Pico: A Human-in-the-Loop Social Robot for AI Coding Agents

Arik Chakma[1], Aong Thwai Wong Marma[1], Sara Jamil[1], Silma Subha[1], and Md Yeasin Sarder[1]

[1] Independent University, Bangladesh, Dhaka, Bangladesh
2210017@iub.edu.bd, 2211021@iub.edu.bd, 2220294@iub.edu.bd, 2220092@iub.edu.bd, 2220168@iub.edu.bd

Abstract. AI coding agents such as Claude Code and Codex edit files and run commands on their own, but they stop and wait when they need permission or an answer from a person. These requests appear in a terminal window that the developer is often not watching, so work stalls, and developers who return in a hurry may approve commands without reading them. This paper presents Pico, a desk robot that gives these requests a physical presence. Agent hooks send each permission request or question to a small Node bridge on the laptop. Pico's ESP32-S3 polls the bridge over Wi-Fi, draws the request and an expressive face on a 2.8-inch screen, plays a spoken Bengali cue and returns the person's choice from four physical buttons. Bengali voice commands, captured by the laptop microphone, map to the same buttons. The agent never depends on Pico: if the robot is offline or nobody answers within 180 s, the agent falls back to its normal terminal prompt. Tests confirmed the complete loop: live Claude Code and Codex requests were answered on the device, a BACK press returned "deny" to the Codex adapter in 2 of 2 logged trials, and a request over Wi-Fi triggered a full Bengali clip. Pico is a research prototype for studying physical supervision of AI agents; no user study has been run yet.

Keywords: Human-robot interaction · AI coding agents · Human-in-the-loop · Tangible interfaces · ESP32 · Bengali voice interaction

1 Introduction

AI coding agents now do real work on a developer's computer. Tools such as Claude Code [1] and Codex [2] read a repository, edit files and run shell commands with little supervision. Research on these agents still measures them mostly by benchmark scores, while the harder problem is how people talk to, steer and trust them [3, 4]. Agents also fail in risky ways when they act without asking, for example by deleting files [5]. For this reason most agents stop before a risky step and ask the person for permission, and they ask questions when a task is unclear. Agents that ask for missing details solve more tasks than agents that guess [6].

These requests create a new problem. The agent shows its question in a terminal or chat window and waits. If the developer is reading documentation,



Fig. 1. The problem Pico addresses. Left: the agent waits for approval in a terminal while the developer looks away. Right: Pico on the desk shows an alert face and a spoken cue, and the developer returns to answer. Concept images generated with an AI image model from our CAD render of Pico; they illustrate the scenario and are not test photographs.

attending a meeting or making tea, the request sits unseen and the task stalls. Developers who check the window again and again lose focus [7], and developers who return in a hurry may approve a command without reading it. Studies of AI-assisted decisions show the same pattern: people over-rely on AI suggestions unless the interface makes them stop and think [8, 9]. Screen-based oversight tools such as Magentic-UI add an "action guard" before irreversible actions [10], and Sidekick adds ambient color cues and spoken summaries for agents working in the background [11]. Both, however, keep the request on the same screen that the person has stopped watching.

Human-robot interaction offers another option. A physically present robot gets more attention and engagement than the same agent shown on video [12], and nonverbal cues such as gaze and expression make a shared task faster and more robust [13]. Commercial desk companions such as Eilik and EMO show that people accept small expressive robots on their desks, but these products cannot show or answer an agent request. Robots that ask for help when they are unsure are an active research topic [14, 15], but this work targets robot planning, not software agents.

Pico connects these two lines of work. It is a small desk robot with a screen, a speaker and four buttons. When a coding agent needs a person, Pico shows the request with an expressive face, plays a short spoken cue in Bengali and accepts the answer by button or by a Bengali voice command (Fig. 1). The answer travels back over Wi-Fi to the waiting agent. If Pico is offline or nobody answers, the agent falls back to its normal terminal prompt, so the robot can never block the agent.

The main contributions of this work are:

- **A physical supervision loop for AI coding agents.** Hooks (commands that the agent runs on events, configured by the user) in Claude Code and Codex send permission requests and questions to a local bridge, and Pico returns the person's decision to the waiting agent.

- **A fail-safe protocol.** The bridge holds each request open, detects an absent device within 5 s and expires unanswered requests after 180 s, so the terminal prompt always remains the fallback.
- **Local-language interaction.** Pico speaks Bengali cues for each event and maps Bengali, Banglish and English voice commands onto its four buttons. Banglish is Bengali written in Latin letters.
- **A prototype built from common parts.** The design uses modules from local shops: an ESP32-S3 board, a 2.8-inch TFT screen, tactile switches, a DFPlayer-type MP3 module and a speaker, in a 3D-printed enclosure.

2 Literature Review and Related Work

We reviewed 51 papers chosen by the five team members. Each paper had to relate to at least one part of Pico: supervising AI agents, coding agents and their users, robots that ask for help, physical and expressive interfaces, or trust and supervisory control. Every entry was checked against its publisher or preprint record before inclusion. Fig. 2 shows how the reviewed papers spread across these areas.

2.1 Oversight and Safety of LLM Agents

Several recent systems add a human checkpoint to large language model (LLM) agents. ToolEmu imitates tools such as a terminal or a bank with an LLM, so agents can be safety-tested without real damage; a common failure was acting without asking [5]. Magentic-UI lets the user and a web agent plan together and stops before irreversible actions to ask for approval [10]. At the tool level, AgentBound gives each Model Context Protocol (MCP) server an Android-style permission list and a sandbox [16], and MCP-Secure holds sensitive tool calls for approval [17]. SAGE scores agents for deployment readiness and can escalate to a person [18], and VALID describes how people move from approving every

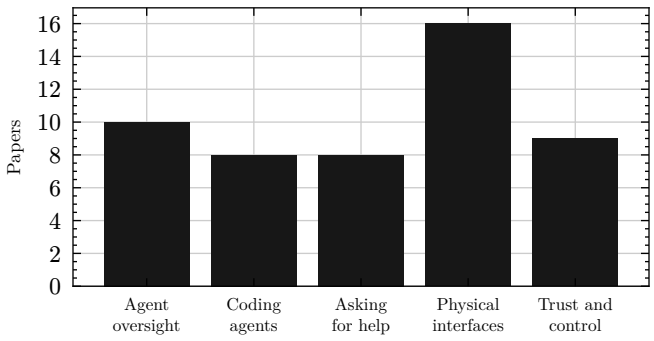

Fig. 2. Distribution of the 51 reviewed papers across the five areas used in this review. Each paper is counted once, in its main area; the counts describe our selection, not research output in each field.

step (human-in-the-loop) to supervising (human-on-the-loop) [19, 20]. Reviews of human-in-the-loop AI and learning summarize these patterns across fields [21, 22], and Freedman and Zilberstein argue that interaction design must address safety as well as user experience [23]. All of these systems decide *when* to ask a person. Pico addresses a different part of the loop: *where* the question appears and how quickly a person can answer it.

2.2 Coding Agents and Their Users

HULA, an agent inside Jira, lets engineers review and edit an AI plan and then the AI's code [4]. Wang et al. argue that humans are missing from coding-agent research [3], and Ambig-SWE shows that agents which can ask a user about an under-specified issue resolve more tasks [6]. Proactive coding assistants must balance helpful interruptions against distraction [24], and Sidekick shows that ambient cues help people multitask while a computer-use agent works [11]. Studies of agent pull requests and agentic refactoring show how much code these agents now produce [25, 26], and tool-integrated agents such as CodeAgent act inside real repositories [27]. Together, this work shows that coding agents increasingly act in real repositories, which creates more decisions for a developer to supervise.

2.3 Robots That Ask for Help

KnowNo turns each planning step of an LLM robot into a multiple-choice question and asks a person only when more than one option remains [14]. Igbinedion and Karaman train a navigation robot to decide when to hand control to a person [15], and Bıyık et al. choose questions that people can answer easily and confidently [28]. Sensible Agent lets an augmented reality (AR) agent ask yes/no or multiple-choice questions that users answer with a nod or gesture [29], and Scherf et al. detect when a person is unsure of a two-choice answer [30]. Studies of proactive robots show that confirmation before acting affects trust [31], and that people want explanations after failures and surprises [32, 33]. Pico applies the same short, option-based question format, but to software agents and with physical buttons.

2.4 Physical, Tangible and Expressive Interfaces

Tangible Bits introduced physical controls and ambient displays at the edge of attention [34], and later reviews map tangible interfaces and data physicalization [35, 36]. Tangible explainable AI and tangible prompting move AI explanations and LLM input into physical objects [37, 38]. A robot's physical presence improves engagement over a video agent [12], and nonverbal cues improve teamwork [13]. Expressive behavior can be generated with LLMs [39], designed through sound [40] or produced by soft actuators [41]. A simple robotic object can increase passengers' sense of safety in a self-driving car [42], and physical AI "cohabitants" such as the Stochastic Parrot run a voice LLM in a robot body [43]. Desk robots already support students: the open-source Robot Study Companion combines a Raspberry Pi, microphones, a speaker, a button and a touch screen [44], and

Blossom accompanied students with ADHD during schoolwork [45]. FurChat shows an embodied LLM receptionist with facial expressions [46], and Lopez et al. design an ESP32 board for a social robot [47]. None of these devices is connected to the approval requests of an AI agent.

2.5 Trust and Supervisory Control

Asking a person to press a button before a robot acts changes how much they trust it [48]. Trust should match what a system can really do, and interventions that force people to think reduce over-reliance [8, 9]. Amershi et al. give 18 guidelines for human-AI interaction, including making clear what the system can do and supporting efficient correction [49]. Research on supervising many robots shows the cost of workload and lost situation awareness [50], JAHRVIS supervises human-robot collaboration at a high level [51], and the level of robot autonomy changes how people accept it [52]. End-user programming studies show a trade-off between user control and perceived robot agency [53, 54]. These findings guide two Pico choices: answers need a deliberate button press, and the robot never replaces the terminal; it only adds a channel.

2.6 Comparative Analysis

Table 1 compares Pico with representative systems. Most agent-oversight systems are screen-based, and most desk robots are not connected to AI agents. Pico combines an agent approval workflow with a physical, expressive and local-language device.

3 Methodology

Pico was built with an iterative, module-first method (Fig. 3). Each part (the screen, the buttons, the MP3 module, the bridge protocol, the hooks and the firmware) was tested on its own before it was connected to the rest. A fault in one module therefore did not hide faults in others, and each test result could be traced to one part. Hardware was first wired on two breadboards with a USB-

Table 1. Comparison of selected related systems with Pico.

Study	**Focus**	**Interface**	**Limitation for Pico's goal**
Ruan et al. [5]	Agent risk testing	LLM-emulated tool sandbox	Offline testing only
Mozannar et al. [10]	Web-agent oversight	Action guard in a web UI	Approval stays on screen
Chang et al. (2026) [11]	Multitasking with agents	Ambient color, summaries, speech	Software only
Takerngsaksiri et al. [4]	Coding agent in Jira	Plan and code review	Review at fixed stages
Ren et al. [14]	LLM robot planning	Multiple-choice help requests	Robot tasks, not code
Baksh et al. [44]	Study desk robot	Pi, mic, screen, button	Not linked to AI agents
Chang et al. (2025) [43]	Physical AI companion	Voice LLM robot parrot	No approval workflow
Pico	**Coding-agent supervision**	**ESP32 robot: faces, Bengali cues, buttons, voice**	**No user study yet**

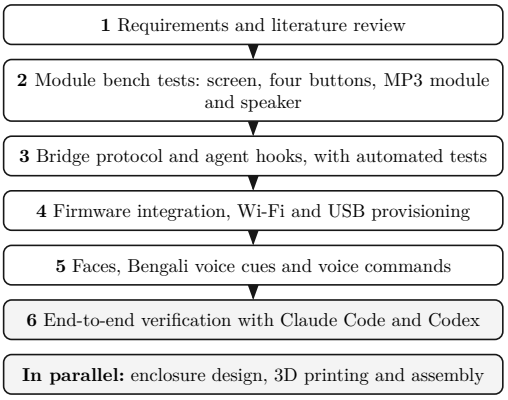

Fig. 3. Development method. Each module was tested on its own before integration.

Table 2. Pico hardware components and pin assignment.

Component	Interface	ESP32-S3 pins	Role
ESP32-S3-WROOM-1 N16R8	Wi-Fi	—	Control, networking
2.8-inch TFT, 240 × 320	SPI, 1 MHz	CS 10, DC 9, RST 8, MOSI 11, SCK 12, MISO 13	Requests and faces
Four tactile switches	GPIO, pull-up	UP 4, DOWN 5, OK 15, BACK 16	Answers
MP3-TF-16P with microSD	UART	GPIO17 to MP3 RX, MP3 TX to GPIO18	Bengali cues
Speaker	SPK1, SPK2	Through MP3 module	Audio output
Microphone module	OUT signal	3V3, GND, OUT	Failed; laptop mic used
Power	USB	3V3 rail	Laptop USB

powered ESP32-S3 board. In parallel, the enclosure was designed in Blender and Python, 3D-printed and assembled around the breadboard build.

3.1 Hardware

The controller is an ESP32-S3-WROOM-1 development board with 16 MB flash and 8 MB PSRAM (N16R8) [55]. It drives a 2.8-inch 240 × 320 thin-film-transistor (TFT) screen over the SPI serial bus at 1 MHz, reads four tactile switches on general-purpose input pins (GPIO) with internal pull-ups and controls an MP3-TF-16P (DFPlayer type) module [56] over a UART serial link. The MP3 module plays Bengali clips from a microSD card through a small speaker. A microphone module was also wired for voice input, but it stopped working during testing, so voice commands are now captured by the laptop's microphone. Table 2 lists the components and pins, and Fig. 4 shows the breadboard build.

Power needed care. The board's 5 V input header (5VIN) has a diode into the internal 5 V rail, so it does not supply USB power outward unless a jumper is set. Early readings of only 0.3–1.3 V at the MP3 module's supply pin were traced to this. Instead of changing the board jumper, the screen and MP3 module run

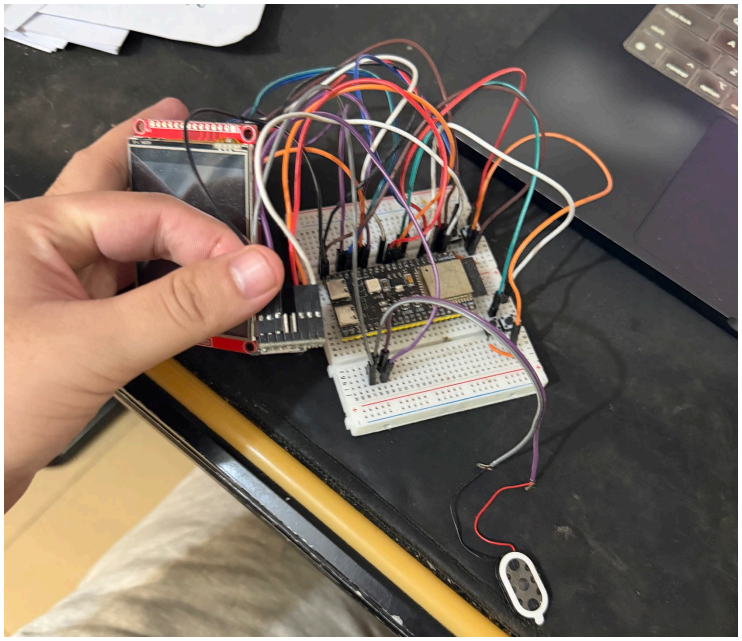

Fig. 4. Breadboard prototype: ESP32-S3 board, 2.8-inch TFT screen (held), jumper wiring, MP3 module and speaker.

from the 3.3 V (3V3) rail, which is inside the DFPlayer's 3.2–5 V supply range. The full wiring is shown in Fig. 5.

3.2 Firmware

The firmware is written in Arduino C++ for the ESP32-S3. A separate task samples the four buttons with a 20 ms debounce, so presses are not lost while the screen redraws. The main loop polls the bridge every 300 ms, redraws the screen every 100 ms and plays cues through the MP3 module. Wi-Fi credentials and access tokens are provisioned over USB rather than written into the source code. The firmware is organized as seven display modes (Fig. 6). The mode is computed each frame from the device state in a fixed order of priority, which keeps the logic small and easy to test.

3.3 Bridge and Agent Hooks

The bridge is a small Node server on the laptop, listening on port 8787. Only two hooks use it [1, 2]. The **Permission** hook handles the `PermissionRequest` event of Claude Code and Codex, and the **Question** hook handles Claude Code's `AskUserQuestion` tool. Each hook posts the request to the bridge and waits. The bridge keeps a queue, serves the oldest open request to Pico and returns the answer to the hook. Every route checks a bearer token (a shared secret sent with

each request): the hooks and the device use different tokens, so the device cannot create requests and the hooks cannot answer them. Requests that Pico cannot draw are rejected at the bridge, and the agent then shows its normal prompt.

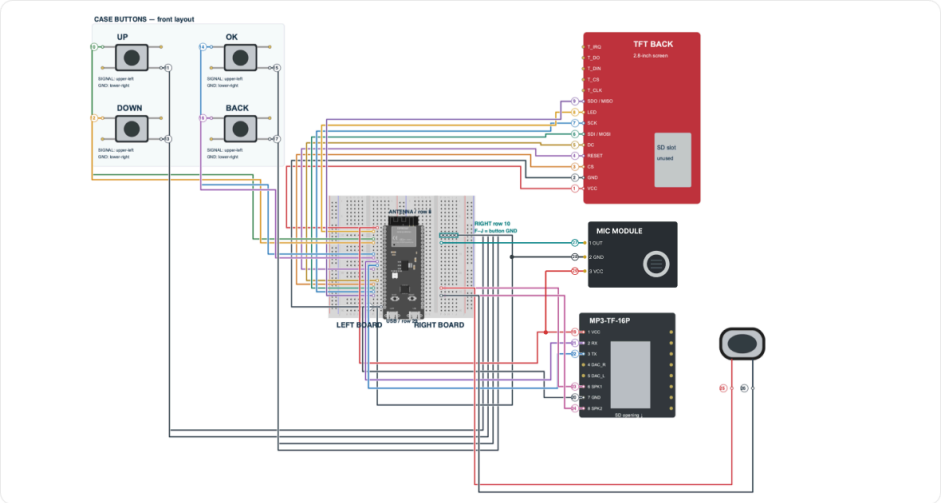

Fig. 5. Wiring of the complete circuit: case buttons, ESP32-S3 on two breadboards, TFT screen (SPI), MP3-TF-16P module (UART) with speaker, and the microphone module (wires 27–29), which later stopped working.

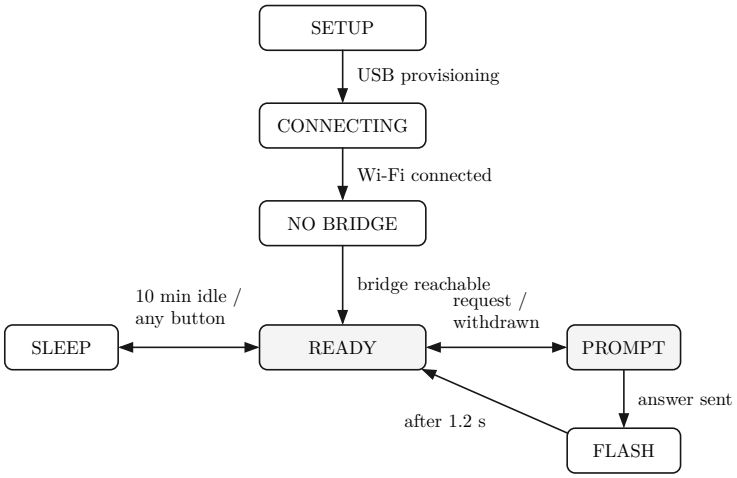

Fig. 6. Firmware display modes. NO BRIDGE (the offline face) means Wi-Fi works but the laptop bridge cannot be reached. If Wi-Fi drops the device returns to CONNECTING.

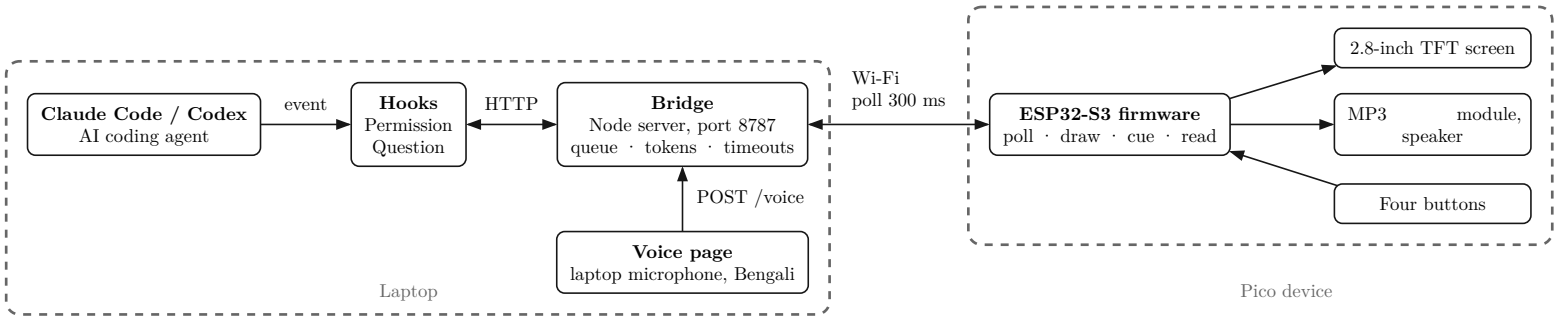

Fig. 7. Overall system architecture. The left group runs on the laptop; the right group is the Pico device. All traffic between them is HTTP over the local Wi-Fi network, started by Pico.

3.4 Development Tools and Testing

The bridge and hooks are written in TypeScript and tested with Node's built-in test runner. The tests cover request validation, the queue, offline detection, an HTTP round trip from hook to device, the Claude Code and Codex output formats, the question flow, voice mapping and the case where the bridge is unreachable. Firmware logic such as debouncing, discarding button presses made before a request appeared, and offline decisions is compiled and tested on the host computer. Physical tests used harmless test requests, so no real command was run during verification.

4 System Design

Pico has three software parts (the agent hooks, the bridge and the firmware) and one physical device (Fig. 7). The coding agent and the bridge run on the developer's laptop; Pico communicates with the bridge over the local Wi-Fi network.

4.1 Request Workflow

Fig. 8 shows one request from start to finish. When Claude Code or Codex needs permission or asks a question, its hook posts the request to the bridge and the connection stays open. Pico polls the bridge every 300 ms. When it receives a request, it shows an arrival face for one second, plays the Bengali cue for the request type and draws the request with its options. The person moves through the options with UP and DOWN, confirms with OK or rejects with BACK. Pico posts the choice to the bridge, which completes the hook's open request, and the agent continues.

Four rules keep the loop safe. **Answer rule:** OK sends the selected option. BACK sends "deny" for a permission request and hands a question back to the terminal. **Offline rule:** if Pico has not polled for 5 s, new requests go straight

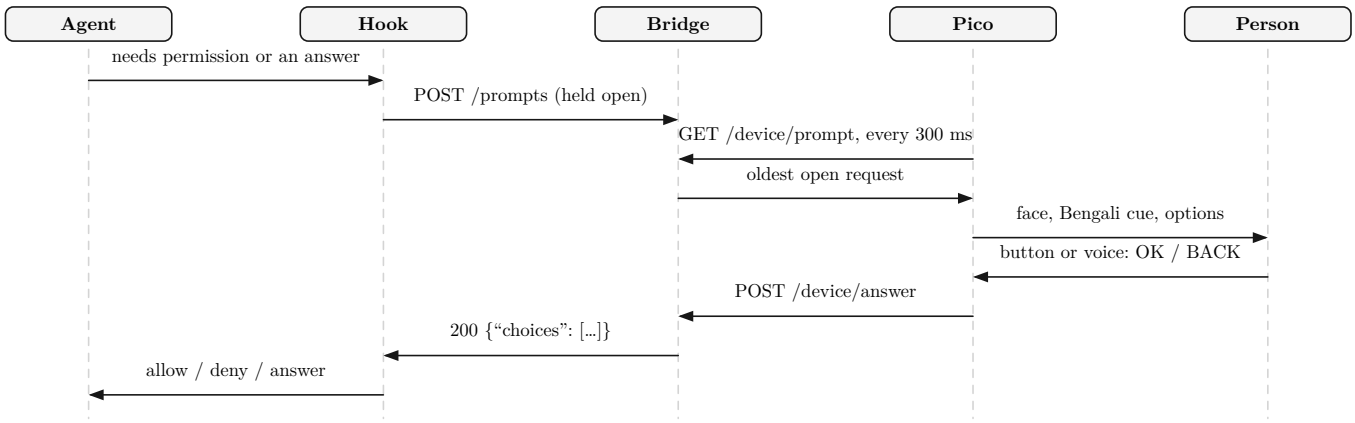

Fig. 8. Request workflow. If Pico has not polled for 5 s the bridge returns 503; if nobody answers within 180 s it returns 408. In both cases the hook prints an empty result and the agent shows its normal terminal prompt.

to the terminal and requests already waiting are released back to it. **Timeout rule:** a request with no answer in 180 s expires and returns to the terminal. **Sleep rule:** after 10 minutes without activity the screen sleeps, and any button wakes it. A button press that wakes the screen is not used as an answer, and presses during the first second of a new request are ignored, so a person cannot approve a request they have not seen. Table 3 lists the timing values.

4.2 Screen Faces and Bengali Voice

Faces give Pico a simple, readable state (Fig. 9). The ready face blinks every few seconds and glances around while connecting. A permission request uses an amber alert face, and a question uses a blue, curious face. When the laptop cannot be reached, a sad face appears with the message “Laptop not reachable. Prompts go to the terminal.” The screen also draws each request with its options and shows the most recent answer on the ready screen.

Table 3. Timing parameters in the firmware and bridge.

Parameter	Value	Purpose
Button debounce	20 ms	Ignore switch bounce
Screen frame	100 ms	Animation and redraw rate
Poll interval	300 ms	Pico asks the bridge for requests
Arrival face	1 s	Block answers until the request is seen
Answer flash	1.2 s	Confirm the answer on screen
Cue length	2 s	Typical Bengali cue; playback is capped at 6 s
Link grace	4 s	Keep the ready face through a missed poll
Device offline	5 s	Bridge sends requests to the terminal
Answer give-up	10 s	Stop resending an answer the bridge did not confirm
Request timeout	180 s	Unanswered request expires
Sleep	10 min	Screen sleeps when idle



Fig. 9. Simplified drawings of Pico's main faces. Permission requests use amber and questions use blue.

Sound reaches the person when they are not looking at the screen. Each event plays a short Bengali clip from the microSD card at volume 30 of 30: track 1 at startup, 3 for a permission request, 4 for a question, 5 when an answer is accepted and 6 when it is rejected. Bengali was chosen because it is the national language of Bangladesh, where Pico is developed and tested. Voice input uses the same four actions as the buttons (Table 4). The bridge's voice parser accepts Bengali, Banglish and English words: for example "haan" (yes) or "ঠিক" means OK, "na" (no) or "বাতিল" means BACK, "opor" means UP and "nich" means DOWN. Because Pico's microphone module stopped working, speech is captured by the laptop microphone on a Chrome page using the Web Speech API [57] with the Bengali (Bangladesh) language setting, which sends the recognized text to the bridge. Because voice is translated into button presses, it follows the same rules and cannot do anything a button cannot.

4.3 Enclosure

The enclosure is a printed desk unit with a sloped screen face and four large keys below it (Fig. 10). Its parts are held with M3 screws and nuts, and the covers are removable so the electronics stay accessible. The design was checked digitally against the ordered modules, then 3D-printed and assembled; it houses the breadboard build used in the tests.

5 Test Data

All tests ran on a macOS laptop with Node 26 and Pico on the same Wi-Fi network, placed on the desk next to the laptop. Team members pressed the buttons. Each test used a harmless request, such as a dummy command or a sample question, so nothing on the laptop was changed. The hardware tests were run on 18 October 2026 and recorded in a verification log; the bridge test suite was run again on the final code while writing this paper. Table 5 lists the recorded values.

Table 4. Examples of voice words that the bridge maps to each button.

Button	**Example words (Bengali, Banglish, English)**
OK	ঠিক, হ্যাঁ, haan, thik, ok, yes
BACK	না, বাতিল, na, batil, back, no
UP	উপর, opor, up, previous
DOWN	নিচ, nich, down, next



Fig. 10. Enclosure design as rendered in Blender before printing: exterior with screen and four keys (left), and the internal layout with screen, key plate and breadboard base (right).

Table 5. Values recorded during prototype testing.

Measured item	Value	Observation
Bridge test suite (current)	11 / 11 pass	Round trip, offline drop, hook formats, voice mapping
Firmware flash size	983,917 B	18 Oct build; upload checked against the compiled file
Static RAM	55,072 B	Same build
Wi-Fi question (n = 1)	"review"	Answered on Pico and returned
Codex adapter, BACK (n = 2)	2 / 2 deny	No command executed
Wi-Fi audio (n = 1)	Volume 30 / 30	Full clip heard; screen stayed on

In the question test, the bridge sent a question and the answer "review" was chosen on Pico and returned with Pico's device name. In the Codex test, a BACK press on Pico returned "deny" to the real Codex hook adapter in both trials, and no command was executed because a harmless test request was used. In the audio test, a request over Wi-Fi played a complete Bengali clip at volume 30 while the screen stayed on; an automatic stop after 6 s guards against a clip that does not end. In an earlier bench test, the button sampler, which runs every 1 ms, showed a largest gap of 1,003 µs during audio playback, with no resets. Table 6 summarizes the functional evaluation.

6 Results and Analysis

The prototype completed the full loop from agent to person and back. A request created on the laptop reached Pico over Wi-Fi, appeared on the screen with a face and a Bengali cue, and the button answer reached the waiting hook. The Codex test is the most important result: a physical BACK press on Pico was returned

Table 6. Functional evaluation of the Pico prototype.

Function evaluated	Result	Observed behavior
Request display	Successful	Question drawn with its options on the TFT screen
Button answer	Successful	Selected answer returned over Wi-Fi
Permission rejection	Successful	BACK returned "deny" to the Codex adapter
Bengali voice cue	Successful	Request over Wi-Fi triggered a full clip at volume 30
Input during audio	Successful	Buttons stayed responsive; no resets
Offline fallback	Tested in software	Requests dropped when polling stops
Voice command mapping	Tested in software	Bengali, Banglish and English words mapped
On-device microphone	Failed	Module stopped working; laptop microphone used
Native agent hooks	Successful	Live Claude Code and Codex requests answered on Pico
User study	Not run	No response-time or trust measurement yet

to a real agent adapter as "deny" in both trials. This shows that the hook output format, the bridge, the network and the firmware work together.

The design keeps the agent independent of the robot. Because the hook holds a normal HTTP request and the bridge answers 503 or 408 when Pico is absent or slow, any failure ends in the agent's own terminal prompt. The bridge tests confirm that requests are released when Pico stops polling and that the hook prints an empty result when the bridge is unreachable, so unplugging Pico sends waiting requests back to the terminal instead of losing them.

The audio results matter for the hardware design. Sound playback did not block button input, because buttons are sampled on their own task. The power investigation found that the 5VIN header does not supply USB power outward on this board, and running the modules from 3V3 gave full playback in our test. Sustained full-volume operation on the shared 3V3 regulator has not been validated.

These results show integration, not effectiveness. With one device and a small number of trials, we cannot report response times, error rates or the effect on developers' attention. Voice input also changed during testing. The microphone module stopped working, so voice now comes from the laptop: a Chrome page recognizes Bengali speech and sends the text to the bridge, which turns it into button presses. The voice parser is covered by the tests, but recognition accuracy has not been measured.

Compared with screen-based oversight tools [10, 11], Pico moves the request into the physical space around the developer and adds sound and a local language. Compared with desk robots and physical AI companions [43, 44], Pico is connected to a real agent workflow. It does not try to decide *when* an agent should ask, as KnowNo and related methods do [14, 15]; it accepts the agent's own request and makes it easier to notice and answer.

7 Discussion

Pico shows that a small robot built from common parts can take part in the control loop of a software agent. Its main strength is the combination: agent

hooks, a fail-safe bridge, an expressive device and local-language interaction work together in one prototype built from common parts. Physical buttons also make approval a deliberate act, which is consistent with evidence that a small physical confirmation changes trust [48] and that forcing a decision step reduces over-reliance [8].

The design has risks. A robot that interrupts too often could become a new distraction, which is the same trade-off found for proactive coding assistants [24]. A short multiple-choice answer is quick, but permission requests can contain long commands that do not fit on a 2.8-inch screen; for these, the bridge rejects the request and the person answers in the terminal. Approving from across the room may also encourage people to approve without reading. These questions need a user study. Security is also limited: the bridge uses plain HTTP with bearer tokens and trusts the local Wi-Fi network, so someone on the same network who captured the device token could answer requests. Encrypted transport (TLS) and a pairing step are needed before use on shared networks.

The current work evaluates functional integration only. No user study, repeated latency measurement, long-term reliability test or speech-recognition accuracy test has been carried out, and battery power has not been built. The contribution should be read as a working research prototype and a platform for future human-robot interaction studies of agent supervision.

8 Conclusion

This paper presented Pico, a desk robot for supervising AI coding agents. Pico receives permission requests and questions from Claude Code and Codex through a local bridge, gets the person's attention with an expressive face and a spoken Bengali cue, and returns a deliberate answer given by button or voice. A fail-safe protocol keeps the terminal prompt as the fallback, so the agent never depends on the robot. Tests confirmed the complete loop with live Claude Code and Codex requests, including a rejection returned to Codex. Pico is a prototype, not a product, but it shows a practical way to move agent supervision off the screen and onto the desk.

9 Future Work

The next steps follow from the limits above.

- **Real-time voice conversation.** Replace the failed microphone module so Pico hears speech itself, and extend single commands into a real-time Bengali conversation with the agent, so a person can ask what a command does before approving it.
- **User study.** Measure response time, missed requests, approval accuracy and trust with developers, comparing Pico with the terminal alone and with on-screen notifications.

- **Battery power.** Add a battery with separate charging and programming ports.
- **More agents and request types.** Support more coding agents, multi-step questions and status updates such as "task finished".
- **Custom PCB.** Replace the breadboards inside the enclosure with a custom printed circuit board for a smaller, more robust device.

10 Team Contributions

Table 7. Team member contributions.

Member	Contribution
Arik Chakma	Project lead; bridge, agent hooks, firmware and website
Aong Thwai Wong Marma	Hardware lead; enclosure design and electronics layout
Sara Jamil	ESP32 firmware for screen, faces, buttons, MP3 playback and Wi-Fi
Silma Subha	Enclosure printing, mechanical assembly and hardware testing
Md Yeasin Sarder	Bridge integration between Claude Code, Codex and Pico

References

1. Anthropic: Claude Code Hooks Reference, https://code.claude.com/docs/en/hooks.
2. OpenAI: Codex Documentation, https://developers.openai.com/codex.
3. Wang, Z.Z., Yang, J., Lieret, K., Tartaglini, A., Chen, V., Wei, Y., Wang, Z., Zhang, L., Narasimhan, K., Schmidt, L., Neubig, G., Fried, D., Yang, D.: Position: Humans are Missing from AI Coding Agent Research. arXiv preprint arXiv:2608.12355. (2026). https://doi.org/10.48550/arXiv.2608.12355.
4. Takerngsaksiri, W., Pasuksmit, J., Thongtanunam, P., Tantithamthavorn, C., Zhang, R., Jiang, F., Li, J., Cook, E., Chen, K., Wu, M.: Human-In-the-Loop Software Development Agents. In: ICSE SEIP 2025, 47th IEEE/ACM International Conference on Software Engineering, Software Engineering in Practice (2025). https://doi.org/10.1109/ICSE-SEIP66354.2025.00036.
5. Ruan, Y., Dong, H., Wang, A., Pitis, S., Zhou, Y., Ba, J., Dubois, Y., Maddison, C.J., Hashimoto, T.: Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In: ICLR 2024 (Spotlight) (2024). https://doi.org/10.48550/arXiv.2309.15817.
6. Vijayvargiya, S., Zhou, X., Yerukola, A., Sap, M., Neubig, G.: Ambig-SWE: Interactive Agents to Overcome Underspecificity in Software Engineering. In: ICLR 2026, International Conference on Learning Representations (2026). https://doi.org/10.48550/arXiv.2502.13069.
7. Mark, G., Gudith, D., Klocke, U.: The Cost of Interrupted Work: More Speed and Stress. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08). pp. 107–110 (2008). https://doi.org/10.1145/1357054.1357072.
8. Buçinca, Z., Malaya, M.B., Gajos, K.Z.: To Trust or to Think: Cognitive Forcing Functions Can Reduce Overreliance on AI in AI-assisted Decision-making. Proceedings of the ACM on Human-Computer Interaction. 5, 188 (2021). https://doi.org/10.1145/3449287.
9. Wischnewski, M., Krämer, N., Müller, E.: Measuring and Understanding Trust Calibrations for Automated Systems: A Survey of the State-Of-The-Art and Future

Directions. In: CHI 2023, ACM Conference on Human Factors in Computing Systems (2023). https://doi.org/10.1145/3544548.3581197.
10. Mozannar, H., Bansal, G., Tan, C., Fourney, A., Dibia, V., Chen, J., Gerrits, J., Payne, T., Maldaner, M.K., Grunde-McLaughlin, M., Zhu, E., Bassman, G., Alber, J., Chang, P., Loynd, R., Niedtner, F., Kamar, E., Murad, M., Hosn, R., Amershi, S.: Magentic-UI: Towards Human-in-the-loop Agentic Systems. arXiv preprint arXiv:2507.22358. (2025). https://doi.org/10.48550/arXiv.2507.22358.
11. Chang, R.-C., Xu, W., Li, D., Wang, B., Guo, A.: Sidekick: Designing Communication for Effective Multitasking with Computer Use Agents. In: UIST 2026, ACM Symposium on User Interface Software and Technology (2026). https://doi.org/10.1145/3830398.3830476.
12. Bainbridge, W.A., Hart, J.W., Kim, E.S., Scassellati, B.: The Benefits of Interactions with Physically Present Robots over Video-Displayed Agents. International Journal of Social Robotics. 3, 41–52 (2011). https://doi.org/10.1007/s12369-010-0082-7.
13. Breazeal, C., Kidd, C.D., Thomaz, A.L., Hoffman, G., Berlin, M.: Effects of Nonverbal Communication on Efficiency and Robustness in Human-Robot Teamwork. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2005).
14. Ren, A.Z., Dixit, A., Bodrova, A., Singh, S., Tu, S., Brown, N., Xu, P., Takayama, L., Xia, F., Varley, J., Xu, Z., Sadigh, D., Zeng, A., Majumdar, A.: Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners. In: CoRL 2023, Conference on Robot Learning (PMLR 229) (2023). https://doi.org/10.48550/arXiv.2307.01928.
15. Igbinedion, I., Karaman, S.: Learning When to Ask for Help: Efficient Interactive Navigation via Implicit Uncertainty Estimation. In: ICRA 2024, IEEE International Conference on Robotics and Automation (2024). https://doi.org/10.1109/ICRA57147.2024.10610965.
16. Bühler, C., Biagiola, M., Grazia, L.D., Salvaneschi, G.: AgentBound: Securing Execution Boundaries of AI Agents. FSE 2026, Proceedings of the ACM on Software Engineering. (2026). https://doi.org/10.1145/3808103.
17. Singh, G., Madisetti, V.K.: MCP-Secure: A Runtime Access Control Layer for Privilege-Aware LLM Agent Tooling. IEEE Open Journal of the Computer Society. (2026). https://doi.org/10.1109/OJCS.2026.3664314.
18. Hamdan, M., Dentamaro, V., Pirlo, G., Cheriet, M.: SAGE: A Stabilize-Assess-Govern-Enforce Framework for Deployment-Oriented Evaluation of LLM-Based Autonomous Agents. IEEE Transactions on Artificial Intelligence (Early Access). (2026). https://doi.org/10.1109/TAI.2026.3731348.
19. Ocasio, J., Mullarkey, M.: From Human-in-the-Loop to Human-on-the-Loop: VALID Framework for Resilient Trust in Autonomous AI Agent Contract Execution. In: DESRIST 2026, Design Science Research in Information Systems and Technology (LNCS 16605) (2026). https://doi.org/10.1007/978-3-032-28316-0_17.
20. Cheruvu, R.: Human-in/on-the-Loop Design for Human Controllability, (2026). https://doi.org/10.1007/978-981-95-3658-0_75.
21. Lazaros, K., Vrahatis, A.G., Kotsiantis, S.: Human-in-the-Loop Artificial Intelligence: A Systematic Review of Concepts, Methods, and Applications. Entropy 28(4), MDPI. (2026). https://doi.org/10.3390/e28040377.
22. Kumar, S., Datta, S., Singh, V., Datta, D., Singh, S.K., Sharma, R.: Applications, Challenges, and Future Directions of Human-in-the-Loop Learning. IEEE Access. (2024). https://doi.org/10.1109/ACCESS.2024.3401547.

23. Freedman, R.G., Zilberstein, S.: Safety in AI-HRI: Challenges Complementing User Experience Quality. In: AAAI 2016 Fall Symposium on Artificial Intelligence for Human-Robot Interaction (Technical Report FS-16-01) (2016).
24. Chen, V., Zhu, A., Zhao, S., Mozannar, H., Sontag, D., Talwalkar, A.: Need Help? Designing Proactive AI Assistants for Programming. In: CHI 2025, ACM Conference on Human Factors in Computing Systems (2025). https://doi.org/10.1145/3706598.3714002.
25. Njoku, A.O., Sharafi, Z., Khomh, F.: When Code Authors Are Agents: A Large-Scale Study of Human–Agent Collaboration in Pull Requests. In: AIware 2026, 3rd ACM International Conference on AI-Powered Software (2026). https://doi.org/10.1145/3805760.3814909.
26. Horikawa, K., Li, H., Kashiwa, Y., Adams, B., Iida, H., Hassan, A.E.: Agentic Refactoring: An Empirical Study of AI Coding Agents. arXiv preprint arXiv:2511.04824. (2025). https://doi.org/10.48550/arXiv.2511.04824.
27. Zhang, K., Li, J., Li, G., Shi, X., Jin, Z.: CodeAgent: Enhancing Code Generation with Tool-Integrated Agent Systems for Real-World Repo-level Coding Challenges. In: ACL 2024, 62nd Annual Meeting of the Association for Computational Linguistics (2024). https://doi.org/10.18653/v1/2024.acl-long.737.
28. Bıyık, E., Palan, M., Landolfi, N.C., Losey, D.P., Sadigh, D.: Asking Easy Questions: A User-Friendly Approach to Active Reward Learning. In: Proceedings of the Conference on Robot Learning (CoRL 2019), PMLR 100. pp. 1177–1190 (2020).
29. Lee, G., Xia, M., Numan, N., Qian, X., Li, D., Chen, Y., Kulshrestha, A., Chatterjee, I., Zhang, Y., Manocha, D., Kim, D., Du, R.: Sensible Agent: A Framework for Unobtrusive Interaction with Proactive AR Agents. In: UIST 2025, ACM Symposium on User Interface Software and Technology (2025). https://doi.org/10.1145/3746059.3747748.
30. Scherf, L., Gasche, L.A., Chemangui, E., Koert, D.: Are You Sure? – Multi-Modal Human Decision Uncertainty Detection in Human-Robot Interaction. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634926.
31. Smith, R., Dragone, M.: Exploring the Impact of Confirmation and Interaction During Human-Robot Collaboration with a Proactive Robot Assistant. In: RO-MAN 2024, 33rd IEEE International Conference on Robot and Human Interactive Communication (2024). https://doi.org/10.1109/RO-MAN60168.2024.10731366.
32. LeMasurier, G., Gautam, A., Han, Z., Crandall, J.W., