Research Project in Computer Systems Engineering

Final Report

Physical Coding Blocks for Robotics and Software Education

[redacted]

Co-worker: [redacted]

Supervisor: [redacted]

Co-Supervisor [redacted]

[redacted]

UNIVERSITY OF AUCKLAND
Waipapa Taumata Rau
NEW ZEALAND

ENGINEERING
DEPARTMENT OF ELECTRICAL, COMPUTER, AND SOFTWARE ENGINEERING

ECSE-2025

Physical Coding Blocks for Robotics and Software Education

[redacted]

ABSTRACT

This research project extends a 2024 tangible user interface system designed to introduce young children to programming and robotics through physical interaction. The previous prototype successfully demonstrated fundamental functionality but exhibited several limitations, including fragile mechanical design, macOS dependency, and minimal user feedback. This continuation project aimed to enhance the system's robustness, cross-platform compatibility, feedback mechanisms, and scalability to support wider classroom adoption.

Key developments include migrating the firmware from C to MicroPython for improved compatibility, redesigning the enclosure with a 3D-printed backplate, aluminium standoffs, and N45 neodymium magnets for structural stability and secure connections; and integrating LED-based visual feedback for real-time operation. A new Next.js web portal was developed to monitor connected Raspberry Pi Pico W devices and mBot2 robots, enabling remote observation of command queues.

Performance evaluation demonstrated a 96% energy efficiency, stable memory usage, and an average Pico W to robot latency of 2.23 seconds from block placement to robot action. While an in-depth user study was postponed due to pending ethics approval, pilot sessions demonstrated strong engagement, collaboration, and creative problem-solving among participants.

These findings advance the development of tangible programming systems by providing a durable, accessible, and extensible platform for robotics education. The outcomes establish a foundation for affordable, scalable, and classroom-ready tangible programming systems that further promote creativity, collaboration, and computational thinking among young learners.

Table of Contents

List of Tables

List of Figures

Glossary of Terms

Term	Definition
Arduino	An open-source electronics platform used for prototyping embedded systems.
CMake	A cross-platform build automation tool that manages the compilation process of software projects.
GET	Hypertext Transfer Protocol (HTTP) “GET” request – retrieves data from a server
Infrared Technology	A communication method using infrared light for proximity sensing or short-range data transmission.
MicroPython	A lightweight implementation of the Python language for microcontrollers and embedded systems.
OS-agnostic	A system or software that operates independently of a specific operating system.
Pico W	Raspberry Pi Pico W, a low-cost microcontroller board with Wi-Fi capability used for communication and block detection.

PLA	A biodegradable thermoplastic used in 3D printing for prototype enclosures
POST	Hypertext Transfer Protocol (HTTP) “POST” request – sends data to a server
RFID	A short-range wireless communication technology used for identification via tapping or proximity
Tangible Programming	A programming approach using physical blocks or objects to represent code instructions, enhancing engagement and comprehension.

Abbreviations

API	Application Programming Interface
BOM	Bill of Materials
CT	Computational Thinking
DAP	Developmentally Appropriate Practices
GPIO	General-Purpose Input/Output
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
LED	Light-Emitting Diode
NZD	New Zealand Dollar
OS	Operating System
PCB	Printed Circuit Board
PLA	Polylactic Acid
RFID	Radio-Frequency Identification
SD	Standard Deviation
SSR	Server-Side Rendering
TUI	Tangible User Interface
UAHPEC	University of Auckland Human Participants Ethics Committee
UI	User Interface

1. Introduction

Programming and computational thinking (CT) have become essential components of early education, enabling children to develop problem-solving, logical reasoning, and creativity skills vital in a technology-driven world [1]. However, conventional screen-based programming environments such as Scratch often limit accessibility and engagement for younger learners, who may struggle to connect abstract digital logic with physical outcomes [2].

Tangible User Interfaces (TUIs) address this gap by allowing children to manipulate physical blocks that represent programming instructions; linking abstract programming concepts to concrete, sensory interaction. TUIs encourage experimentation, collaboration, and hands-on learning, making them particularly effective for young children [3]. Prior systems such as KIBO, and Tern have demonstrated how tangible programming can foster computational thinking and early engagement with robotics [4, 5].

Building on this pedagogical foundation, the 2024 TUI prototype developed at the University of Auckland [6, 7] introduced a tangible programming system that enabled children to assemble command blocks on a baseplate to control an mBot2 robot. The system successfully demonstrated basic programming constructs such as sequencing and repetition, achieving accurate block detection and reliable robot control. However, several limitations were identified: the 3D-printed baseplate and connector design were mechanically fragile, the firmware programming was limited to macOS environments, and user feedback was minimal. These issues hindered long-term educational deployment and scalability.

The 2025 continuation project addressed these shortcomings by enhancing the system's robustness, cross-platform compatibility, feedback mechanisms, and scalability. The main contributions of this work include:

- Firmware migration from C to MicroPython, enabling operating system independence and simplifying future feature expansion.
- Hardware redesign, incorporating a reinforced 3D-printed backplate, aluminium standoffs, and N45 neodymium magnets for improved alignment and mechanical durability.
- Integrated LED feedback system, providing immediate visual indicators for connection status, successful command execution, and error detection.
- Development of a Next.js-based web portal, offering real-time monitoring of connected Pico W devices and mBot2 robots, visualising command queues and connection states.

Together, these improvements aim to create a durable, accessible, and classroom-ready tangible programming platform that supports both independent exploration and collaborative problem-solving. Technical evaluation focused on energy efficiency, communication latency, and memory usage, while pilot studies provided preliminary insight into system usability and engagement.

Although a formal user study involving children aged 6–10 could not be conducted due to pending ethics approval (UAHPEC Project ID 29493), unstructured pilot sessions confirmed functional reliability and demonstrated promising engagement patterns. Participants displayed creativity, collaborative reasoning, and enthusiasm while constructing programs, supporting the system's potential as a learning tool.

This report presents the design, implementation, and evaluation of the improved tangible programming system.

2. Related Works

Tangible programming involves physical objects, like coding blocks, that children can use to create programs, offering a hands-on alternative to screen-based programming methods. This approach is constructive for young learners, as it supports their natural tendency for play and physical interaction, making the abstract concepts of sequencing and loops more intuitive and friendly. Studies, such as Pugnali et al. [8], suggest that tangible interfaces can positively influence children's competence in CT concepts, especially in early childhood education, by providing developmentally appropriate pathways to learning.

2.0.1 The Need for Early Computational Thinking Education

According to the Computer Science Teachers Association [9], CT is defined as a problem-solving process that includes formulating problems, logically organising & analysing data, representing the data, automating solutions, and identifying, analysing, and implementing possible solutions.

CT is a critical skill for everyone in today's technology-driven world in the 21st century. As stated by Wing [1], CT is a universally applicable skill set that develops a strong foundation for problem-solving, algorithmic thinking, and system design, which extends past the scope of computer science to everyday tasks.

Barr & Stephenson [10] emphasise its importance in K-12 education, describing CT as a structured approach to breaking down complex problems, identifying patterns, and creating step-by-step solutions. It is a vital tool for preparing students for future careers and digital literacy; primarily when introduced early, in developmentally appropriate ways that match children's cognitive and motor capacities.

2.0.2 Tangible Programming as a Solution

Using physical objects like coding blocks, tangible programming offers a screen-free, hands-on alternative to traditional programming environments, which often rely on a mouse, keyboard, or touchscreen. This is particularly relevant for children, who are fascinated by robots and eager to control them.

Research, such as Wing [1], emphasises CT as a critical skill, with tangible interfaces possibly lowering barriers for young learners by aligning with developmental theories of play and physical exploration [11], which encourages kids to be technology creators rather than merely consumers [4]. Pugnali et al. [8] studied the impact of user interfaces on young children's CT, comparing graphical (ScratchJr) and tangible (KIBO) systems. They found that tangible interfaces promote concept proficiency and positive interpersonal behaviours, making them a promising solution for delivering developmentally appropriate CT education.

2.1 Theoretical Foundations

Theoretical frameworks are necessary for understanding why tangible programming is effective, particularly for young learners. These theoretical foundations provide crucial insights into how young children learn and interact with physical objects, guiding the development of practical and tangible programming interfaces.

2.1.1 Constructionism

Papert's [12] constructionist theory suggests that learning occurs most effectively when children actively construct knowledge by creating tangible artefacts. Tangible programming aligns with this by allowing children to manipulate physical objects (e.g. blocks or robots) to build programs, fostering creativity and sharing. Bers et al. [4] and Bers [11] apply this theory to systems like KIBO, where children program robots with wooden blocks, embodying Papert's vision of "learning-by-making".

2.1.2 Developmentally Appropriate Practices

Developmentally Appropriate Practices (DAP) emphasise tailoring education to children's age, individuality, and socio-cultural context [4]. Tangible programming aligns with DAP by using play-based, concrete tools suitable for young learners' motor and cognitive stages. For example, KIBO [4] employs wooden blocks scanned by a robot, enabling young students to engage with CT through familiar, hands-on activities rather than abstract screen-based interfaces. This approach supports gradual skill development, ensuring accessibility and engagement without overwhelming young minds [13].

2.1.3 Cognitive Load Theory

Sweller's [14] cognitive load theory suggests that learning is optimised when minimising unnecessary cognitive load. Tangible interfaces reduce this load by providing concrete representations of abstract programming concepts (e.g. loops, conditionals), making them easier to grasp. This is evident in tools like Tern. [5] where physical pieces are directly mapped to code structures.

2.1.4 Social Learning Theory

Rooted in Bandura's work (1977, adapted from [15]) suggests that learning occurs through observation, imitation, and interaction. Tangible programming fosters this through collaborative activities, as demonstrated by Lee et al. [15], who found that kindergartners using robotics programs like KIBO increased social interaction and teamwork. Children observe peers' block arrangements and problem-solving strategies, modelling CT skills in a group setting. This social dynamic enhances engagement and reinforces learning, aligning with the collaborative potential of tangible tools [16].

2.1.5 Embodied Interaction

Based on embodied cognition principles [17], suggest that physical actions enhance cognitive processes by linking body and mind. Tangible programming leverages this by engaging motor skills to reinforce understanding, as seen in roBlocks [18]. Children physically assemble blocks to create sequences, embedding spatial and logical concepts in bodily actions. Antle [17] argues that this embodied approach makes abstract CT concepts more intuitive, particularly for young learners who rely on sensorimotor exploration.

