI-generated content was reviewed, edited, and verified by the authors, who take full responsibility for the content of this paper.

REFERENCES

- [1] O'Rourke, Anne. Caring about virtual pets: an ethical interpretation of Tamagotchi. University of Wollongong. Journal contribution. 1998. [Online]. Available: <https://hdl.handle.net/10779/uow.27825507.v1>
- [2] "MAMAgotchi," Tangible Interaction Project Blog, HTW Dresden, 2026. [Online]. Available: <https://tangibleinteraction.github.io/projects/mamagotchi/>
- [3] E. Kirchner, M. Ruprecht, and M. Kadabashi, "Sensing and tracking in tangible interfaces," Tangible Interaction Theory Blog, HTW Dresden, 2026. [Online]. Available: <https://tangibleinteraction.github.io/theory/sensing-and-tracking-in-tangible-interfaces/>
- [4] Espressif Systems, "ESP-NOW overview," ESP-IDF Programming Guide, 2026. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html
- [5] J.-R. Schuster, J. Engel, H. L. Nguyen, and F. Adam, "Communication infrastructure and state visualization," MAMAgotchi project blog, HTW Dresden, 2026. [Online]. Available: https://tangibleinteraction.github.io/projects/mamagotchi/central_infrastructure/
- [6] A. Pech, O. Mey, and S. Beckert, "Model making," MAMAgotchi project blog, HTW Dresden, 2026. [Online]. Available: <https://tangibleinteraction.github.io/projects/mamagotchi/model-making/>

AI-Driven Longitudinal Monitoring of Neovascular Age-Related Macular Degeneration Progression via Optical Coherence Tomography Analysis

Mariam Surguladze

Georgian Technical University, Tbilisi, Georgia

Abstract—Neovascular age-related macular degeneration (nAMD) requires long-term longitudinal monitoring to guide treatment decisions. Conventional clinical evaluation relies on visual, subjective inspection of Optical Coherence Tomography (OCT) scans, which suffers from intra- and inter-observer variability. In this work, we present a cloud-native artificial intelligence (AI) decision-support framework deployed on Azure Machine Learning that transitions from isolated binary OCT classification (which achieved 98.04% accuracy post-augmentation) to patient-wise longitudinal comparison. By pairing sequential B-scans from consecutive clinical visits, the system extracts CNN abnormality probabilities (P_{Time1} , P_{Time2}) and calculates a probabilistic differential progression indicator ($\Delta AI = P_{Time2} - P_{Time1}$). Based on threshold parameters, scan pairs are stratified into Improved, Stable, or Worsened categories. Furthermore, we outline a methodological roadmap leveraging the RETOUCH benchmark dataset and 3D U-Net/V-Net architectures to achieve volumetric segmentation and quantitative biomarker tracking for intraretinal fluid, subretinal fluid, and central retinal thickness.

Index Terms—Artificial intelligence, Longitudinal progression monitoring, Neovascular age-related macular degeneration, Optical coherence tomography

I. INTRODUCTION

Neovascular age-related macular degeneration (nAMD) is a progressive retinal pathology requiring continuous longitudinal assessment and timely clinical intervention. Optical Coherence Tomography (OCT) is the gold standard imaging modality for visualizing pathological hallmarks, including intraretinal fluid (IRF), subretinal fluid (SRF), pigment epithelial detachment (PED), and changes in macular thickness. In current ophthalmic workflows, B-scan interpretation is performed manually by clinicians, introducing subjectivity and potential inconsistencies.

In the initial phase of this research, a PyTorch convolutional neural network (CNN) was trained to classify individual B-scans into "Normal" or "Pathological". While the untrained model performed near chance level (~51%), supervised training improved accuracy to ~90%, and data augmentation yielded 98.04% accuracy. However, binary classification merely confirms pathology presence—a fact already known for diagnosed

nAMD patients. It provides no insight into disease dynamics, such as whether fluid burden has resolved, stabilized, or exacerbated following anti-VEGF therapy.

To address this limitation, this paper introduces a patient-wise longitudinal comparison framework that evaluates sequential OCT B-scans ($V1 \rightarrow V2$) from the same individual. By tracking variations in model abnormality probability (ΔAI), the system yields a standardized AI progression indicator to assist clinicians in evaluating disease trajectories.

II. RELATED WORK

Automated retinal image analysis has evolved rapidly, moving from single-image diagnostic classification toward deep learning-driven biomarker tracking.

Clinical AI in Ophthalmology: Prior works by Kermany et al. and Leingang et al. demonstrated deep learning capabilities in detecting retinal pathologies. Schmidt-Erfurth et al. highlighted the importance of AI in tracking fluid volume dynamics during anti-VEGF treatment regimens.

Deep Learning Architectures for OCT: Fast single-stage detectors like YOLO allow rapid bounding-box screening across large longitudinal B-scan datasets. For pixel-level anatomical characterization, U-Net, 3D U-Net, V-Net, and DeepLabV3+ provide semantic segmentation of retinal layers and fluid compartments. Multi-instance tools like Mask R-CNN isolate individual fluid pockets, while vision transformer hybrids (TransUNet, Swin-UNet) model long-range spatial context. A comparison of the principal architectures considered in this study is provided in Table I.

TABLE I
DEEP LEARNING ARCHITECTURES FOR OCT ANALYSIS

Architecture	Key Strengths	3D / Longitudinal
YOLO	Real-time object detection, fast screening	No
U-Net / 3D U-Net	High-precision layer/fluid segmentation	Yes
Mask R-CNN	Instance segmentation of fluid pockets	No (2D Slices)

Architecture	Key Strengths	3D / Longitudinal
V-Net	Voxel-wise 3D volume segmentation	Yes
DeepLabV3+	Boundary-sensitive layer detection	No
TransUNet / Swin	Long-range context & progression detection	Yes

Benchmark Datasets: The Retinal OCT Fluid Detection and Segmentation Benchmark (RETOUCH) provides expert-annotated ground-truth masks for IRF, SRF, and PED, offering the gold-standard dataset for training semantic segmentation models.

III. PROPOSED METHOD

The proposed system relies on intra-patient chronological pairing rather than isolated B-scan analysis.

A. AI Progression Indicator (ΔAI)

Given two consecutive abnormal B-scans from a patient (V1 at t1 and V2 at t2), the pre-trained CNN computes predicted abnormality probabilities P_Time1 and P_Time2 . The AI progression differential is defined as:

$$\Delta AI = P_Time2 - P_Time1$$

where $\Delta AI < 0$ indicates reduced abnormality probability at follow-up (improvement), and $\Delta AI > 0$ indicates increased abnormality probability (worsening).

$$\begin{aligned} \text{Status} &= \text{Improved, if } \Delta AI < -T \\ \text{Status} &= \text{Stable, if } |\Delta AI| \leq T \\ \text{Status} &= \text{Worsened, if } \Delta AI > T \end{aligned}$$

B. Progression Stratification

Using a predefined sensitivity threshold T (derived empirically or via ROC curve optimization), longitudinal status is categorized as:

C. Cloud Architecture & Explainability

The framework is deployed on Microsoft Azure ML. Raw scans stored in Azure Blob Storage undergo standardized processing before PyTorch inference. Explainable AI (XAI) modules using Grad-CAM generate visual heatmap overlays on target regions of interest (ROIs) to provide algorithmic transparency.

IV. SYSTEM ARCHITECTURE & WORKFLOW

The longitudinal comparison framework operates through a structured multi-stage data processing and inference pipeline, illustrated in Fig. 1.

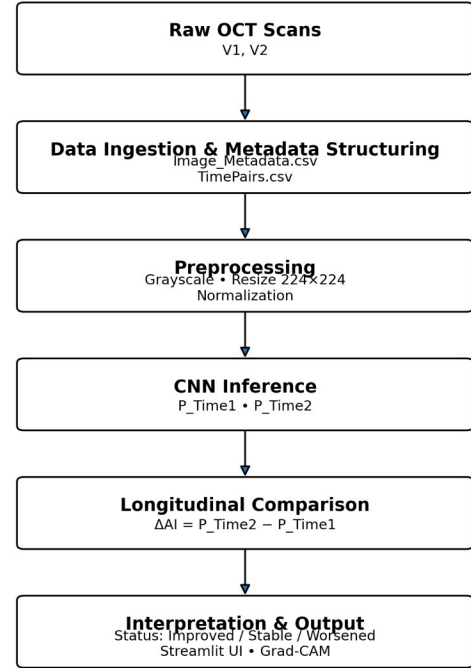


Fig. 1. System architecture and longitudinal inference workflow.

1. **Data Ingestion & Pairing:** Image_Metadata.csv tracks attributes (Patient_ID, Visit, Class, File Path). Scans are automatically grouped chronologically via TimePairs.csv (V1 → V2, V2 → V3).

$$V_fluid = \sum (Area_k \times \Delta z)$$

Central Retinal Thickness (CRT, μm) is measured automatically as the perpendicular distance between the Internal Limiting Membrane (ILM) and Retinal Pigment Epithelium (RPE).

2. **Preprocessing Pipeline:** Scans are converted to grayscale, resized to 224×224 pixels, intensity-normalized, and transformed into PyTorch tensors.

3. **CNN Inference & Differential Engine:** Computes P_Time1 and P_Time2 , outputs ΔAI , and applies thresholding logic.

4. **Interactive Dashboard:** Results are rendered in a side-by-side Streamlit web interface displaying clinical IDs, raw metrics, and classification badges.

5. **Quantitative Segmentation Roadmap:** Implementation of a semantic segmentation pipeline mapping B-scan pixels to background (0), IRF (1), SRF (2), or PED (3). Volumetric fluid integration across N sequential slices yields metric fluid loads in cubic millimeters (mm^3):

V. EVALUATION & RESULTS

The longitudinal framework was tested on a structured dataset of 102 OCT scans deployed on Azure ML (Standard_E4ds_v4 CPU VM environment).

Classification Baseline Performance: The single-image PyTorch CNN achieved an accuracy of 98.04% post-data augmentation.

Longitudinal Pair Case Study: For a representative patient pair (V1 $\rightarrow$ V2), baseline probability $P_{Time1} = 0.91$ and follow-up probability $P_{Time2} = 0.76$ yielded $\Delta AI = -0.15$. With threshold $T = 0.05$, the system categorized the longitudinal trajectory as Improved. The capabilities of the proposed system are compared with vendor OCT functionality in Table II.

TABLE II
SYSTEM CAPABILITY COMPARISON

Capability	Vendor OCT	Proposed AI System
Fluid Detection	Basic automated	Voxel-level 3D segmentation
Fluid Volume (mm ³)	Limited / locked	Automated volumetric load
Longitudinal % Delta	Not computed	Volume % change calculation
Layer Segmentation	Basic tracing	Multi-layer 3D segmentation
Cross-Device Standard	Hardware-locked	Vendor-agnostic AI model
Explainability (XAI)	Limited overlay	Grad-CAM visual heatmaps

VI. CONCLUSION

In this paper, we presented a cloud-native deep learning framework for longitudinal monitoring of neovascular age-related macular degeneration (nAMD) progression using sequential Optical Coherence Tomography (OCT) scans. Building upon an initial binary classification baseline that achieved 98.04% accuracy post-data augmentation, our system expands from cross-sectional image evaluation to an intra-patient comparative workflow. By computing a probabilistic differential indicator ($\Delta AI = P_{Time2} - P_{Time1}$) across consecutive clinical visits and applying threshold-based stratification, the system categorizes progression into Improved, Stable, or Worsened states. Deployed on Azure Machine Learning with interactive Streamlit visualization and Grad-CAM explainability, the platform offers clinicians objective, scalable decision support that reduces inter-observer variability in anti-VEGF treatment evaluation.

