IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2026

DOI: 10.17148/IJIREEICE.2026.14354

Raspberry Pi Pico Hybrid HID Controller

Surya Sevak Singh[1], Siddharth Sonawane[2], Vineet Raipure[3], Siddhesh Gaikwad[4], Pranav Kamble[5], Om Sakpal[6]

Students, Electronic and telecommunication, Bharti Vidyapeeth Institute of Technology, Navi Mumbai India[1-6]

Abstract: This paper presents the complete design, hardware implementation, and firmware development of a low-cost, multi-mode USB Human Interface Device (HID) controller built around the Raspberry Pi Pico (RP2040) microcontroller. The proposed system integrates a dual-axis analog joystick HW-504 and eleven programmable tactile buttons into a single platform capable of simultaneously emulating both a USB keyboard and a USB mouse without requiring any custom driver installation on the host system. The controller is implemented using the TinyUSB stack via the Pico C++ SDK and communicates over USB Full-Speed using standard HID class descriptors. Hardware was iteratively refined across three breadboard revisions, with Version 3 representing the most stable configuration. The system was successfully demonstrated and validated on Windows 11 with commercial game titles including Grand Theft Auto V and Red Dead Redemption 1, and was exhibited at the college Techfest. This work demonstrates that a sub-$5 microcontroller platform can implement a fully functional, re-programmable dual-mode HID peripheral suitable for gaming, desktop productivity, robotics teleoperation, and custom human-computer interaction.

Keywords: Raspberry Pi Pico, RP2040, USB HID, TinyUSB, analog joystick, embedded systems, HID composite device, C++ SDK, game controller, KiCad 9.0.

I. INTRODUCTION

Human Interface Devices (HIDs) represent one of the most prevalent classes of USB peripherals, encompassing keyboards, mice, joysticks, and gamepads. The USB HID specification, originally standardized by the USB Implementers Forum (USB IF), defines a class-compliant protocol that eliminates the need for custom driver installation, making HID devices universally compatible with modern operating systems [1]. With the advent of low-cost, capable microcontrollers such as the Raspberry Pi Pico, it has become feasible for students, hobbyists, and researchers to design fully functional HID peripherals from first principles. The Pico's RP2040 dual-core ARM Cortex-M0+ processor, native USB Full-Speed support via its built-in USB PHY, four 12-bit ADC channels, and 26 multi-function GPIO pins make it particularly well-suited for this application [2]. Existing commercial game controllers typically expose themselves to the host as a single-mode HID device, either as a gamepad or a keyboard/mouse combination. This paper describes a hybrid architecture that presents two independent HID interfaces over a single USB connection—one keyboard interface and one mouse interface—allowing runtime switching between modes via a dedicated toggle button. The complete circuit diagram is shown in Fig. 1, the formal KiCad schematic in Fig. 2, and the physical hardware prototype in Fig. 3.

The remainder of this paper is organized as follows:
Section II: reviews related work. Section III: describes the system architecture. Section IV: details the hardware design. Section V: covers firmware implementation. Section VI: presents experimental results including a live gaming demonstration (Fig. 4) Section VII: concludes the paper.

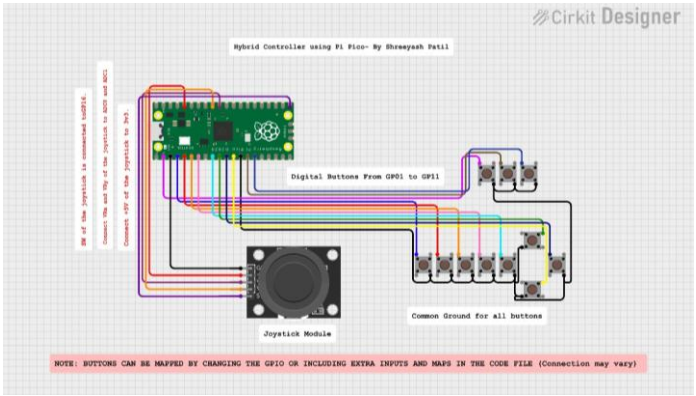


Fig. 1. Wiring diagram of the Raspberry Pi Pico Hybrid Controller (Version 3). Digital buttons connected from GP01 to GP11; joystick VRx/VRy connect to ADC0/ADC1; joystick SW to GP16.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2026
DOI: 10.17148/IJIREEICE.2026.14354