2.1.6 Constructive Learning Theory

Piaget's [19] constructivism emphasises active knowledge construction through interaction with the environment. Tangible programming supports this by enabling hands-on exploration, as seen in roBlocks [18], where children experiment with electronic blocks to understand system behaviours.

These theories collectively explain how tangible programming bridges abstract coding with physical interaction, helping children engage with concepts and tools that would otherwise have a steep learning curve. They ensure that the tools are not only technically sound but also pedagogically effective, supporting the natural learning process of young children while promoting the development of CT skills.

2.2 Existing Research

Several studies have explored tangible programming in robotics education, particularly for early childhood. Bers et al. [4] investigated the TangibleK robotics program, pairing developmentally appropriate tools with a constructionist curriculum for kindergarteners, finding high concept mastery but varying with difficulty. Horn et al. [20] explored Tern, a tangible language, showing its efficacy in young learners. Bers et al. [4] demonstrated KIBO's success in teaching robotics and coding to preschoolers. Wang et al. [21] introduced T-Maze, an economic tangible tool for 5-9-year-olds, using computer vision for real-time feedback in maze games, with a user study of 7 children showing it fosters CT skills like abstraction and creativity. These studies indicate that tangible interfaces are promising for robotics education and potentially more engaging than screen-based methods.

Additional examples include McNerney [22], who developed Tangible Programming Bricks, a platform for creating microworlds for children to explore computational and scientific thinking through physical interfaces. Wyeth [23] designed Electronic Blocks for children aged 3-8, using sensor, action and logic blocks to program dynamic behaviours without symbolic notation. These systems illustrate diverse approaches to tangible programming, each tailored to different age groups and educational goals.

2.3 Tangible Programming Tools and Interfaces

The development of tangible programming tools has evolved significantly over the past two decades, with various approaches emerging to address the unique needs of early childhood learners. This section examines tangible programming systems' design principles and examples, focusing on their practical implementation.

2.3.1 Design Principles

Effective tangible systems prioritise intuitive interaction, immediate feedback, and scaffolding [17, 22]. Intuitive interaction ensures children can naturally manipulate objects, as in AlgoBlock [24]. Immediate feedback reinforces learning, like lights or sounds in roBlocks [18]. Scaffolding supports progression, as seen in KIBO's graduated complexity [4].

2.3.2 Physical vs. Graphical Interfaces

Tangible interfaces reduce cognitive load and provide concrete representations of abstract concepts, compared to graphical interfaces, by aligning physical actions with programming outcomes [20,25]. This enhances engagement and accessibility for young learners struggling with traditional screen-based interfaces.

2.3.3 Types of Tangible Programming Tools

- **Tangible Programming Languages**: AlgoBlock [22] uses connectable blocks to teach sequence and logic, while Tern [5] employs puzzle-like pieces for similar purposes.
- **Electronic Blocks**: roBlocks [18] enable children to build dynamic systems with embedded sensors and actuators.

- **Physical Elements**: GameBlocks [26] integrate tangible controls into game design, fostering creativity.
- **Robotics-Based Systems**: KIBO [4] combines wooden blocks with robotics, allowing children to program movements without screens.
- **Hybrid Approaches**: Systems like McNerney's [22] Tangible Computation Bricks blend physical and digital elements for flexibility.

These tools illustrate the diversity and potential innovation of tangible programming for educational contexts by effectively introducing programming concepts to young learners while supporting the development of CT skills.

2.4 Design and Implementations

Designing tangible programming systems involves hardware and software integration, often using microcontrollers like Arduino and sensors like RFID. TanPro-Kit, designed for 5-9-year-olds, exemplifies this, using Arduino, RFID, LED, wireless, and infrared technologies for a low-cost system [27]. It includes programming blocks and an LED pad for feedback, with a user study of 16 children showing attractiveness and ease of use. The system translates block arrangements into programming semantics, focusing on detecting and interpreting blockchains via Arduino and RFID.

Other implementations, such as roBlocks [18], use RFID for robotic construction kits, allowing users to experiment with sensors, logic, and actuator blocks. GameBlocks [25] employs magnets for contactless information transfer, controlling a toy robot. Horn and Jacob [5] developed Tern, using wooden blocks with a scanning station while focusing on cost-effectiveness and durability. These studies suggest Arduino and RFID are viable for tangible programming, with the potential for interfacing with robots like mBot2 or Nao. Feedback mechanisms, such as real-time visual and auditory cues, are critical, as seen in T-Maze's use of computer vision for live feedback, enhancing user engagement.

2.5 Evaluation and Assessment

Evaluating tangible programming systems requires systematic approaches to assess their effectiveness and technical performance. Comparative studies often benchmark tangible interfaces against graphical ones. Horn et al. [20] evaluated Tern in a museum setting, observing 260 visitors and interviewing 13 families, finding tangible interfaces led to more active participation and longer interaction times, especially for children, with girls more involved. Sapounidis and Demetriadis [25] explored preferences across age groups (5-6, 7-8, 11-12), finding younger children favoured tangibles for intuitiveness.

Beyond comparisons, usability and collaboration are key focus points. Schneider et al. [16] assessed a tangible interface's impact on collaborative learning, noting improved interaction quality among peers, a finding confirmed in classroom settings by Lee et al. [15] with robotics programs. However, few studies rigorously evaluate tangible systems in formal education contexts.

2.5.1 Research Methodologies

Research on tangible programming employs diverse methods, each offering unique insights:

- Qualitative Methods: Observations and interviews capture user experience and per-

ception [20], providing objective data but often limited to short-term effects.

- Quantitative Methods: Pre/post-tests measure learning gains, such as sequencing improvements in Kazakoff et al. [3], providing objective data but often limited to short-term effects.
- Longitudinal and Comparative Studies: Rarely employed (noted as a gap in [28]), these are needed to assess long-term impacts and compare tangible versus other approaches to validate tangible systems' broader impact.
- Design-Based Research: Iterative design and testing, as in Bers et al. [4] with KIBO to refine prototypes using real-world feedback to enhance usability and learning.

This variety ensures robust evaluation but highlights areas for further methodological exploration.

2.5.2 Success Criteria and Metrics

Evaluating tangible programming systems requires clear metrics:

- Learning Outcomes: CT skill assessments (e.g. loops, conditionals; [4]) and sequencing tests [3] quantify the educational impact.
- User Engagement: Interaction time, participation rates, and qualitative feedback [20] reflecting motivation and interest.
- Usability: Ease of use, error frequency, and user satisfaction [25] ensure accessibility, which is critical for young learners.
- System Performance: Sensor reliability and feedback responsiveness [29] validate functionality.

These metrics will provide the base framework for the project's evaluation.

2.6 Learning Outcomes

Tangible programming delivers significant educational benefits, enhancing cognitive, computational and social skills, with implications for long-term development.

2.6.1 Cognitive Development

Kazakoff et al. [3] found improved sequencing and problem-solving skills in children using tangible tools, while Wyeth [23] noted enhanced spatial reasoning in 3-8-year-olds through physical manipulation.

2.6.2 Computational Thinking Skills

Bers et al. [4] and Pugnali et al. [8] report gains in understanding loops, conditionals, and abstraction as children translate physical actions into logical sequences with tools like KIBO, reinforcing CT's applicability for digital literacy.

2.6.3 Engagement and Motivation

Horn et al. [20] and Lee et al. [15] highlight increased participation and persistence, attributing this to tangible systems' playful, hands-on nature, highlighting that tangible systems boost engagement.

These outcomes underscore the value of tangible programming in early education.

2.7 Gaps and Opportunities

Although research on tangible programming has made significant progress, several gaps prevent its full potential in educational settings. This section outlines these gaps and the current limitations and suggests future research directions to improve the research field. These points directly relate to designing and evaluating the proposed tangible programming system with the mbBot2 or Nao.

2.7.1 Current Limitations

Current tangible programming systems face practical challenges that limit their widespread adoption. Systems like KIBO [4] and roBlocks [18], while effective, often rely on specialised hardware components (e.g., sensors, microcontrollers) that increase production costs. This makes them less practical for underfunded schools or large-scale production. This contrasts with screen-based tools like Scratch [2], which benefit from widespread digital infrastructure.

Another limitation is teacher training and curriculum integration. Bers et al. [30] mention that educators often do not have the technical knowledge or teaching strategies to integrate robotics and tangible programming into existing programs, especially outside of early childhood education, which restricts the tools to be used only in specialised programs. Furthermore, technical reliability poses challenges, e.g., RFID-based systems (e.g., TanPro-Kit) may suffer from unreliable sensor detection or require frequent calibration [31], potentially disrupting learning experiences.

From a research perspective, there is a lack of standardised evaluation methods for evaluating CT in young students. Misirli and Komis [32] highlight the difficulty in measuring skills like debugging in preschoolers, pointing to a reliance on improvised or qualitative metrics rather than validated, quantitative, age-appropriate tools.

3. Design & Improvements

The redesign of the tangible programming system was guided by four primary objectives:

1. Enhancing physical robustness for educational environments
2. Improving cross-platform compatibility and maintainability
3. Providing immediate and intuitive user feedback, and
4. Ensuring cost-effective scalability for classroom deployment.

These goals addressed critical limitations identified in the 2024 prototype (Figure 1) by Jones [6] and Taylor [7], where hardware fragility, platform-specific dependencies, and a lack of user feedback hindered practical educational applications. The predecessor prototype

Figure 1 Bottom View of 2024 Prototype

featured a baseplate with pogo pins combined with multiplexers for block detection, a Pico W for data processing, and wireless communication to an mBot2 robot, establishing a functional foundation with support for basic movements and loops.

Physical robustness emerged as the foremost priority following observations of vulnerability during preliminary testing sessions. The original prototype exhibited loose electrical connections, unstable block positioning, and exposed circuitry susceptible to damage from young learners' interactions. Education environments demand systems that can withstand repeated handling, accidental impacts, and continuous operation without degradation in performance. The redesign, therefore, focused on creating a fully enclosed system with secure mechanical fastenings and protected electrical components while maintaining the essential qualities that make tangible interfaces effective for learning.

Cross-platform compatibility represented a significant technical challenge in the original C implementation, which proved difficult to reproduce across different operating systems despite the provided documentation. The development toolchain dependencies created barriers to future maintenance and expansion. The migration to MicroPython addressed these concerns by providing an OS-agnostic solution that runs consistently across Windows, macOS, and Linux platforms while significantly reducing code complexity and debugging overhead, as indicated by Table 1, where lower complexity values reflect, simpler, more maintainable code.