Future work will focus on advancing from probabilistic abnormality scores to full volumetric biomarker quantification. By training 3D U-Net and V-Net architectures on the RETOUCH benchmark dataset, we aim to implement automated pixel- and voxel-level semantic segmentation to directly measure intraretinal

fluid (IRF), subretinal fluid (SRF), pigment epithelial detachment (PED) volumes in cubic millimeters (mm³), and central retinal thickness (CRT) in micrometers (μ m).

REFERENCES

- [1] U. Schmidt-Erfurth et al., "AI-based monitoring of retinal fluid in disease activity and under therapy," *Progress in Retinal and Eye Research*, vol. 86, p. 100972, 2022.
- [2] A. Ucar, M. Karakose, and N. Kırımca, "Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends," *Appl. Sci.*, vol. 14, no. 2, p. 898, 2024.
- [3] O. Leingang, S. Riedl, J. Mai et al., "Automated deep learning-based AMD detection and staging in real-world OCT datasets (PINNACLE study report 5)," *Sci. Rep.*, vol. 13, p. 19545, 2023.
- [4] V. Mares, M. B. Nehemy, H. Bogunovic et al., "AI-based support for optical coherence tomography in age-related macular degeneration," *Int. J. Retin. Vit.*, vol. 10, p. 31, 2024.
- [5] M. H. Akpinar et al., "Artificial intelligence in retinal screening using OCT images: A review of the last decade (2013–2023)," *Comput. Methods Programs Biomed.*, vol. 254, p. 108253, 2024.
- [6] D. S. Kermany et al., "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.
- [7] R. L. W. Hanson et al., "Optical coherence tomography imaging biomarkers associated with neovascular age-related macular degeneration: a systematic review," *Eye*, vol. 37, pp. 2438–2453, 2023.
- [8] E. Crincoli et al., "Updates on novel and traditional OCT and OCTA biomarkers in nAMD," *Eye*, vol. 39, pp. 1662–1672, 2025.
- [9] C. Metrangolo et al., "OCT Biomarkers in Neovascular Age-Related Macular Degeneration: A Narrative Review," *J. Ophthalmol.*, vol. 2021, Art. ID 9994098, 2021.
- [10] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015, pp. 234–241.
- [11] K. Yildirim et al., "U-Net-Based Segmentation of Current Imaging Biomarkers in OCT-Scans of Patients with Age Related Macular Degeneration," *Stud. Health Technol. Inform.*, vol. 302, pp. 947–951, 2023.
- [12] R. Holland et al., "Deep Learning–Based Clustering of OCT Images for Biomarker Discovery in Age-Related Macular Degeneration," *Ophthalmol. Sci.*, vol. 4, no. 6, p. 100543, 2024.
- [13] T. Schlegl et al., "Fully Automated Detection and Quantification of Macular Fluid in OCT Using Deep Learning," *Ophthalmology*, vol. 125, no. 4, pp. 549–558, 2018.
- [14] F. Milletari, N. Navab, and S. A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *3DV*, 2016.
- [15] G. Moraes et al., "Quantitative Analysis of OCT for Neovascular Age-Related Macular Degeneration Using Deep Learning," *Ophthalmology*, vol. 128, no. 5, pp. 693–705, 2021.
- [16] M. Pawloff et al., "Performance of retinal fluid monitoring in OCT imaging by automated deep learning versus human expert grading in neovascular AMD," *Eye*, vol. 37, pp. 3793–3800, 2023.
- [17] RETOUCH Challenge, "Retinal OCT Fluid Detection and Segmentation Benchmark Dataset."
- [18] N. Ndipenoch et al., "nnUNet RASPP for retinal OCT fluid detection, segmentation and generalisation over variations of data sources," *arXiv:2302.13195*, 2023.

Agentic AI Across the STLC: Gains, Costs, and Limits

Nino Tsitsriashvili

Georgian Technical University, Tbilisi, Georgia

Abstract—Large language models generate test cases quickly, but the tests are shallow and cannot judge their own correctness. This paper reports a case study of an agentic QA toolchain on a nine-person payments team, organised by the six phases of the Software Testing Life Cycle. QA introduced two plugins for an LLM coding agent: a checklist generator grounded in the code diff, and a live validation agent that issues verdicts only from evidence. Repository mining six months before and 2.65 months after adoption shows median time from first commit to main fell from 4.82 to 2.97 days at unchanged release cadence, and the share of backend fix commits including a test rose from 21.5% to 52.4%, almost entirely through the QA-owned API suite. Post-release fixes were too few to show a direction. The agent removed enumeration from the tester's work and left judgement with the tester.

Index Terms—agentic AI, case study, software testing life cycle, test automation.

I. INTRODUCTION

Software testing is dominated by activities that transform one artifact into another: requirements into test cases, test cases into scripts, failures into bug reports. Large language models (LLMs) perform such transformations fluently, and a growing body of work applies them to test generation [1], [3]. The same body of work reports a consistent weakness: generated tests reach reasonable coverage but are frequently shallow, sometimes wrong, and unable to judge whether the observed behaviour is correct [2], [4]. A generator cannot be its own oracle.

Agentic systems differ from generators. In this paper a system is agentic if the model reads the repository and executes tools on it rather than only answering a prompt; the more capable ones also inspect the running system and act on it. A checklist derived by an agent from the code diff reflects what actually changed rather than what a tester described. A verdict issued by an agent that read the database after the action reflects what actually happened. Whether this difference matters in a real testing process, across all of its phases rather than in a benchmark, few studies address it [5], [6].

This paper reports a case study that addresses it. The setting is a small team building a subscription and payments product, a domain in which a wrong pass has direct financial consequences and in which the author works as one of the QA engineers. In the summer of 2026 QA introduced two plugins for an LLM coding agent, and the team's repositories were mined for the periods before and after. The Software Testing

Life Cycle (STLC), with its six phases of requirement analysis, test planning, test case development, environment setup, execution and closure, is the analytical frame. Three research questions follow the title: what was gained (RQ1), what it cost (RQ2), and where the tool reached its limit (RQ3).

II. RELATED WORK

Wang et al. survey 102 studies of LLMs in software testing and organise them by task rather than by process phase [1]. Schäfer et al. showed that an LLM re-prompted with failing output reaches about 70% statement coverage on JavaScript packages [3]; on real-world functions, however, generated tests average 41% accuracy, 30% branch coverage and 40% mutation score [4]. Barr et al. observe that where automated oracles are unavailable, the human remains the final source of expected behaviour [2]; the validation agent in this study is designed on exactly that premise. Wang et al. automate a complete test process with LLMs in an automotive setting [5], and an interview study reports how practitioners use LLMs in testing [6]. A randomised study found that AI assistance lowers comprehension of work that is delegated to it [7]. The present study differs in that the tool is agentic, the process is observed in production, and the unit of analysis is a whole delivery pipeline.

III. CASE AND TOOLCHAIN

The product has a Go backend, a TypeScript web application and an admin panel. The team comprises five developers, one product manager and three QA engineers. Releases flow from feature branches to a staging branch to the main branch; there are no release tags. The author's work is manual: analysing tickets, planning and preparing tests, validating features on branch, staging and production. It leaves almost no trace in version control (one fix commit in six months), so the study measures the delivery system the tester's tooling acts on, not the tester's output.

Developers had the coding agent Claude Code from August 2025. QA planning used an Excel checklist template. A Go API test suite (`api_tests`) was introduced in March 2026 and runs on pull requests to staging; developers add API tests for their features, QA reviews coverage and specifies the cases for any gap, and developers implement them. From March the author also ran a programme to cover existing features that had shipped without API tests. That programme spans both study windows and is treated as a confound.

The intervention is two Claude Code plugins built by the team. The checklist generator (30 June 2026) is a planning tool:

N. Tsitsriashvili is with the Computer Information Technologies programme, Georgian Technical University, Tbilisi, Georgia.

it reads the ticket with all comments, the diff across the three repositories, the entity specification and the existing API tests, and produces a per-ticket checklist in the team's template together with the endpoints in the change that have no test. It checks nothing against a running system. The validation agent (7 September 2026) is an execution tool that validates a feature live, one check at a time. It arrived too late in the window to be evaluated and is assessed as a design.

IV. SYSTEM ARCHITECTURE

Fig. 1 shows the validation agent's check loop. For each check the agent reads a baseline from the database, records the expected result before anything happens, gives the tester a single UI instruction, gathers evidence from database rows, pod logs and CLI output, and issues a verdict of PASS, NOT-EXERCISED, INCONCLUSIVE or BLOCKED bound to an evidence identifier in a hash-chained journal. A verdict requires a marker in real evidence; code inspection may form a hypothesis but never a pass.

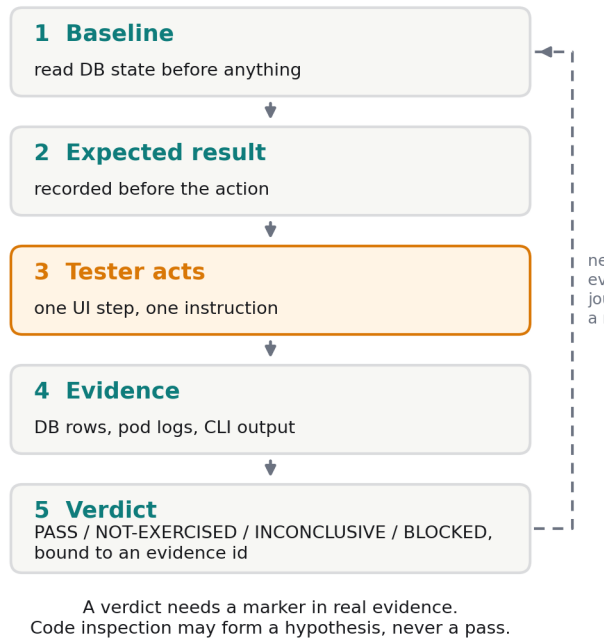


Fig. 1. The validation agent's check loop.

Before a run the agent verifies the deployed build per repository and stops on mismatch. It seeds test data through gated database writes that record the prior value and a rollback, or by running the backend CLI in the pod; production is read-only in code. A critic sub-agent reviews the plan before the run and the verdicts during it. Bug tickets, API-test gaps and specification updates are drafted from templates; nothing is posted without explicit human approval. Under time pressure it posts a partial result rather than omit a step. These refusals are a specification of the division of labour: the agent gathers evidence, the human states the expectation and approves the outcome.

V. EVALUATION

Design. An embedded single-case study following Runeson

and Höst [8]. BEFORE spans 30 December 2025 to 29 June 2026 (5.98 months); AFTER spans 30 June to 18 September 2026 (2.65 months). Three data sources were combined: read-only mining of the three repositories, analysis of the plugins' artifacts, and the author's retrospective account. Every finding is graded measured (repository data), documented (an artifact) or reported (the author). Cycle time is the interval from a ticket's first commit to its first appearance on the main branch, excluding release-train merges. Fix commits are identified by message keywords and split by whether their diff touches the QA-owned `api_tests` suite or only developer unit tests. Post-release fixes are fix commits referencing a ticket that reached main in an earlier month.

Gains (RQ1). Median cycle time fell from 4.82 to 2.97 days pooled ($n = 592, 264$) and from 4.80 to 2.06 days in the backend ($n = 296, 154$), while release merges rose from 22.1 to 24.6 per month and tickets shipped were 118 and 107 per month. The reduction is in waiting, not in volume (Fig. 2a). It is confined to the backend, the only repository with the API suite; the web application, which has none, did not improve (4.09 to 4.75 days). The share of backend fix commits that include a test rose from 21.5% to 52.4%; commits touching `api_tests` rose from 7.3% to 37.3% while unit-test-only commits were flat at 14.3% to 15.1% (Fig. 2b). All `api_tests`-bearing fix commits in the AFTER period were authored by developers. The suite grew from 589 to 1,005 cases; part of that growth belongs to the coverage programme and the share attributable to plugin proposals was not traced. The web application acquired 204 unit tests and a deploy gate, added three days after the intervention.

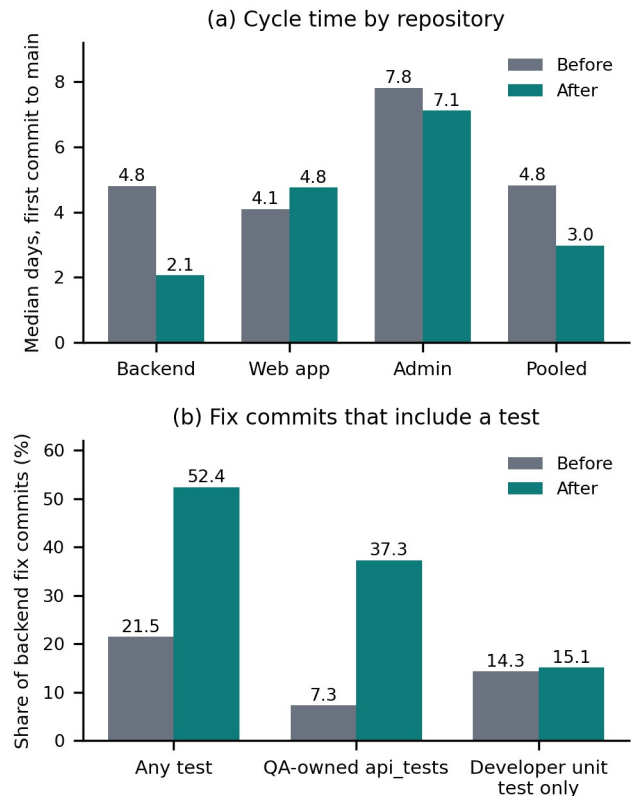


Fig. 2. (a) Median days from first commit to main by repository. (b) Share of backend fix commits that include a test, by test type.

Costs (RQ2). Reported: at analysis time the tester knows the feature less than before, because the agent supplies the affected areas and the detailed model now forms during execution; an area the agent misses is caught later or not at all. Reported: reviewing generated checklists, proposals and drafts replaced authoring as the bottleneck. Measured: 37 commits of plugin upkeep in eleven weeks, made by the team lead and approved by the tech lead, QA lead and product manager. Fact: the code is read by a hosted model, with no restriction or oversight during the study.

Limits (RQ3). Found: the agent cannot know what is absent from the diff, such as priorities across tickets, release collisions and payment invariants (no double charge, no refund left inconsistent); it does not decide when suites run, since all three CI trigger policies were added by people on recorded dates; and escaped defects did not measurably change, with 20 post-release fixes before and 11 after, too few to read a direction. Chosen: no pass from code, no writes to production, nothing posted without approval, no skipped step under time pressure. A consequence is that a wrong expectation is confirmed rather than caught; the tester remains the oracle.

Threats. The design is before-and-after, not controlled. Developers' use of the coding agent grew through 2026 and faster development alone could shorten cycle time; the web application result argues against this being the whole explanation but does not exclude it. The API suite was already growing and the coverage programme ran through both windows. Releases and fixes are proxied from commit messages and first-parent history. The AFTER window is short. The analysis was run by Claude Code, the product under study; the metric definitions are simple enough to re-run by hand. One team, one tester's account: the findings describe this case.

VI. CONCLUSION

An agentic QA toolchain changed every phase of the testing life cycle on a payments team, but unevenly and not for free. Enumeration became cheap: scope from the diff, checklists per ticket, API cases proposed and implemented. Tickets stopped waiting: cycle time fell by more than a third at unchanged cadence, and testing moved into developers' pull requests through the suite QA owns. The costs were later feature understanding and review replacing authoring. The limits are partly found, in what no diff contains and no agent decides, and partly chosen, in a validation agent built so that it cannot declare a feature correct on its own authority. Escaped defects did not measurably change; a longer window is needed. The tester's role moved from executing checks to specifying them. Future work will evaluate the validation agent from its run journals, in particular how often INCONCLUSIVE and NOT-EXERCISED verdicts prevent a false pass.

REFERENCES

[1] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software Testing With Large Language Models: Survey, Landscape, and Vision," *IEEE Trans. Softw. Eng.*, vol. 50, no. 4, pp. 911–936, 2024.

[2] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The Oracle Problem in Software Testing: A Survey," *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 507–525, 2015.

[3] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation," *IEEE Trans. Softw. Eng.*, vol. 50, no. 1, pp. 85–105, 2024.

[4] D. Huang, J. M. Zhang, M. Harman, Q. Zhang, M. Du, and S.-K. Ng, "Benchmarking LLMs for Unit Test Generation from Real-World Functions," *ACM Trans. Softw. Eng. Methodol.*, art. 3805043, 2026, in press.

[5] S. Wang, Y. Yu, R. Feldt, and D. Parthasarathy, "Automating a Complete Software Test Process Using LLMs: An Automotive Case Study," in *Proc. ICSE*, 2025, pp. 373–384.

[6] M. D. Santana, C. Magalhaes, and R. de Souza Santos, "Software Testing with Large Language Models: An Interview Study with Practitioners," in *Proc. IEEE/ACM AIware*, 2025, pp. 96–104.

[7] Anthropic, "How AI Assistance Impacts the Formation of Coding Skills," research report, 2026.

[8] P. Runeson and M. Höst, "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical Softw. Eng.*, vol. 14, no. 2, pp. 131–164, 2009.

Integration of an Online Payment System into a Web-Based E-Commerce Application

Nataliia Tsurkan, Thanh-Loan Katharina Pham

Abstract— *Modern e-commerce applications combine client-side interfaces, backend services, databases, and external payment providers. This paper presents the design and implementation of a web-based e-commerce application developed as a gift shop, with a focus on the integration of online payment processing. The system uses Vue.js as a Single-Page Application (SPA) for product browsing, cart management, and administrative functions. A PHP-based backend and MariaDB database handle product and order data. Payment processing is implemented using the Stripe Checkout API in test mode. The paper focuses on the transformation of local cart data into Stripe-compatible payment payloads and the synchronization of order and inventory data after payment. The implementation was evaluated through internal functional testing to assess the implemented workflow and identify limitations of the current approach.*

Index Terms— *E-commerce, Online Payment Processing, Payment Gateway Integration, Stripe Checkout*

I. INTRODUCTION

Modern e-commerce applications combine client-side interfaces, backend services, databases, and external payment providers. Single-Page Applications (SPAs) are commonly used to provide dynamic user interfaces, while backend APIs handle data persistence and application logic. Developing such an application requires more than implementing the product interface. The system must coordinate shopping cart data, product and order information, and payment processing across multiple application components. In particular, integrating an external payment provider requires the application to transform its local data into the format expected by the provider and to process the result of a completed payment.

Implementing payment processing directly within an application would require the system to handle additional security and compliance requirements. Using a third-party payment provider can reduce the amount of payment-related data handled by the application. However, the integration still requires a consistent data flow between the client application, backend, database, and external payment service. In addition, the application must ensure that completed payments are correctly associated with orders and that corresponding inventory data is updated. These requirements make the

integration of payment processing an important part of the overall e-commerce system.

In this paper, we present the design and implementation of a web-based e-commerce application developed as a gift shop, with a particular focus on the integration of online payment processing. The system combines a Vue.js frontend with a PHP backend and a MariaDB database. Stripe Checkout is used as the external payment provider. The paper describes how cart data is transferred from the client to the backend, transformed into the data structure required by Stripe, and used to create a Checkout session. It further describes how completed payments are processed by the application and how order and inventory data are updated. Finally, the implementation is evaluated through internal functional testing, and limitations of the current payment and synchronization workflow are discussed.

II. RELATED WORK

Online payment systems can be integrated into e-commerce applications using different approaches. One possibility is to process payment-related data directly within the merchant's application. This gives the application control over the payment process, but also introduces additional security requirements. The PCI Data Security Standard (PCI DSS) defines technical and operational requirements for environments that handle or can affect payment card data [1].

Another approach is to use payment components provided by an external payment service. For example, Stripe Elements allows payment forms to be embedded directly into a web application. This approach keeps the payment interface within the merchant's website while using Stripe for payment-related functionality [2].

A different approach is a hosted checkout page. Stripe Checkout provides a prebuilt payment page to which customers can be redirected from the merchant's website. The application creates a Checkout Session and transfers the required information about the purchase to Stripe [3].

When integrating a third-party payment provider, the communication between the application and the payment service must also be handled carefully. OWASP identifies aspects such as order initialization, payment verification, and the handling of payment results as important parts of third-party payment integration [4].

III. PROPOSED METHOD

The presented application uses the hosted checkout approach with Stripe Checkout. The focus is not on implementing payment processing itself, but on connecting the local shopping cart and backend with Stripe and processing the resulting payment within the application's order workflow. The implemented approach consists of four main components: authentication before checkout, external address collection, server-side transformation of cart data, and post-payment order and inventory processing.

(1) The checkout process requires the user to be logged in before a payment session can be created. (2) The required shipping information is provided through Stripe Checkout rather than through additional address fields in the application. This keeps the checkout form within the hosted payment workflow and reduces the amount of checkout information that has to be managed by the application itself. (3) The shopping cart is sent from the Vue.js application to a PHP backend endpoint. The backend transforms the received cart data into the structure required for a Stripe Checkout Session, including product information, prices, and quantities. This creates the connection between the local product and cart data and the Stripe payment request. (4) After a successful test payment, the application processes the resulting order and stores the order information in the MariaDB database. The corresponding product stock is also updated so that the local product data reflects the completed order.

IV. SYSTEM ARCHITECTURE

The payment workflow connects the Vue.js frontend, the PHP backend, Stripe Checkout, and the MariaDB database. The main communication and data flow between these components is illustrated in Fig. 1.

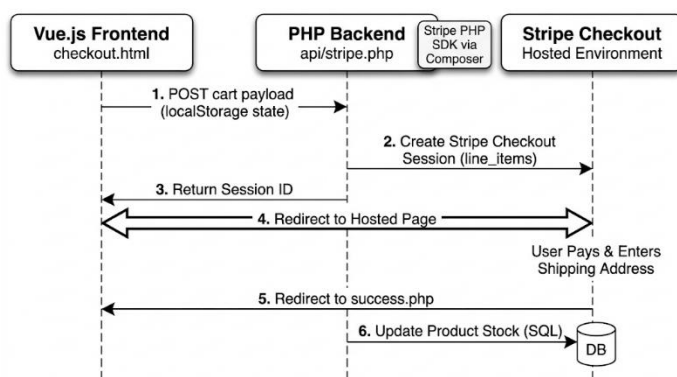


Fig. 1. Payment and order processing architecture.

A. Client-Side Checkout Initiation

When the user starts the checkout process, the Vue.js application first checks whether an active user session exists. Unauthenticated users are redirected to the login page. For authenticated users, the cart stored in the browser's localStorage is sent to the PHP backend using an HTTP POST request. The backend response contains the Stripe Checkout Session ID, which is then used to redirect the user to the Stripe-hosted checkout page.

```

fetch('api/stripe.php', {
  method: "POST",
  headers: {"Content-Type": "application/json"},
  body: JSON.stringify(this.cart)
})
.then(res => res.json())
.then(data => {
  const stripe = Stripe("pk_test_...");
  stripe.redirectToCheckout({ sessionId: data.id });
});
  
```

B. Backend Payload Transformation

The PHP endpoint api/stripe.php receives the cart data and creates a Stripe Checkout Session. The local cart structure is transformed into Stripe's line_items format. Product names, prices, and quantities are mapped to the corresponding fields required by Stripe. The session also defines the success URL and enables shipping address collection for the supported countries:

```

$line_items = [];

foreach ($cart as $item) {
    $line_items[] = [
        'price_data' => [
            'currency' => 'eur',
            'product_data' => ['name' =>
$item['title']],
            'unit_amount' => $item['price_cents'],
        ],
        'quantity' => $item['quantity'],
    ];
}

$session = \Stripe\Checkout\Session::create([
    'line_items' => $line_items,
    'mode' => 'payment',
    'shipping_address_collection' =>
        ['allowed_countries' => ['DE', 'AT', 'CH']],
    'success_url' =>
'https://.../success.php?session_id={CHECKOUT_SESSION_ID}'
]);
  
```

C. Post-Payment Processing

After the Stripe Checkout process, the customer is redirected to success.php. The application retrieves the corresponding Stripe session and processes the order data. The order and order-item data are stored in MariaDB, and the corresponding product stock is updated based on the purchased quantities.

V. EVALUATION

As no formal performance benchmark or external user study was conducted, the evaluation is based on internal functional testing of the implemented payment workflow. The main functions tested were user authentication, cart data transmission, Stripe Checkout session creation, and post-payment order processing. During testing, unauthenticated users were redirected to the login page, while authenticated users could proceed to the checkout process. Cart data stored in `localStorage` was successfully sent to the PHP backend and transformed into the `line_items` structure required by Stripe. The payment workflow was tested using Stripe's test mode. The Stripe Checkout page was opened successfully, and shipping address information could be entered through the hosted checkout page. After the test payment, the customer was redirected to `success.php`. The application then stored the order and order-item data in MariaDB and updated the corresponding product stock.

A limitation of the current implementation is the use of the browser redirect to `success.php` for post-payment processing. If the browser is closed immediately after the payment, the order and inventory updates may not be performed even though the payment has been completed. A server-side webhook could be used in a future version to process the payment result independently of the browser redirect.

VI. CONCLUSION

This paper presented the integration of an online payment system into a web-based e-commerce application. The implementation combines a Vue.js Single-Page Application with a PHP backend, a MariaDB database, and Stripe Checkout. The main focus was the integration of an external payment provider into the existing application and the connection between the shopping cart, payment process, and local order management.

The project shows how a hosted checkout solution can be integrated into a web shop while keeping the payment interface separate from the main application. The implemented architecture provides a clear data flow between the client, backend, external payment service, and database. Further development could extend this approach by introducing server-side webhooks for handling payment events independently of the client-side checkout process.

REFERENCES

- [1] PCI Security Standards Council, *Payment Card Industry Data Security Standard (PCI DSS), Version 4.0*, March 2022.
- [2] Stripe, *Payment Element Documentation*, Stripe Documentation.
- [3] Stripe, *Checkout Documentation*, Stripe Documentation.
- [4] OWASP Foundation, *Secure Integration of Third-Party Payment Gateways Cheat Sheet*, OWASP Cheat Sheet Series.

The Decline of Linguistic Diversity in Recursively Trained LLMs

Jonas K. Zimmermann

As generative AI becomes more widespread, an increasing share of online text is itself AI-generated, raising the possibility that future language models will be trained recursively on synthetic data produced by earlier models. This paper examines how such recursive training affects linguistic diversity in large language models. Drawing on recent research on model collapse, it discusses how repeated training on model-generated data can cause models to progressively lose information about the original data distribution, and how this process manifests in language models as a measurable decline in lexical, syntactic, and semantic diversity.

The paper further considers whether these findings support the claim that AI-generated language is becoming less human-like, arguing that the evidence points to specific training procedures rather than an inevitable trend. Finally, it discusses approaches — such as detecting synthetic text and carefully balancing synthetic with human-generated data — that may help mitigate model collapse and preserve linguistic diversity in future language models.

Index Terms — language model training, linguistic diversity, model collapse, synthetic data

I. INTRODUCTION

Large language models (LLMs) have fundamentally changed the way artificial intelligence processes and generates human language. These models are trained on enormous quantities of textual data, much of which has traditionally been produced by humans. However, as generative AI becomes increasingly widespread, a growing proportion of online content may itself be generated by AI. This creates a potential feedback loop: AI-generated text can become part of the training data for future models, which then produce new synthetic text for subsequent training. This process is known as recursive training.

Synthetic data offers several advantages, particularly the ability to generate large quantities of training material. However, research suggests that repeatedly training models on data produced by previous models can have negative consequences. Shumailov et al. [1] describe this phenomenon as model collapse, in which models progressively lose information about the original data distribution. For language models, Guo et al. [2] found that recursive training can reduce lexical, syntactic, and semantic diversity.

II. MODEL COLLAPSE AND RECURSIVE TRAINING

Model collapse describes a degenerative process that can occur when generative models are repeatedly trained on data generated by earlier generations. Shumailov et al. [1] found that recursively generated training data can cause models to progressively lose information about the original data distribution, particularly from the tails of the distribution: in natural language, some words, sentence structures, and topics occur frequently, while others are relatively rare, and it is the latter that tend to be lost first. Over successive generations, the model can therefore become increasingly concentrated around common patterns.

This process can be illustrated by a simple cycle: a model trained on human-written text generates synthetic text, which is then used to train another model. If this process is repeated and synthetic data increasingly replaces the original human data, small changes in the data distribution can accumulate, so that the resulting model increasingly learns from the outputs of previous models rather than directly from human-generated language [1].

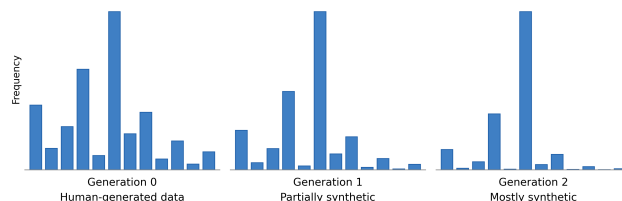


Fig 1: Distributional narrowing across recursive training generations

III. THE DECLINE OF LINGUISTIC DIVERSITY

Guo et al. [2] examined this problem specifically in the context of language. Their study simulated several generations of language models that were successively trained using synthetic text produced by earlier models. The researchers measured lexical diversity (the variety of words and expressions), syntactic diversity (the variety of sentence structures), and semantic diversity (variation in meaning and content). One of the metrics used to quantify lexical diversity is the type-token ratio (TTR), the proportion of unique words to total words in a text; a lower TTR indicates a narrower vocabulary. Their results showed a decline in all three dimensions across successive generations, particularly in tasks requiring creative language generation. This decline is

illustrated by the TTR values reported for the news summarization task, which fall from 7.36 for human-written text to 3.66 after six recursive training generations [2].

Importantly, Guo et al. [2] note that recursive training can affect linguistic diversity even when a model's outputs remain grammatically correct and meaningful: the generated text may become increasingly predictable and less varied without exhibiting obvious errors. If unusual linguistic patterns disappear while highly probable patterns are repeatedly reinforced, the resulting language may contain less of the variation characteristic of human communication. Human language is naturally diverse: individuals use different vocabulary, sentence structures, cultural references, and writing styles. A model that increasingly concentrates on the most probable linguistic patterns could therefore produce language that is technically accurate but less diverse.

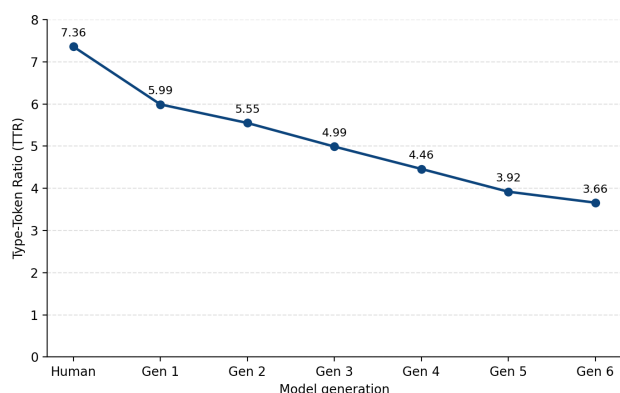


Fig 2: Lexical diversity decline across recursive training generations

IV. WHY LANGUAGE COULD BECOME LESS HUMAN-LIKE

The findings of Shumailov et al. [1] and Guo et al. [2] help explain how recursive training could lead to increasingly standardized language generation: if rare patterns are progressively removed from the training distribution, models have fewer opportunities to learn them, while frequently occurring patterns are repeatedly reinforced.

In a brief commentary on Shumailov et al.'s findings, Smith et al. [3] discuss this problem in relation to human language transmission. They suggest that studying how language is transmitted between humans can provide insights into how linguistic information may be preserved in artificial systems, and that repeated transmission through generative models can favor highly probable linguistic patterns, consequently altering the distribution of language over generations.

However, the term less human-like should be used carefully. Existing research does not demonstrate that every new generation of AI models automatically becomes less human-like. Instead, the evidence indicates that particular recursive training procedures can reduce linguistic diversity and cause models to reproduce patterns from previous generations more strongly.

V. APPROACHES TO PRESERVING LINGUISTIC DIVERSITY

The potential problems associated with synthetic data do not mean that AI-generated data must be completely excluded from

training. Synthetic data can be useful for increasing the amount of training material and developing models for specific tasks; the important question is how synthetic and human-generated data are combined.

Drayson et al. [4] investigate whether machine-generated text can be identified and treated differently during training. Their experiments with GPT-2 and SmolLM2 suggest that detecting synthetic text and assigning greater weight to human-generated material can help prevent model collapse under the conditions examined in their study.

Kazdan et al. [5] similarly show that the consequences of synthetic data depend on the training procedure: while replacing real data with successive generations of purely synthetic data can lead to model collapse, alternative approaches that combine synthetic and real data can remain stable. Thus, synthetic data itself is not necessarily the problem; its relationship with original human-generated data is crucial.

VI. CONCLUSION

The increasing use of AI-generated text as training data creates a potential feedback loop in the development of large language models. Research on model collapse suggests that repeatedly training models on data produced by previous generations can cause them to lose information about the original data distribution [1]. In language models, this process can result in a measurable reduction in lexical, syntactic, and semantic diversity [2].

This decline in linguistic diversity provides a possible explanation for why AI-generated language could become increasingly standardized and less varied. However, current research does not support the claim that AI language will inevitably become less human-like. The effects depend substantially on how synthetic data is generated and incorporated into subsequent training [4], [5].

As AI-generated content becomes increasingly common online, preserving access to diverse human-written text may therefore become an important challenge in the future development of large language models.

REFERENCES

- [1] I. Shumailov, Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson, and Y. Gal, "AI models collapse when trained on recursively generated data," *Nature*, vol. 631, pp. 755–759, 2024.
- [2] Y. Guo, G. Shang, M. Vazirgiannis, and C. Clavel, "The curious decline of linguistic diversity: Training language models on synthetic text," in *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024, pp. 3589–3604.
- [3] K. Smith, S. Kirby, S. Guo, and T. L. Griffiths, "AI model collapse might be prevented by studying human language transmission," *Nature*, vol. 633, p. 525, 2024.
- [4] G. Drayson, E. Yilmaz, and V. Lampos, "Machine-generated text detection prevents language model collapse," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 29657–29673.
- [5] J. Kazdan, R. Schaeffer, A. Dey, M. Gerstgrasser, R. Rafailov, D. L. Donoho, and S. Koyejo, "Collapse or thrive: Perils and promises of synthetic data in a self-generating world," in *Proceedings of the 42nd International Conference on Machine Learning*, vol. 267, 2025, pp. 29469–29494.