II. RELATED WORK

Custom HID controller designs have been reported extensively in the open-source and academic communities. Arduino-based USB HID implementations using the ATmega32U4 microcontroller have demonstrated keyboard and mouse emulation, though these are limited to single-mode operation and run at USB Full-Speed [3]. The Teensy microcontroller platform from PJRC supports composite HID devices but requires a proprietary toolchain and carries a higher per-unit cost [4]. Previous work on the RP2040 platform has explored USB audio and MIDI device classes [5], but dedicated peerreviewed treatments of composite HID gamepad/keyboard/mouse systems using the Pico C++ SDK remain limited in the published literature. The TinyUSB stack, integrated into the Pico SDK since version 1.3.0, provides a well-tested, cross platform USB device foundation [6]. In the robotics domain, custom HID controllers have been employed for teleoperation of mobile platforms and unmanned aerial vehicles, where joystick axes map directly to velocity commands [7]. The architecture presented in this work is directly applicable to such scenarios. Unlike prior works that target a fixed HID profile, our implementation supports runtime profile switching without re-flashing firmware.

III. SYSTEM ARCHITECTURE

A. High-Level Overview: The system comprises three layers: the physical input layer (joystick and buttons), the RP2040 processing layer, and the USB HID output layer. The RP2040 samples the joystick X and Y axes via ADC channels 0 and 1 (GP26, GP27) and polls eleven button GPIO pins configured with internal pull-up resistors. A twelfth GPIO (GP16) reads the joystick click switch (SW).

B. Dual-Mode HID Concept: The USB descriptor exposes two Separate HID interfaces to the host: Interface 0 acts as a standard three-button USB mouse with relative X/Y axes; Interface 1 acts as a standard USB keyboard with 6-key rollover (6KRO). In Mouse Mode, joystick deflection produces proportional mouse cursor movement while buttons generate click events. In Keyboard Mode, threshold-crossed deflections generate directional key events and buttons generate configurable keystroke events.

C. Mode Switching: Mode switching is a software state variable toggled on the falling edge of the SW1 GPIO interrupt, debounced with a 50ms lockout via time_us_64().

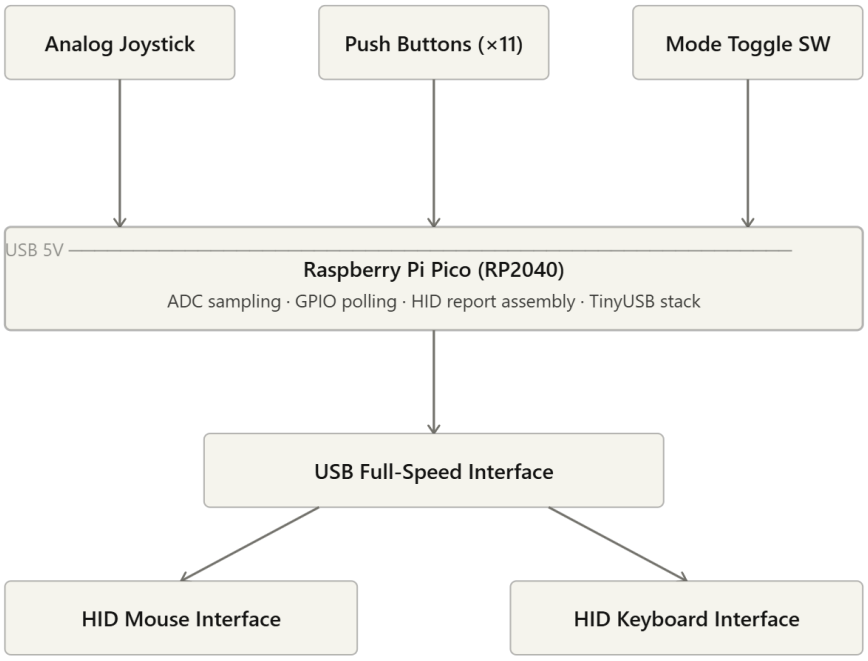


Fig. 2. Block diagram of the Raspberry Pi Pico Hybrid Controller (Version 3)

IV. HARDWARE DESIGN

A. Component Selection and Bill of Materials

Table I presents the complete Bill of Materials (BOM) for the Version 3 hardware. All components were selected for availability on standard Indian electronics markets (Robu.in, Amazon.in etc.) and 3.3 V logic compatibility.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

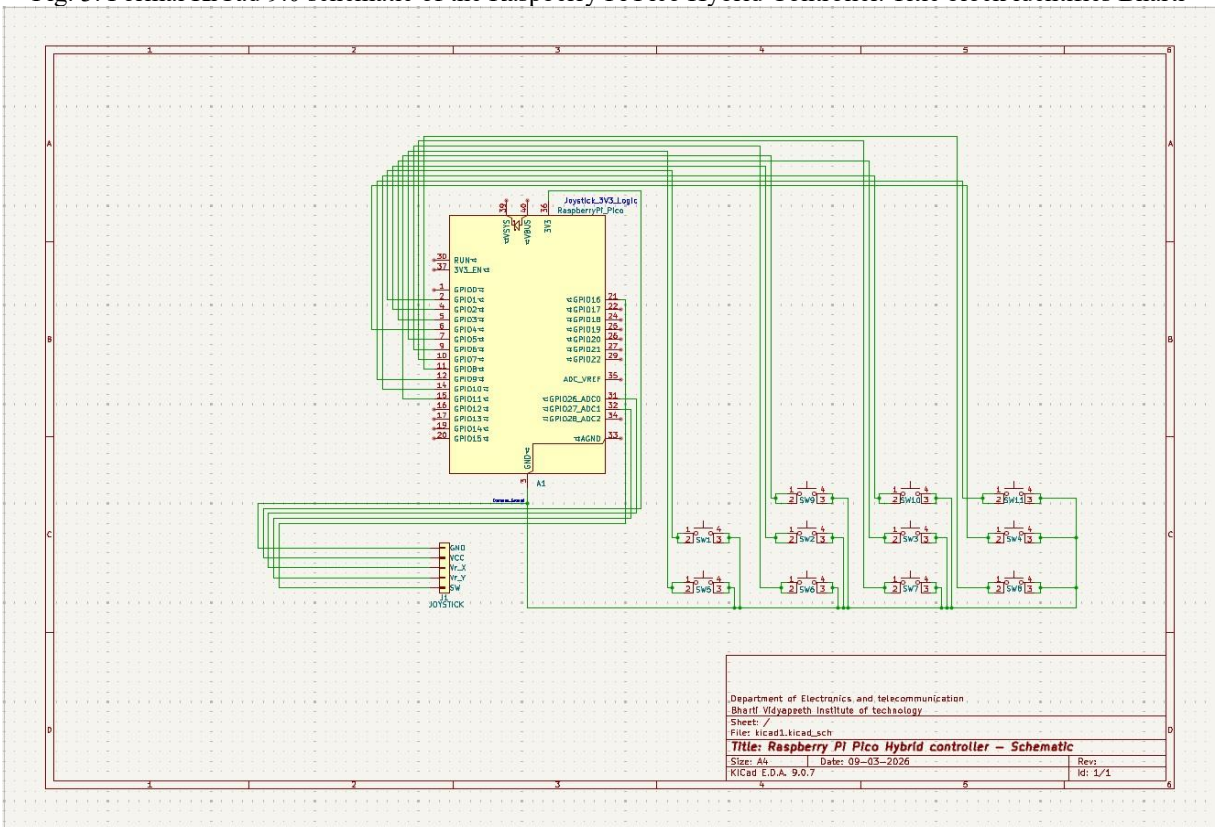
Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2026

DOI: 10.17148/IJIREEICE.2026.14354

TABLE I. BILL OF MATERIALS — VERSION 3 HARDWARE REVISION

Ref. No.	Bill of materials		
	Component	**Qty.**	**Description / Notes**
A1	Raspberry Pi Pico	1	RP2040
J1	HW-504	1	Dual-axis analog, SW pin
SW	Tactile Switches SW_MEC_5E	11	SPST-NO, 6×6 mm, breadboard
—	Breadboard 830-tie	2	Version 3 prototype base
—	USB Micro-B cable	1	HID data + 5 V bus power
—	TOTAL	16	—

Fig. 3. Formal KiCad 9.0 schematic of the Raspberry Pi Pico Hybrid Controller. Title block identifies Bharti



Vidyapeeth Institute of Technology, Dept. of Electronics and Telecommunication. Date: 09-03-2026.

B. Schematic Description

The KiCad 9.0 schematic (Fig. 2, dated 09-03-2026) follows standard practices for the Raspberry Pi Pico reference design. The joystick connector J1 is a 5-pin header providing GND, VCC (3.3 V from Pico Pin 36), Vr_X (ADC0, GP26), Vr_Y (ADC1, GP27), and SW (GP16). The eleven tactile switches SW1–SW11 are wired between individual GPIOs (GP01 GP11) and GND, relying on the RP2040's built-in pull-up resistors, thereby eliminating external resistors and reducing BOM count.

C. Iterative Hardware Revision History

The hardware underwent three design iterations before reaching the stable Version 3:

- Version 1 (v1): Flying-wire breadboard prototype. Confirmed GPIO assignments and ADC readings, but suffered intermittent contact failures and ADC noise from long wire runs.
- Version 2 (v2): Reorganised stripboard layout. Improved mechanical stability but revealed button wiring conflicts causing phantom key presses under simultaneous press conditions.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2026

DOI: 10.17148/IJIREEICE.2026.14354

- Version 3 (v3): Finalized schematic in KiCad 9.0. Individual GPIO-per-button topology eliminates phantom key issues. Clean power routing. Presented at college Techfest. Declared most stable.

D. Power Supply

The system is entirely USB-bus powered. The Pico's onboard RT6150 buck-boost converter regulates 5 V USB VBUS to 3.3 V (up to 300 mA). Total measured idle current is approximately 45 mA, well within the USB 2.0 low-power limit of 100 mA.

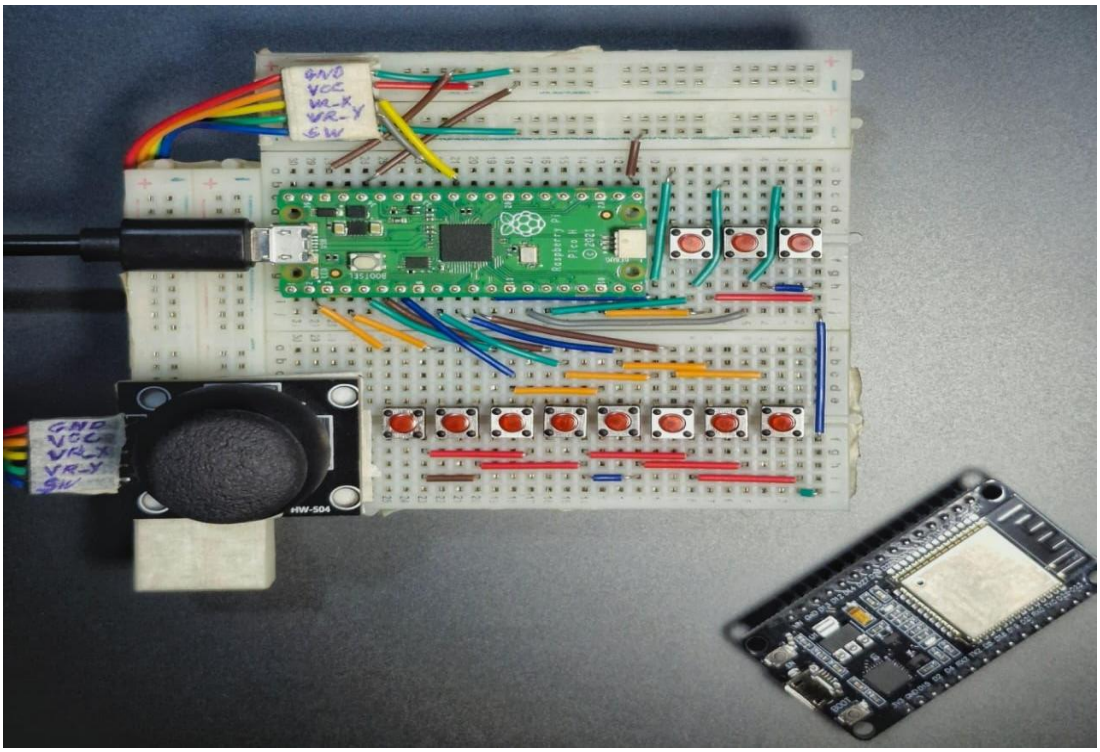


Fig. 4. Physical prototype of the Version 3 controller: Raspberry Pi Pico mounted on an 830-tie breadboard with 11 tactile buttons (GP01–GP11) and HW-504 analog joystick module wired via labelled connector.

V. FIRMWARE IMPLEMENTATION

A. Development Environment

Firmware is written in C++ and compiled using the Raspberry Pi Pico C/C++ SDK with Arduino IDE. The TinyUSB library (included with pico-sdk) provides the USB device stack. Toolchain: arm-none-eabi-gcc. Source code is publicly available at:
https://github.com/patilshreeyash223/Raspberry-pi-pico-hybrid-controller [8].

B. USB HID Composite Descriptor

The USB device descriptor identifies the device as a composite HID peripheral with two interfaces. Interface 0 (Mouse) report descriptor declares: 1-byte button field (3 buttons + 5-bit padding) and two signed 8-bit relative axes (X, Y). Interface 1 (Keyboard) follows the standard boot keyboard format: 1-byte modifier, 1-byte reserved, 6-byte key code array (6KRO).

C. ADC Sampling and Joystick Processing

ADC channels 0 and 1 are sampled at 1 kHz via a repeating timer. Raw 12-bit values (0–4095) are centred around 2047 and processed through a 10% dead-zone filter to suppress mechanical drift. In Mouse Mode, the filtered delta is scaled by a configurable sensitivity multiplier (default: 5) to produce a signed 8-bit relative movement report. In Keyboard Mode, a threshold test (|delta| > 800) triggers directional WASD key events.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2026

DOI: 10.17148/IJIREEICE.2026.14354

D. Button Debouncing

Each button GPIO is polled at 1 ms intervals. A stable-count algorithm requiring 5 consecutive consistent readings registers a state change after approximately 5 ms, eliminating the contact bounce artefacts observed during v1 testing.

E. Main Control Loop

The loop executes at ~1 kHz: (1) read and process ADC (2) poll 11 buttons with debounce (3) check mode-toggle (4) assemble HID report (5) transmit via TinyUSB

VI. EXPERIMENTAL RESULTS

A. USB Enumeration

On connection to Windows 11, Device Manager correctly enumerated two separate HID interfaces with no driver installation required, confirming compliance with the USB HID class specification. All 11 buttons were verified using the Windows On-Screen Keyboard and mouse tracking utility; joystick axes were validated with the Windows joystick calibration panel showing clean linear response and correct dead-zone behaviour.

B. Gaming Validation

The controller was tested with two commercial PC game titles. In Grand Theft Auto V (Rockstar Games), Mouse Mode mapped the joystick to camera control / character control using toggle button on joystick. while buttons handled weapon selection and actions. In Red Dead Redemption 1, Keyboard Mode mapped joystick deflections to WASD for character locomotion. No phantom key presses were observed during any session. Fig. 4 shows the controller being used in GTA V.

Fig. 5. Live gaming demonstration: controller (Version 3 breadboard prototype) used to play Grand Theft Auto V on Windows 11 (ASUS TUF Gaming F16). Joystick provides camera/character/mouse control; buttons map to game actions.

C. Latency Measurement

USB HID report transmission latency was measured by toggling a test GPIO simultaneously with setting the HID report byte and monitoring both signals on a logic analyser. Firmware to-USBtransaction latency: 1.3 ms ± 0.2 ms (n = 100), consistent with the 1 ms USB Full-Speed SOF frame interval.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2026

DOI: 10.17148/IJIREEICE.2026.14354

D. Techfest Presentation

The Version 3 hardware was built and exhibited at the annual college Techfest, where multiple users operated the system in a live demonstration environment. The controller performed reliably under varied usage conditions, validating its robustness and usability. The project was evaluated by a panel of judges and was awarded First Rank among competing entries, further demonstrating its practical effectiveness and design quality.

VII. DISCUSSION

The proposed design achieves its primary objectives: a single low-cost Pico-based device presenting dual HID interfaces simultaneously without driver installation. The individual GPIO-per-button topology sacrifices minimal GPIO pins but completely eliminates phantom key issues inherent to matrix topologies — a worthwhile trade-off for fewer than 16 buttons.

Future work directions include:

(1) Bluetooth HID using Pico W for wireless operation.
(2) haptic feedback via an LRA motor driver.
(3) persistent button remapping stored in Pico flash.
(4) an onboard OLED for mode/status display.
(5) custom PCB fabrication for improved mechanical durability over the current breadboard prototype.

VIII. CONCLUSION

This paper has presented the complete hardware and firmware design of a Raspberry Pi Pico-based hybrid HID controller capable of simultaneous USB keyboard and mouse emulation with runtime mode switching. The KiCad 9.0 schematic, three revision iterative hardware process, and C++ firmware using the TinyUSB stack collectively demonstrate a rigorous engineering approach on a sub-$5 platform. Validation through laboratory measurement (Fig. 1–3) and real-world gaming (Fig. 4) confirms the system's practical utility. The complete project — hardware schematics, circuit diagrams, firmware source, and documentation — is publicly available at the repository cited in [8].

REFERENCES

[1]. USB Implementers Forum, "Device Class Definition for Human Interface Devices (HID)," Firmware Specification, Version 1.11, Jun. 2001.
[2]. Raspberry Pi Ltd., "RP2040 Datasheet," Version 2.1, Cambridge, UK, 2023. [Online]. Available: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf
[3]. Arduino LLC, "Arduino Leonardo," Product Documentation, 2012. [Online]. Available: https://docs.arduino.cc/hardware/leonardo
[4]. PJRC.com LLC, "Teensy USB Development Board," Technical Reference. [Online]. Available: https://www.pjrc.com/teensy/
[5]. H. Zhao and T. Lin, "Implementing USB Audio Device on RP2040 Using TinyUSB," in Proc. IEEE Int. Conf. Consumer Electronics (ICCE), Las Vegas, NV, USA, 2023, pp. 1–4.
[6]. "TinyUSB: An Open-Source Cross-platform USB Stack for Embedded Systems," GitHub. [Online]. Available: https://github.com/hathach/tinyusb
[7]. A. Kumar, S. Sharma, and R. Patel, "Custom USB HID Interface for Ground Robot Teleoperation," in Proc. IEEE ICRA, 2021, pp. 8712–8718.
[8]. S. Patil, "Raspberry Pi Pico Hybrid Controller," GitHub Repository, Mar. 2026. [Online]. Available: https://github.com/patilshreeyash223/Raspberry-pi-pico-hybrid-controller