User feedback mechanisms were identified as being crucial for maintaining engagement and facilitating learning progression. The 2024 prototype provided a lack of visual indication of system state, requiring users to monitor the server interface for confirmation of successful command transmission. The enhanced system incorporates immediate physical feedback through discrete LEDs while maintaining comprehensive digital feedback through the web portal, creating a multi-modal feedback system that accommodates different learning preferences and environmental conditions.

Cost-effectiveness and scalability considerations influenced every design decision, as educational deployment requires multiple units at accessible price points. The Bill of Materials (BOM) analysis (Appendix B) targeted a maximum unit cost of 89.13NZD while supporting expansion capabilities for larger classroom installations. Component selection priorit-

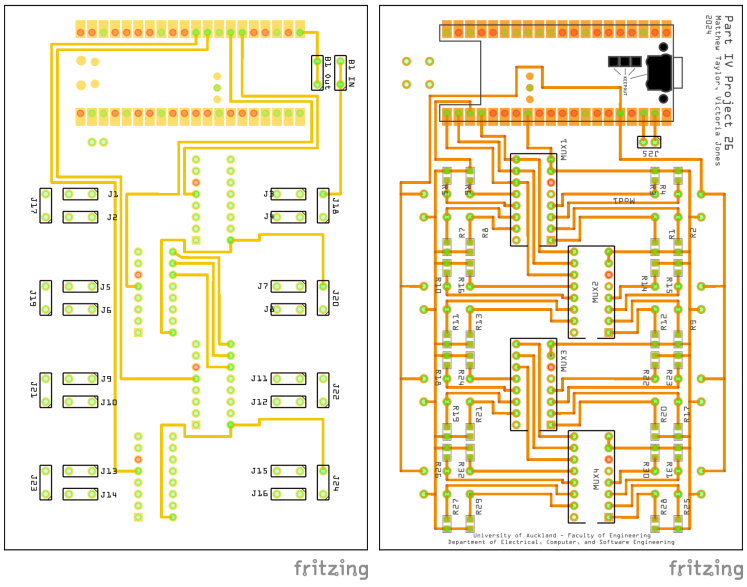
Table 1 Code Complexity Comparison

Module	Language	Lines of Code	Complexity
Wi-Fi	C	25	2
Wi-Fi	Python	45	3.67
API	C	218	4.6
API	Python	29	3
LED	C	26	1
LED	Python	11	1
Command Reader	C	128	8.3
Command Reader	Python	120	3.77
Button	C	19	1.5
Button	Python	18	2

ised readily available, standardised parts with established supply chains to ensure long-term availability and repairability in educational settings.

These design goals collectively established a framework for creating a robust, maintainable, and educationally effective tangible programming system that addresses the practical requirement of classroom deployment.

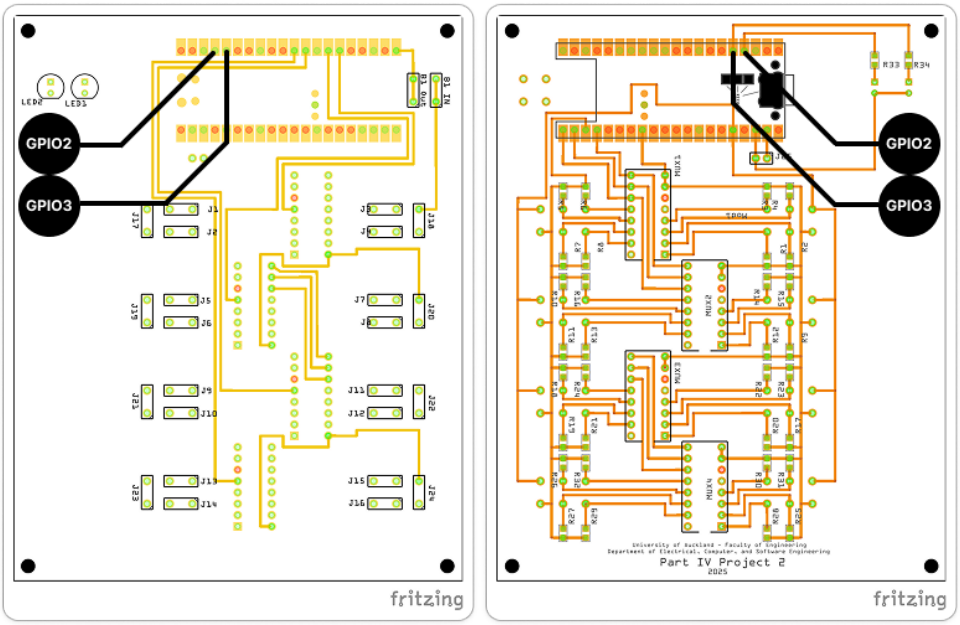
3.1 Electrical Design



(a) Top View of PCB **(b)** Bottom View of PCB

Figure 2 PCB Schematic of 2024 Prototype

The original $80x120mm$ (Figure 2), 2-layer PCB was extended to $100.8x120mm$ (Figure 3), providing additional space for visual feedback components. The dimensional increase primarily served to integrate two discrete $3mm$ through-hole LEDs positioned for optimal visibility during normal operation. The error (red) LED (`GPIO 2`) and success (green) LED (`GPIO 3`) were strategically placed to provide immediate feedback without interfering with the existing pogo pin array or multiplexer placement. Each LED incorporates a $22k\Omega$ current limiting resistor, calculated to provide sufficient brightness in any setting while minimising power consumption from the 3-cell AA battery supply. LEDs provide immediate status



(a) Top View of PCB **(b)** Bottom View of PCB

Figure 3 PCB Schematic of 2025 Prototype

feedback: pulsing during system activity, green for successful command queue additions, and red for errors (e.g. API timeouts, empty commands). The LED circuit uses firmware logic implemented in MicroPython to toggle states based on server responses.

Mounting holes were added to accommodate M3 aluminium standoffs [33], providing secure mechanical attachment to the 3D printed enclosure (Figure 4). The four mounting positions were located at the corners with sufficient clearance from electrical components to prevent short circuits while maintaining structural stability during normal use.

Female headers and IC sockets were incorporated to the updated PCB design to enhance modularity, serviceability, and long-term maintainability of the system. By replacing directly soldered components with removable header interfaces, key modules—such as the microcontroller and multiplexers—can be easily detached or replaced without desoldering. This approach significantly reduces the risk of thermal damage during maintenance and enables rapid prototyping or upgrades in future hardware revisions. Additionally, a two-pin terminal header was introduced to provide a more secure and reliable connection for the battery leads, replacing the temporary female header used in the prototype. This change improves electrical robustness and ensures consistent power delivery during extended operation.

These PCB modifications successfully integrated the enhanced feedback capabilities while improving physical robustness and maintaining backward compatibility with existing programming blocks and detection algorithms. Although the design changes resulted in an increased cost by 26% to 89.13NZD from 70.70NZD, the improvements in reliability and educational effectiveness justify the additional investment.

It is important to note that no changes were made to the core hardware responsible for block detection and data transmission. The existing circuit design—including the pogo pins, multiplexers, button, and Pico W—provided a robust foundation for sensing and interpreting block positions. As such, the majority of improvements were concentrated on firmware and software enhancements.

3.2 Firmware Development

The original 2024 system was equipped with C for programming the Pico W microcontroller, which managed block detection and command transmission to the .NET web server. Since the original development was done on macOS, reproducing the build process on other operating systems proved unreliable. Despite creating a valid CMake file, the program failed to run as expected, prompting a move away from the C toolchain.

To enhance accessibility and cross-platform compatibility, the Pico W firmware was migrated to MicroPython `v1.26.0`, a lightweight Python implementation optimised for microcontrollers. This shift makes the system OS-agnostic, allowing it to run on any device capable of executing Python scripts, such as Windows, macOS, or Linux machines. The migration involved rewriting core functions for block interpretation, Wi-Fi connectivity, and HTTP requests to the web server, while maintaining the logic for interpreting block placements into robot commands.

MicroPython was selected for its simplicity, library support and ease of debugging, which aligns with the timeline for this project. This also reduced code complexity heavily compared to the C version, as depicted in Table 1, simplifying debugging and future maintenance. This implementation ensured the firmware retained compatibility with the existing pogo pin array and multiplexer setup.

Wi-Fi connectivity utilises WPA2-PSK authentication with exponential backoff algorithms that prevent network congestion while ensuring rapid connection establishment. HTTP communication utilises the `urequests` library to implement REST API interactions with comprehensive error handling for network timeouts, server errors, and malformed responses. Request retry logic implements exponential backoff to prevent server overload while ensuring reliable data transmission.

LED control algorithms provide visual feedback through GPIO pin manipulation using MicroPython's `machine` module. State machine implementation ensures consistent LED behaviour across operational modes, with pulsing patterns during system operation, green for successful operations, and red for error conditions.

A critical enhancement involved addressing inaccuracies in the command block configurations inherited from the 2024 prototype. Initial testing revealed that the provided binary states for command blocks were misinformed. To resolve this, a simple Python script (Listing A1) was developed to systematically record the binary configurations of each block. These new configurations were compared against the outdated values, which were found to contain errors in bit assignments, and the firmware's lookup table was updated accordingly (Table 2).

3.3 Software Development

Building on the firmware migration, the software ecosystem was overhauled to enhance user interaction, performance, and system scalability. The legacy .NET web server was replaced with a modern Next.js application, written in TypeScript to improve type safety, maintainability, and scalability across future developments. The new web portal provides an intuitive digital interface that visualises system activity in real time—displaying interpreted commands and queued commands, execution status, and a list of all Pico W units and their assigned robots.

Key improvements include a live command feed for each Pico W (Figure C2), dynamically

Table 2 Comparison of Old and New Binary Values

Command	Old Binary Values	New Binary Values
Forwards	0b0011	0b0011
Backwards	0b0001	0b0010
Turn	0b1011	0b1011
Repeat	0b1111	0b1111
One	0b0010	0b0100
Two	0b0011	0b1100
Three	0b0111	0b1101
Four	0b1111	0b1111
90°	0b0010	0b0100
180°	0b0011	0b1100
270°	0b0111	0b1001

updated as commands propagate to the server, and rendered in real time to mirror physical block placements. A ping-based status system was implemented to identify online and active devices (Figure C1), enabling fault detection and disconnection awareness. In addition, future support for daisy-chained configurations was integrated into the software architecture, allowing multiple microcontrollers to be assigned to a single robot through the portal (Figure C3), significantly enhancing scalability for larger classroom or multi-robot setups.

The portal leverages Next.js server-side rendering (SSR) to ensure fast load times and responsive performance, with API routes handling communication between the frontend and backend (Table 3). These API endpoints manage frequent polling for live data while maintaining a lightweight and maintainable structure. Implementation required the integration of HTTP request handling, reactive UI logic, and asynchronous data updates to support continuous synchronisation between the client and the server.

Development challenges primarily involved maintaining low-latency updates and reliable real-time feedback. These were addressed through server response optimisations, efficient state management, and simplified API design to facilitate straightforward maintenance and future iteration. Collectively, the redesign provides a robust, extensible foundation for continued software expansion and long-term educational deployment.

Table 3 Backend API routes

Route	Methods	Purpose
api/register	GET, POST	Pings the server to register online status
api/live	GET, POST	Posts the active commands to server to display a feed
api/commands	GET, POST	Posts the active commands to server to append to command queue
api/robot/register	GET, POST	Pings the server to register online status
api/robot/commands	GET	Gets incompleted commands from the server for execution

3.4 Physical Design

The 2024 prototype (Figure 1), while innovative, exhibited vulnerabilities in physical robustness, such as loose components and weak block connections, which occasionally led to failed reads during user-led interactions. To mitigate this, the physical enclosure underwent

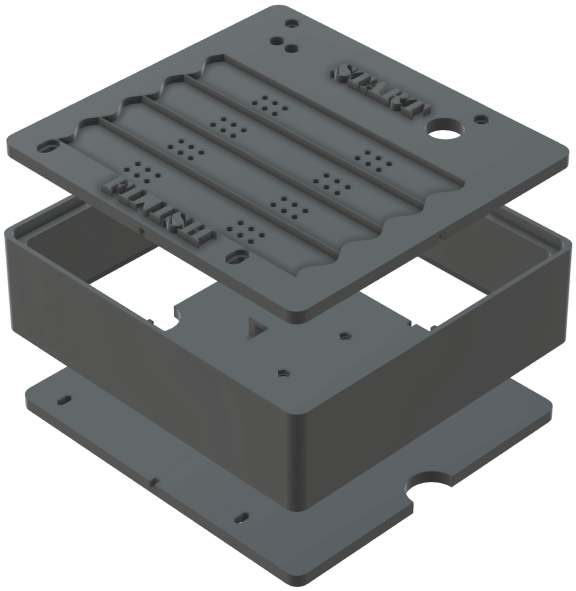


Figure 4 Exploded View of 2025 Prototype

a substantial redesign focused on improving durability, structural integrity, and overall user reliability.

The complete enclosure system (Figure 4) now consists of a 3D printed PLA top plate, a reinforced PCB mounting system, and a newly added bottom back-plate that provides comprehensive protection for internal components. These changes were developed through multiple design iterations using 3D modelling software, i.e. Fusion 360, and validated through multiple stages of laser cutting and 3D printing to optimise placements, fits, and tolerances.

The top plate utilises $5mm$ thick PLA construction to provide robust protection against impacts and wear in a classroom setting. Mounting holes were added to replace the unreliable adhesive mounting of the previous prototype with four M3 aluminium standoffs ($30mm$ height) that provide secure PCB mounting while creating adequate clearance for component height. The bottom back-plate completes the enclosure system, providing protection for the previously exposed PCB and wiring connections while adding structural integrity.

Finally, magnetic positioning was improved by upgrading from N35 to N45 grade neodymium magnets, providing a stronger holding force while maintaining the same physical dimensions. This upgrade ensures more reliable block retention during normal operation without creating excessive resistance that might frustrate young users.

4. Methodology

The methodology serves to validate the enhancements to the tangible programming system against the project's objectives: enhancing physical robustness, improving cross-platform compatibility, providing intuitive user feedback, and ensuring cost-effective scalability for classroom deployment. The process involved iterative prototyping and evaluation, designed to ensure replicability. Figure 5 illustrates the overall process, encompassing implementa-

tion, testing, and planned evaluation steps.

A pilot study was conducted to assess basic functionality, followed by a proposed user study for future comprehensive evaluation. The proposed user study was not conducted due to ethics approval delays; its design and ethical considerations are included to provide a

complete research framework. The methodology is presented to ensure others are able to replicate the procedure and can conduct similar investigations.

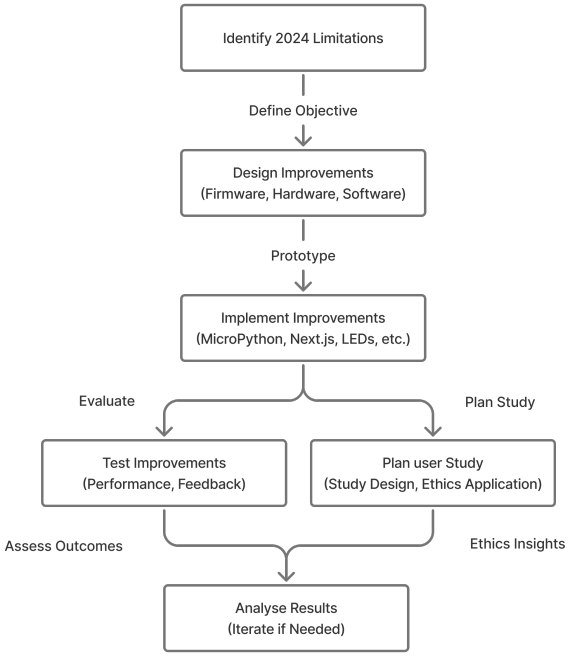


Figure 5 Overall View of Research Process

4.1 Pilot Study

A pilot study (UAHPEC Project ID 29494) was conducted with the research team in an unstructured setting to verify basic functionality, including block detection accuracy, LED feedback intuitiveness, and overall system operation. The study involved qualitative assessments through user-led interactions, focusing on enjoyment, physical robustness, and feedback systems, with results guiding further refinements.

4.2 Proposed User Study

To evaluate the system's educational effectiveness, a user study was planned with children aged 6–10 to assess usability, engagement and learning outcomes. The study was designed to test the hypothesis: "Can a tangible programming system using RFID and Arduino or connector-based systems improve children's understanding of programming concepts and engagement compared to traditional screen-based coding methods?" Due to delays in ethics approval, the study was not conducted, but its planning and ethical considerations are detailed here to highlight their importance and inform future research.

4.2.1 Study Design

The proposed study involved 10–12 children (aged 6–10) from a local primary school, recruited via school (principal) and parental consent. Multiple workshops would be run if interest exceeded 12 participants per session, ensuring each child participated in one workshop to avoid fatigue. Participants would be formed into groups of 3–4 for collaborative tasks, promoting interaction.

The study comprised of:

- **Demographics Collection**: A brief questionnaire at the start of the workshop to gather basic information (e.g. age, gender, prior exposure to programming), ensuring diversity and allowing for subgroup analysis.
- **Task-Based Testing**: Participants would program mBot2 robots using the TUI to perform tasks (e.g., navigate to a target, execute loops like repeating movements). Tasks were designed to align with computational thinking concepts (sequencing, iteration) and to be age-appropriate.
- **Control Testing**: Participants would also use a graphical interface (similar to Scratch) for comparable tasks, enabling direct comparison of effectiveness, enjoyment, and collaboration.
- **Free Play Session**: Following structured tasks, children would engage in unstructured play with both systems to assess natural preferences and creative usage.
- **Observation and Focus Groups**: Researchers would observe and video-record interactions (e.g., time to complete tasks, error frequency, collaboration patterns). A post-workshop focus group with all participants would assess enjoyment and perceived difficulty using child-friendly questions.

Sessions were planned for 1.5 hours each in a classroom setting with the research team and a teacher present for safety and facilitation. Outcomes would be analysed qualitatively (observations and focus group responses) and quantitatively (task completion rate), comparing TUI vs. screen-based systems to measure improvements in interest and skills.

4.2.2 Ethics Application Process

An ethics application (Project ID 29493) was submitted to the University of Auckland Human Participants Ethics Committee (UAHPEC). The application included:

- **Documentation**: A detailed study protocol, participant information sheets for schools, teachers, parents, and children; assent for children; consent forms for guardians, schools, and teachers; and a risk assessment ensuring no physical or psychological harm.
- **Considerations**: Privacy protection (no identifiable data in outputs, codes instead of names, no stored identity mappings), voluntary participation with rights to withdraw, and age-appropriate communication.
- **Risk Management**: Risks were assessed as low–physical (robot malfunction, mitigated by supervision); psychological (frustration from tasks, addressed via encouragement and teacher availability).

- **Data Handling**: Digital data (videos, logs) stored on password-secured University servers; physical forms (consents) in locked cabinets accessible only to researchers. Videos used for analysis, with non-essential footage deleted post-study.

This should provide a replicable framework for future researchers to execute the study, addressing limitations in this project's scope.

5. Results

A pilot study (UAHPEC Project ID 29494) was conducted with the research team in an unstructured setting to verify basic functionality, supplemented by user feedback. Due to the project timeline, no formal quantitative metrics were recorded. However, qualitative observations and user insights are reported, along with measured energy efficiency, memory usage, and latency. Results are contextualised against the objectives of robustness, compatibility, feedback, and scalability.

5.1 System Performance and Functionality

5.1.1 Firmware and Software Functionality

The migration to MicroPython enabled a functional prototype with independent module testing (e.g., block detection, Wi-Fi connectivity, and command transmission). Tests confirmed that each module operated correctly before integration, with no reported failures in basic command execution to the mBot2 robot.

The Next.js portal displayed real-time command queues and device statuses during the pilot study, with the research team noting intuitive visualisation of block placements. No specific execution rate was measured, but the system responded promptly to block changes.

5.1.2 Latency from Pico W to Server to Robot

Latency was assessed in two segments to better understand end-to-end responsiveness (Table D1).

1. Pico-to-Server Latency

   Ten trials measured the time from button press to transmit commands to receipt of the corresponding acknowledgement from the server.

   This step reflects the time taken for interrupt handling, Wi-Fi transmission, and server processing and validation.

   The average latency was $1216ms$ (SD = $569ms$), indicating generally acceptable but somewhat variable performance, likely influenced by the interrupt handler.

2. Server-to-Robot Latency

   A second set of ten measurements was recorded for the time between the server issuing commands and the robot receiving and executing them.

   This represents the server-to-robot command transmission latency and robot-side handling time.

   The average latency for this stage was $1009ms$ (SD = $85ms$), which is comparatively stable and slightly faster than the Pico-to-Server stage, likely because it relies on a straightforward `GET` request and less interrupt-driven overhead.

3. Total End-to-End Latency

   Combining both segments yields an overall average latency of $2226ms$ (SD = $574ms$) from button press to robot action.

   While this delay remains perceptible to users, it was considered acceptable for this application as it provides a short buffer for users to shift their focus from the interface to the robot.

Future iterations could reduce latency through optimised server polling, or more direct robot communication channels.

5.1.3 Physical Robustness

The addition of a backplate, standoffs, and N45 magnets resulted in a noticeably more rigid prototype. Tests indicated improved stability, with no loose components or electrical disconnections observed during handling.

Block alignment was enhanced due to the stronger magnets, reducing instances of inconsistent contact as with the 2024 version. The research team noted that the blocks snapped securely into place, improving detection reliability.

5.1.4 Feedback System Effectiveness

LEDs provided consistent feedback during tests across various scenarios: pulsing during operation, green for successful command transmissions, and red for errors (e.g. empty commands, API timeouts, API response errors). However, pilot study observations indicated that the LEDs were often unnoticed or filtered out by users.

The Next.js portal's live feed was functional; displaying interpreted and queued commands, though specific accuracy rates were not quantified.

5.1.5 Energy Efficiency

Energy efficiency was evaluated by comparing the measured usable energy under normal operating load to the nominal energy capacity of the 3-cell AA battery pack. The system was powered by three 1.5V AA batteries (nominal total 4.5V, average 2.75Ah capacity), with voltage and current measured using a digital multimeter. Measurements were taken while the system was connected to the server—the typical operational state—under two conditions: LED off (idle) and LED on (active), summarised in Table 4.

Table 4 Electrical Measurements of the LEDs under Idle and Active States

Condition	**Voltage** (V)	**Current** (A)	**Power** (W)
LED Off (Idle)	4.3	0.06	0.258
LED On (Active)	4.3	0.10	0.430

The instantaneous power is calculated as:

$$P = V \times I$$

where P is power in watts (W), V is voltage in volts (V), I is current in amperes (A).

For a battery, total energy capacity can be expressed as:

$$E = V \times Q$$

where E is energy in watt-hours (Wh), V is voltage in volts (V), Q is capacity in ampere-hours (Ah).

Efficiency is computed as:

$$\text{Efficiency} = \frac{E_{\text{measured}}}{E_{\text{nominal}}} \times 100\%$$

Nominal energy capacity was:

$$E_{\text{nominal}} = 4.5 \text{ V} \times 2.75 \text{ Ah} = 12.375 \text{ Wh}$$

Measured usable energy under load was:

$$E_{\text{measured}} = 4.3 \text{ V} \times 2.75 \text{ Ah} = 11.825 \text{ Wh}$$

Resulting energy efficiency was:

$$\text{Efficiency} = \frac{11.825}{12.375} \times 100\% \approx 95.6\%$$

Battery life estimates, based on the measured current draw with an average 2.75Ah battery capacity, are presented in Table 5.

Table 5 Battery Life Estimates Per Measured Current

Condition	**Current (A)**	**Battery Life(h)**
LED Off (Idle)	0.06	45.8
LED On (Active)	0.10	27.5

This high efficiency ($\approx 96\%$) demonstrates minimal voltage drop and effective power utilisation under typical operation. The estimated battery life indicates that the prototype can sustain multiple class sessions without replacement. No overheating or voltage instability was observed during hour-long test sessions.

5.1.6 Memory Usage on Pico W

Memory usage on the Pico W was logged using MicroPython's `mem_info()` function during operation. Idle usage was approximately 29 kilobytes used (171 kilobytes free) out of a total 200 kilobyte heap, with stack consumption at 8% (604 bytes of 7936 bytes). During block scanning and transmission, memory usage remained stable with no indication of leaks or overflows. This demonstrates efficient resource management, supported by Micropython's effective garbage collection and optimised code execution.

5.1.7 Scalability and Modularity

Modular headers allowed easy swapping of the Pico W and multiplexers during testing, with no observed performance issues. The research team confirmed the system's potential for daisy chaining, though this was not fully tested with multiple systems.

Cost estimates remain preliminary, with the Bill of Materials suggesting a target of approximately 89.13NZD per unit (Table B2), based on standardised components.

5.2 Pilot Study Findings

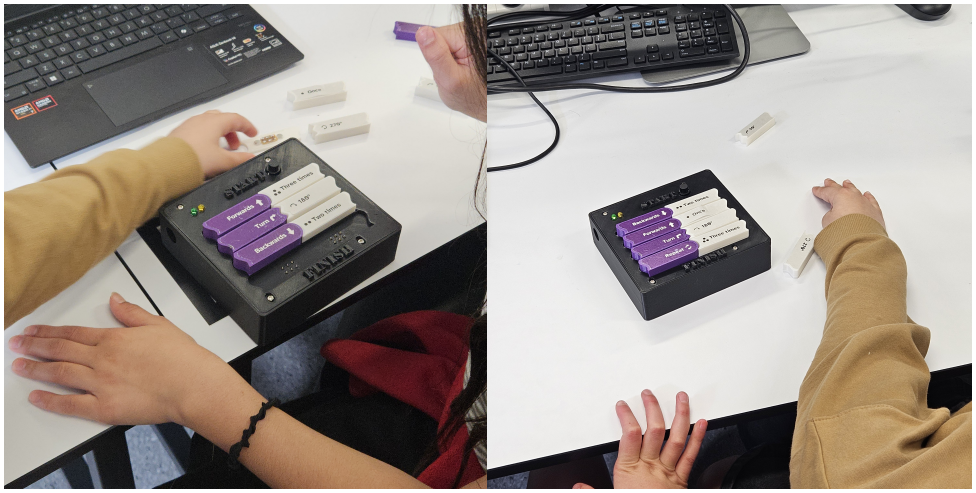


Figure 6 Participants interacting with the tangible programming system during the pilot study session.

The unstructured pilot study (Figure 6) provided participants with the opportunity to engage in freeplay with the system, while giving a starting challenge—for example, "Make the robot move in a square"—to prompt initial engagement with problem-solving. Initially, participants tended to approach the challenges individually; however, as they experimented with the system, collaboration naturally emerged. Participants began discussing strategies, sharing ideas, and introducing additional objectives of their own, such as navigating the robot to a peer or reaching specific destinations. This demonstrates that the system encourages creative problem-solving and peer-to-peer learning.

5.2.1 Positive Feedback

Participants appreciated the robot's responsive movements corresponding to the blocks, and the 3D printed design and feel. The number of blocks was considered appropriate by participants, stating that this challenges them to approach problems creatively without being overwhelmed. Users also found the system engaging, with many expressing interest in taking it home if a complete version were available.

5.2.2 Negative Feedback

Some issues were observed with inconsistent robot actions, mainly caused by occasional block contact problems, requiring participants to press hold certain blocks in place for reliable detection. While the system provides LED-based status feedback, participants largely ignored the LEDs, instead relying on visual confirmation by observing the robot directly. A few participants noted that the robot's actions did not always match their expectations, highlighting areas for improvement in block detection and reliability.

Overall, the pilot study highlighted that the system fosters group collaboration, encourages creative engagement, and effectively motivates participants to explore additional challenges beyond the given tasks, despite minor technical limitations.

Figure 7 Assembled final prototype of the tangible programming system.

6. Discussion

This section explores the broader implications of the improved tangible programming system (Figure 7) for educational innovation, with an emphasis on fostering CT, creativity, and collaboration in young learners. The prototype's advancements are contextualised against the 2024 version and commercial tools, emphasising potential for classroom integration, curriculum enhancement, and future research in robotics education.

6.1 Evaluation of Objectives

The prototype successfully met its primary objectives of improving rigidity, cross-platform compatibility, feedback mechanisms, and scalability. Migrating from C to MicroPython provided cross-platform support and simplified firmware updates, making the system compatible with a wider range of classroom devices. The redesigned hardware with aluminium standoffs, a full 3D printed enclosure, and stronger magnets significantly improved structural stability and block alignment, reducing mechanical and electrical faults.

Energy efficiency (96%) and low latency (average 2.23 seconds from block to robot) demonstrate the system's suitability for classroom use, where consistent and responsive behaviour is crucial for maintaining engagement. The LED feedback and Next.js portal improved transparency of system states, though user feedback indicated that children relied more on the robot's response and movement than on LED indicators.

6.2 Educational Implications

From an educational perspective, this system reinforces the pedagogical value of tangible interfaces in promoting CT and problem-solving. TUIs align with constructionist learning theory, where knowledge is built through physical manipulation and experimentation rather than abstract instruction. Pilot study sessions revealed collaborative behaviours similar to those observed in other tangible programming studies—children transitioned from individual problem solving to group collaboration and creative goal-setting.

The system's simplicity and physical immediacy can make abstract programming concepts like sequencing and cause-and-effect relationships more intuitive. However, the current

set of blocks limits the system's expressions. Expanding the block set to include control structures (e.g. conditionals or sensor-based inputs) could deepen learning by introducing children to foundational programming logic while maintaining tangible engagement.

6.3 Scalability and Classroom Integration

The modular hardware design and web-based architecture offer promising scalability for classroom environments. The Pico W's Wi-Fi capability allows multiple systems to operate concurrently, each linked to a central web server for monitoring and management. This structure supports multi-group classroom activities, where each group can experiment with its own robot while the teacher monitors progress.

Additionally, the physical robustness achieved through the redesigned enclosure makes the system more suitable for repetitive use, a critical factor for classroom adoption. Future iterations could include daisy-chained baseplates or stackable modules to accommodate larger programs or group projects.

6.4 Cost Analysis

A key advantage of the redesigned system is its affordability. The Bill of Materials (Appendix B) estimates a total cost of approximately 89.13NZD per unit, which compares favourably to commercial educational robotics kits such as LEGO Education [34] or Bee-Bot TacTile Reader [35] sets that can easily exceed 300NZD. Using off-the-shelf components such as the Pico W, and 3D printed parts reduces production costs and simplifies maintenance.

This cost-efficiency makes the system viable for schools with limited budgets and aligns with the project's goal of developing accessible educational technology. Component sourcing strategies focused on availability and long-term supply chain stability rather than purely minimising unit costs, ensuring repairability and scalability in educational contexts. This approach aligns with the goal of cost-effective deployment, though bulk purchasing could further reduce costs in future iterations.

6.5 Technical Challenges

Despite its success, the system still faces several technical challenges that must be addressed for future deployment. The most significant is the unreliable pogo pin connections used to interface with the blocks. Future designs could explore alternative approaches, such as short-range RFID, capacitive touch, or magnetic connectors, to improve reliability. The LED indicators, while functional, were often overlooked by users; future designs could incorporate multimodal feedback such as sound, vibration, or LEDs on each block slot. Finally, integration testing across multiple devices and networks will be necessary to ensure consistent behaviour in real-world classroom conditions.

7. Future Work

These recommendations are for advancing the tangible programming system beyond the current prototype. The suggestions build on the project's achievements in robustness, compatibility, and modularity while addressing limitations such as untested scalability, feedback visibility, and empirical validation through user studies. These proposals aim to enhance the system's suitability for educational environments by incorporating more advanced features and rigorous evaluation.

7.1 User Study Implementation

Given the functional state of the current prototype, a primary focus for future work should be conducting the proposed user study to empirically assess its educational effectiveness. The study, as outlined in the UAHPEC application (Project ID 29493), involves 1.5-hour workshops with 10-12 children per session at local schools, comparing the TUI to a screen-based system similar to Scratch.

Re-submitting the ethics application with the advised revisions is essential, drawing from feedback from the latest submission this year. Once approved, the study can proceed with the prepared materials, including participant information sheets, assent forms, consent forms, observation schedule, questionnaires, and focus group questionnaires. This will validate the hypothesis that tangible programming improves enjoyment, collaboration, and introductory skills, providing quantitative data and qualitative insights. Conducting multiple workshops (at least two, as planned) will ensure reliable sample sizes for subgroup analysis, ultimately confirming the system's real-world impact.

7.2 Hardware Improvements

7.2.1 Block Detection

To address the occasional contact inconsistencies noted in testing, two alternative approaches to the current pogo pin connectors are proposed: short-range RFID blocks and magnetic pogo pin connectors.

- **Short-Range RFID**: Implementing low-cost RFID tags would eliminate physical contact issues, enhancing reliability and reducing wear. It could use low-cost tags embedded in blocks and readers on the baseplate. While adding an extra unit cost, it would enhance durability and user experience, aligning with scalability goals for expanded block sets.
- **Magnetic Pogo Contacts**: Replacing the existing pogo pin and magnet array with a magnetic pogo pin connector (Figure 8) would combine the precision of electrical contacts with magnetic guidance for seamless block placement. This hybrid design ensures reliable electrical connectivity without manual pressure.

7.2.2 Per-Block LED Feedback

The current success/error LEDs were functional but often unnoticed by children, suggesting a redesign to include an LED per block placement. This would provide granular feedback, making debugging more intuitive and engaging. Integration into the baseplate via the Pico W's GPIO pins would require minimal firmware changes, but iterative prototyping in Fusion 360 would ensure compatibility with the enclosure. This upgrade addresses the feedback limitation, potentially increasing child engagement by visual cues at each interaction point.

7.2.3 Redesigned Block Shapes

Changing block shapes to simpler geometrics, similar to Figure 9, would facilitate easier reproduction and expansion. Using CAD tools like Fusion 360, new shapes could maintain magnetic syntax enforcement while allowing for modular additions (e.g., new command types).

(a) Male Connector [36] **(b)** Female Connector [37]

Figure 8 Magnetic Pogo Contacts

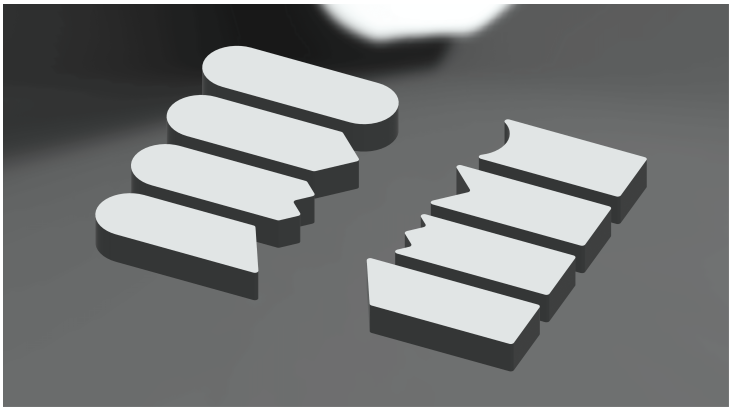


Figure 9 Newly Proposed Block Shapes

7.2.4 Integrated Display for Commands

Incorporating a small display or seven-segment display per row on the baseplate would allow real-time visualisation of interpreted commands, complementing the Next.js portal. This could be OLED modules connected to the Pico W, displaying sequences (e.g. "Forward 2" or "F2") to aid in debugging without the portal. It would enhance stand-alone usability in resource-limited settings, addressing the need for more immediate, child-friendly feedback beyond LEDs.

These recommendations, if implemented, would evolve the prototype into a more versatile educational tool, with priority given to the user study for validation.

7.3 Future Expansion Opportunities

The results of this project demonstrate a stable and scalable foundation for tangible programming, supporting further development in both hardware and software. Several potential expansion directions are identified based on the current performance and system design.

Firstly, the modular architecture allows for additional command and modifier blocks, extending the programming expressiveness beyond movement and repetition. New block categories, such as conditionals, sensor triggers, or function calls, could be implemented to introduce more advanced programming concepts while maintaining the tangible nature of

interaction.

Second, the system could be expanded to support multiple robots or networked group activities, leveraging the existing server-client model to coordinate collaborative tasks between different groups of learners.

Finally, future iterations could integrate sensing and feedback mechanisms—for example, colour or proximity sensors using the camera that allow the robot to react to its environment. These features would support more complex challenges, such as navigation, decision-making, or creative storytelling, aligning with constructionist learning approaches that emphasise open-ended exploration.

7.4 Pathways for Educational Integration

The performance and usability outcomes indicate strong potential for classroom deployment. The system's low latency, high energy efficiency, and rigid construction make it well suited for repeated use in primary education settings.

To introduce the system effectively into an educational context, a scaffold curriculum could be developed that aligns tangible programming tasks with computational thinking concepts. For example, early lessons could focus on sequencing simple movements, while later activities could introduce loops, logic, or event-driven programming using expanded block sets.

The web portal could serve as a central management tool for teachers, displaying active devices, current programs, and student progress in real time. This would allow teachers to monitor classroom activity, and assist students easily.

For long-term adoption, partnerships with schools and teacher-training programs could help evaluate learning outcomes and refine the interface based on educator feedback.

8. Conclusions

This research project successfully extended the 2024 tangible programming prototype into a more resilient, multi-platform, and educationally viable system. Through hardware redesign, firmware migration to MicroPython, and the development of a Next.js monitoring portal, the prototype achieved greater structural integrity, usability, and system transparency. Testing confirmed a 96% energy efficiency, stable memory use, and acceptable command latency between the Pico W and mBot2 robot at 2.2seconds, confirming the technical viability of the enhanced design.

Beyond its design achievements, the project reinforces the educational values of TUIs for teaching CT. Pilot sessions revealed that children naturally transitioned from independent experimentation to collaborative problem-solving. These observations affirm that tangible programming can serve as an accessible entry point to programming and robotics for early learners through hands-on physical interaction.

The redesigned system also demonstrates clear potential for classroom integration. It's modular, low-cost design enables scalability across multiple students, and it's cross-platform software framework supports easy setup and maintenance for educators. Together, these features position the system as a practical foundation for developing affordable, engaging, and durable educational robotics tools. It also makes the system well suited for integration into structured classroom activities, robotics workshops, or informal learning environments.

However, several challenges remain, particularly improving block connection reliability and expanding the range of programming blocks. Furthermore, conducting comprehensive user studies, refining sensing mechanisms, and exploring multimodal feedback will further improve usability and engagement.

In summary, this research demonstrates that tangible programming systems can effectively bridge the physical interaction with computational reasoning, making programming education for young learners both intuitive and inclusive. The outcomes of this work provide a strong foundation for continued exploration of tangible interfaces as scalable, engaging tools for the next generation of learners.

References

[1] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[2] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[3] E. R. Kazakoff, A. Sullivan, and M. U. Bers, "The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood," *Early Childhood Education Journal*, vol. 41, pp. 245–255, 2013.

[4] M. U. Bers, L. Flannery, E. R. Kazakoff, and A. Sullivan, "Computational thinking and tinkering: Exploration of an early childhood robotics curriculum," *Computers & education*, vol. 72, pp. 145–157, 2014.

[5] M. S. Horn and R. J. Jacob, "Designing tangible programming languages for classroom use," in *Proceedings of the 1st international conference on Tangible and embedded interaction*, 2007, pp. 159–162.

[6] V. Jones, "Physical coding blocks for robotics and software education," Oct. 2024, unpublished undergraduate dissertation.

[7] M. Taylor, "Physical coding blocks for robotics and software education," Oct. 2024, unpublished undergraduate dissertation.

[8] A. Pugnali, A. Sullivan, and M. U. Bers, "The impact of user interface on young children's computational thinking," *Journal of Information Technology Education. Innovations in Practice*, vol. 16, p. 171, 2017.

[9] C. S. T. Association *et al.*, "Operational definition of computational thinking," *Report*, p. 1, 2011.

[10] V. Barr and C. Stephenson, "Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community?" *ACM inroads*, vol. 2, no. 1, pp. 48–54, 2011.

[11] M. U. Bers, *Designing digital experiences for positive youth development: From playpen to playground*. Oxford University Press, 2012.

[12] S. Papert, *Children, computers, and powerful ideas*. Eugene, OR, USA: Harvester, 1980, vol. 10.

[13] M. U. Bers, "The tangiblek robotics program: Applied computational thinking for young children." *Early Childhood Research & Practice*, vol. 12, no. 2, p. n2, 2010.

[14] J. Sweller, "Cognitive load during problem solving: Effects on learning," *Cogn. Sci.*, vol. 12, no. 2, pp. 257–285, 1988.

[15] K. T. Lee, A. Sullivan, and M. U. Bers, "Collaboration by design: Using robotics to foster social interaction in kindergarten," *Computers in the Schools*, vol. 30, no. 3, pp. 271–281, 2013.

[16] B. Schneider, P. Jermann, G. Zufferey, and P. Dillenbourg, "Benefits of a tangible interface for collaborative learning and interaction," *IEEE Transactions on Learning Technologies*, vol. 4, no. 3, pp. 222–232, 2010.

[17] A. N. Antle, "Designing tangibles for children: what designers need to know," in *CHI'07 Extended Abstracts on Human Factors in Computing Systems*, 2007, pp. 2243–2248.

[18] E. Schweikardt and M. D. Gross, "roblocks: a robotic construction kit for mathematics and science education," in *Proceedings of the 8th international conference on Multimodal interfaces*, 2006, pp. 72–75.

[19] J. Piaget, *Science of education and the psychology of the child*. Trans. D. Coltman, 1970.

[20] M. S. Horn, E. T. Solovey, R. J. Crouser, and R. J. Jacob, "Comparing the use of tangible and graphical programming languages for informal science education," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2009, pp. 975–984.

[21] D. Wang, T. Wang, and Z. Liu, "A tangible programming tool for children to cultivate computational thinking," *The Scientific World Journal*, vol. 2014, no. 1, p. 428080, 2014.

[22] T. S. McNerney, "From turtles to tangible programming bricks: explorations in physical language design," *Personal and Ubiquitous Computing*, vol. 8, pp. 326–337, 2004.

[23] P. Wyeth, "How young children learn to program with sensor, action, and logic blocks," *The Journal of the learning sciences*, vol. 17, no. 4, pp. 517–550, 2008.

[24] H. Suzuki and H. Kato, "Algoblock: a tangible programming language, a tool for collaborative learning," in *Proceedings of 4th European Logo Conference*, 1993, pp. 297–303.

[25] T. Sapounidis and S. Demetriadis, "Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences," *Personal and ubiquitous computing*, vol. 17, pp. 1775–1786, 2013.

[26] A. C. Smith, "Using magnets in physical blocks that behave as programming objects," in *Proceedings of the 1st international conference on Tangible and embedded interaction*, 2007, pp. 147–150.

[27] D. Wang, Y. Qi, Y. Zhang, and T. Wang, "Tanpro-kit: a tangible programming tool for children," in *Proceedings of the 12th International Conference on Interaction Design and Children*, ser. IDC '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 344–347. [Online]. Available: https://doi.org/10.1145/2485760.2485841

[28] I. Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, J. Malyn-Smith, and L. Werner, "Computational thinking for youth in practice," *Acm Inroads*, vol. 2, no. 1, pp. 32–37, 2011.

[29] N. Nusen and A. Sipitakiat, "Robo-blocks: A tangible programming system with debugging for children," in *Proceeding of the 19th International Conference on*

Computers in Education, ICCE 2011, Thailand Science Park, Chiang Mai, Thailand, November 28 - December 1, 2011. Asia-Pacific Society for Computers in Education, 2011. [Online]. Available: https://library.apsce.net/index.php/ICCE/article/view/2626

[30] M. U. Bers, I. Ponte, C. Juelich, A. Viera, and J. Schenker, "Teachers as designers: Integrating robotics in early childhood education," *Information technology in childhood education annual*, vol. 2002, no. 1, pp. 123–145, 2002.

[31] P. Marshall, "Do tangible interfaces enhance learning?" in *Proceedings of the 1st international conference on Tangible and embedded interaction*, 2007, pp. 163–170.

[32] A. Misirli and V. Komis, "Computational thinking in early childhood education: The impact of programming a tangible robot on developing debugging knowledge," *Early Childhood Research Quarterly*, vol. 65, pp. 139–158, Oct. 2023, zSCC: 0000038. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0885200623000765

[33] DigiKey, "970300366." [Online]. Available: https://www.digikey.co.nz/en/products/detail/würth-elektronik/970300366/9488510

[34] PBTech, "Buy the LEGO Education 45400 BricQ Motion Prime Set, Ages 10+, 564 pcs (LEG45400) online." [Online]. Available: https://www.pbtech.co.nz/product/TOYLEG0012/LEGO-Education-45400-BricQ-Motion-Prime-Set-Ages-1

[35] ——, "Buy the Bee-Bot Education STEM TTSB1118 BlueBot Programming Tactile Reader (TTSB1118) online." [Online]. Available: https://www.pbtech.co.nz/product/TOYBEB0005/Bee-Bot-Education-STEM-TTSB1118-BlueBot-Programmin

[36] DigiKey, "685B0621250124E." [Online]. Available: https://www.digikey.co.nz/en/products/detail/edac-inc/685B0621250124E/21299686

[37] ——, "686B0621250124E." [Online]. Available: https://www.digikey.co.nz/en/products/detail/edac-inc/686B0621250124E/21299720

[38] Pcbway.com, "China pcb prototype & fabrication manufacturer - pcb prototype the easy way," 2024. [Online]. Available: https://pcbway.com

[39] DigiKey, "RP2040-PICO-W." [Online]. Available: https://www.digikey.co.nz/en/products/detail/olimex-ltd/RP2040-PICO-W/22479731

[40] ——, "CD74HC251E." [Online]. Available: https://www.digikey.co.nz/en/products/detail/texas-instruments/CD74HC251E/1506904

[41] ——, "RC0201FR-0722KL." [Online]. Available: https://www.digikey.co.nz/en/products/detail/yageo/RC0201FR-0722KL/3202394

[42] ——, "HDR200MET40F-G-V-TH." [Online]. Available: https://www.digikey.co.nz/en/products/detail/chip-quik-inc/HDR200MET40F-G-V-TH/18869954

[43] ——, "61304011121." [Online]. Available: https://www.digikey.co.nz/en/products/detail/würth-elektronik/61304011121/4846884

[44] ——, "110-93-316-41-001000." [Online]. Available: https://www.digikey.co.nz/en/products/detail/mill-max-manufacturing-corp/110-93-316-41-001000/14041

[45] ——, “D6R90 F1 LFS.” [Online]. Available: https://www.digikey.co.nz/en/products/detail/c-k/D6R90-F1-LFS/1466335

[46] ——, “1800001.” [Online]. Available: https://www.digikey.co.nz/en/products/detail/phoenix-contact/1800001/4482846

[47] ——, “151031VS06000.” [Online]. Available: https://www.digikey.co.nz/en/products/detail/würth-elektronik/151031VS06000/4489988

[48] ——, “151051RS11000.” [Online]. Available: https://www.digikey.co.nz/en/products/detail/würth-elektronik/151051RS11000/4490012

[49] ——, “7913-0-15-20-77-14-11-0.” [Online]. Available: https://www.digikey.co.nz/en/products/detail/mill-max-manufacturing-corp/7913-0-15-20-77-14-11-0/8827314

[50] ——, “RM3X10MM 2701.” [Online]. Available: https://www.digikey.co.nz/en/products/detail/apm-hexseal/RM3X10MM-2701/3712296

[51] ——, “2465.” [Online]. Available: https://www.digikey.co.nz/en/products/detail/keystone-electronics/2465/303814

[52] A. Magnetics, “Neodymium Disc Magnet - 6mm x 2mm | N35 | Diametrically Magnetised.” [Online]. Available: https://magnet.com.au/products/neodymium-disc-6mm-x-2mm-diametric-n35

[53] ——, “Neodymium Disc Magnet - 6mm x 1.5mm | N45.” [Online]. Available: https://amfmagnets.co.nz/products/neodymium-disc-magnet-6mm-x-1-5mm-n45

Appendix A Block Recording Python Script

```python
def select_channel(ch: int):
    """Set the select lines to choose channel 0-7"""
    sel0_pin.value((ch >> 0) & 1)
    sel1_pin.value((ch >> 1) & 1)
    sel2_pin.value((ch >> 2) & 1)


def read_muxes(ch: int):
    """Read all 4 muxes on the selected channel"""
    select_channel(ch)
    sleep_us(5)  # tiny settle time
    return [mux.value() for mux in mux_outputs]


def read_primary_blocks():
    """Read only the primary blocks (channels 0-3)"""
    print("=== PRIMARY BLOCKS (Action Type) ===")
    primary_data = []
    for channel in range(4):  # Channels 0-3
        values = read_muxes(channel)
        primary_data.append(values)
        print(f"Channel {channel}: {values}")
    return primary_data


def read_secondary_blocks():
    """Read only the secondary blocks (channels 4-7)"""
    print("=== SECONDARY BLOCKS (Magnitude/Quantity) ===")
    secondary_data = []
    for channel in range(4, 8):  # Channels 4-7
        values = read_muxes(channel)
        secondary_data.append(values)
        print(f"Channel {channel}: {values}")
    return secondary_data


def read_blocks_separated():
    """Read both primary and secondary blocks separately"""
    primary = read_primary_blocks()
    print()  # Add spacing
    secondary = read_secondary_blocks()
    return primary, secondary
```

Program A1 Script to record binary configurations of blocks.

Appendix B Bill of Materials

The original source for the pogo pins could not be identified from 2024's reports; therefore, an equivalent component from DigiKey was selected. Resistors, headers, sockets, Raspberry Pi Pico boards, and LEDs were sourced from the University's component store, with DigiKey equivalents listed for reference.

Table B1 Bill of Materials for 2024 Prototype (NZD)

Reference	Component	Part Number	Quantity	Unit Cost	Total
[38]	PCB		1	5	5
[39]	Raspberry Pi Pico W	RP2040-PICO-W	1	10.26	10.26
[40]	8x1 Multiplexer	CD74HC251E	4	0.8024	3.2096
[41]	22kohm Surface Mount Resistors	RC0201FR-0722KL	32	0.0072	0.2304
[42]	Female Headers	HDR200MET40F-G-V-TH	1	1.89	1.89
[43]	Male headers	61304011121	1	0.8136	0.8136
[44]	IC Socket	110-93-316-41-001000	0	1.888	0
[45]	Pushbutton	D6R90 F1 LFS	1	1.2622	1.2622
[46]	Terminal header	1800001	0	1.1618	0
[47]	Green LED	151031VS06000	0	0.1396	0
[48]	Red LED	151051RS11000	0	0.1396	0
[49]	Pogo pin	7913-0-15-20-77-14-11-0	48	$0.76	36.5088
[50]	M3 Screws 10mm	RM3X10MM 2701	0	$0.42	0
[33]	M3 Standoff 30mm	970300366	0	$1.12	0
[51]	Battery holder	2465	1	1.45	1.45
[52]	N35 magnet	93201	16	0.63	10.08
[53]	N45 magnet	D-D6H1.5-N45	0	0.73	0
				Total Cost	**70.7046**

Table B2 Bill of Materials for 2025 Prototype (NZD)

Reference	Component	Part Number	Quantity	Unit Cost	Total
[38]	PCB		1	5	5
[39]	Raspberry Pi Pico W	RP2040-PICO-W	1	10.26	10.26
[40]	8x1 Multiplexer	CD74HC251E	4	0.8024	3.2096
[41]	22kohm Surface Mount Resistors	RC0201FR-0722KL	34	0.0072	0.2448
[42]	Female Headers	HDR200MET40F-G-V-TH	1	1.89	1.89
[43]	Male headers	61304011121	1	0.8136	0.8136
[44]	IC Socket	110-93-316-41-001000	4	1.888	7.552
[45]	Pushbutton	D6R90 F1 LFS	1	1.2622	1.2622
[46]	Terminal header	1800001	1	1.1618	1.1618
[47]	Green LED	151031VS06000	1	0.1396	0.1396
[48]	Red LED	151051RS11000	1	0.1396	0.1396
[49]	Pogo pin	7913-0-15-20-77-14-11-0	48	$0.76	36.48
[50]	M3 Screws 10mm	RM3X10MM 2701	8	$0.42	3.36
[33]	M3 Standoff 30mm	970300366	4	$1.12	4.48
[51]	Battery holder	2465	1	1.45	1.45
[52]	N35 magnet	93201	0	0.63	0
[53]	N45 magnet	D-D6H1.5-N45	16	0.73	11.68
				Total Cost	**89.1296**

Table B3 Bill of Materials Delta for 2024 and 2025

Component	Part Number	Delta Cost (NZD)
PCB		0
Raspberry Pi Pico W	RP2040-PICO-W	0
8x1 Multiplexer	CD74HC251E	0
22kohm Surface Mount Resistors	RC0201FR-0722KL	0.0144
Female Headers	HDR200MET40F-G-V-TH	1.89
Male headers	61304011121	0
IC Socket	110-93-316-41-001000	7.552
Pushbutton	D6R90 F1 LFS	0
Terminal header	1800001	1.1618
Green LED	151031VS06000	0.1396
Red LED	151051RS11000	0.1396
Pogo pin	7913-0-15-20-77-14-11-0	0
M3 Screws 10mm	RM3X10MM 2701	3.3568
M3 Standoff 30mm	970300366	4.4608
Battery holder	2465	0
N35 magnet	93201	-10.08
N45 magnet	D-D6H1.5-N45	11.68

Table B4 Bill of Materials for 3D Printed Components

Component	Weight (g)	Cost (NZD)
Top Plate	53.57	1.55
Middle	92.69	2.69
Bottom Plate	54.72	1.59

Appendix C Web Portal Interface

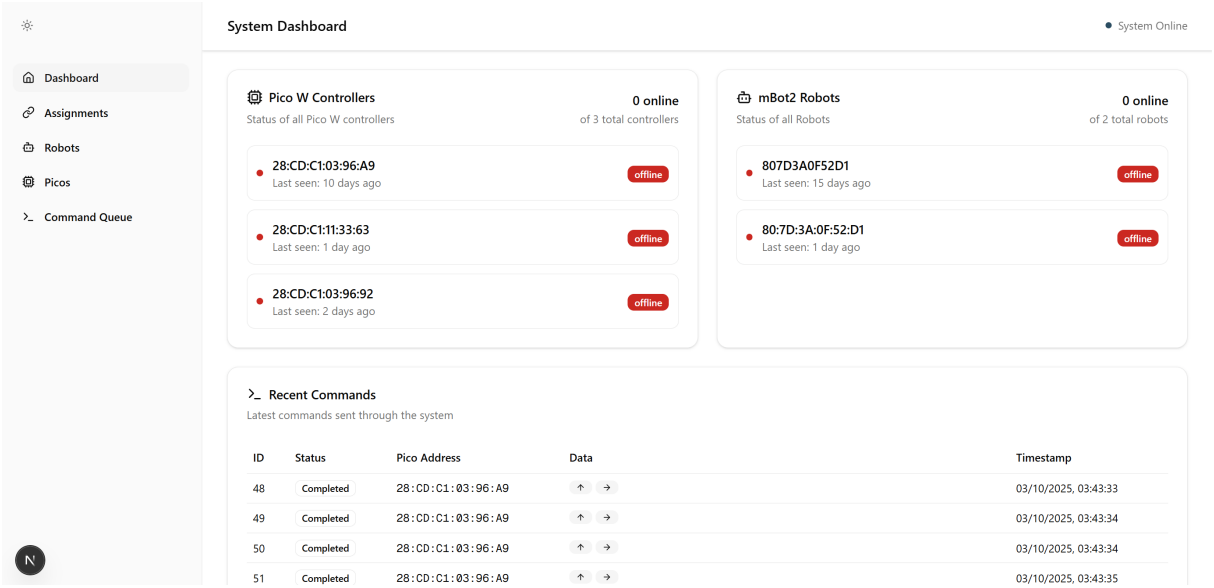


Figure C1 Home Page of New Web Server

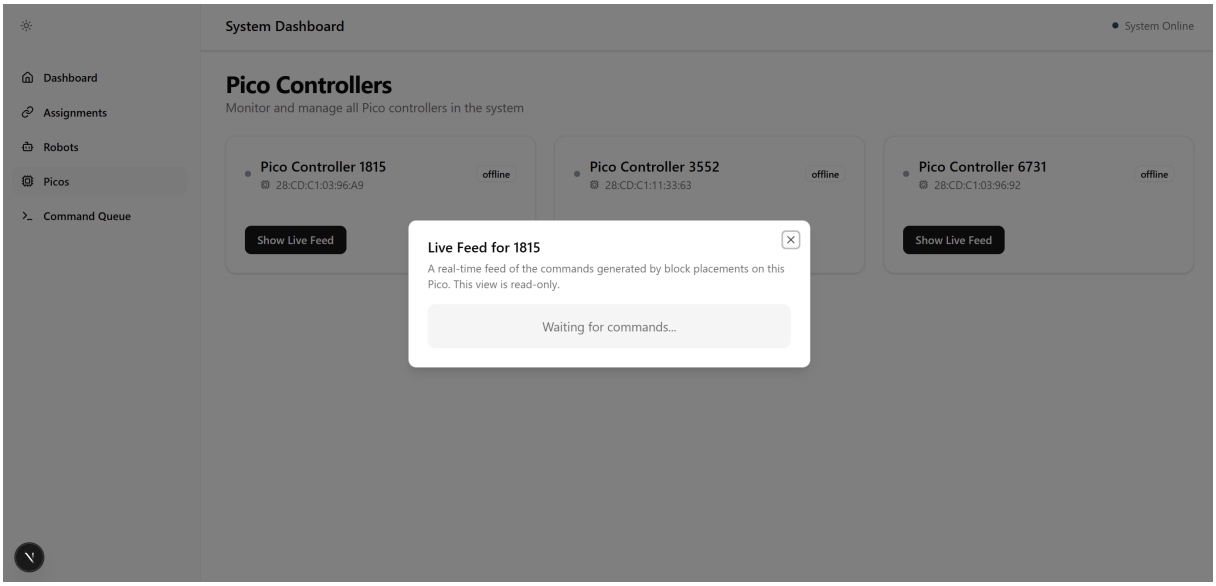


Figure C2 Live Feed of a Pico W System

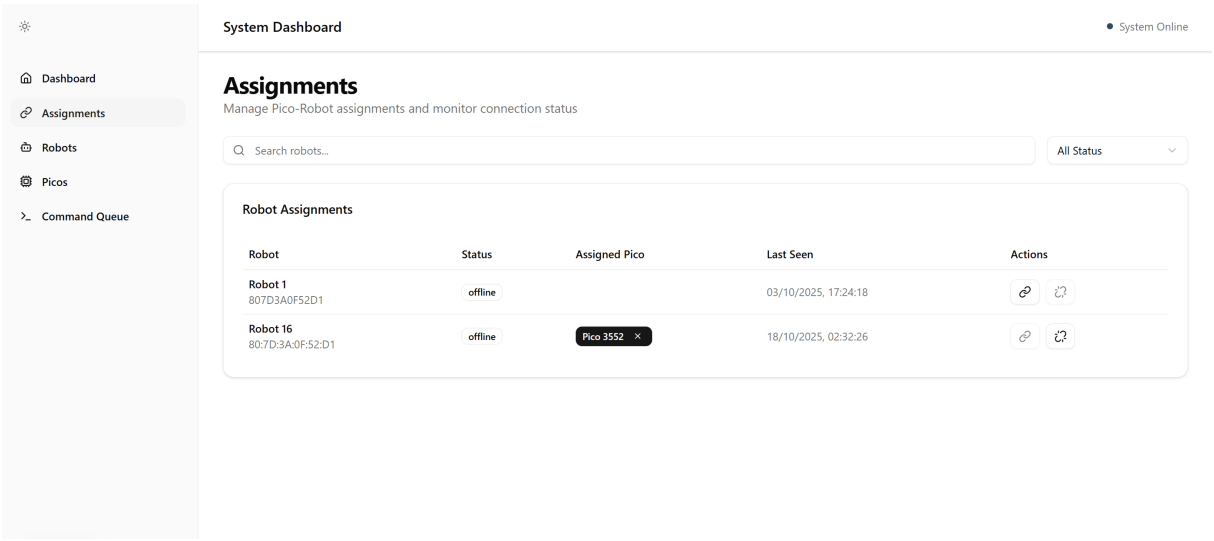


Figure C3 Assignments List of Picos to Robots

Appendix D Latency Measurement Data

Table D1 Latency Measurements Between System Components

Trial	Pico to Server (ms)	Server to Robot (ms)
1	1060	917
2	1091	919
3	915	1138
4	2274	1030
5	915	923
6	2222	1032
7	642	1137
8	821	1036
9	1274	1029
10	950	931
Average	**1216**	**1009**
Standard Deviation (SD)	**569**	**85**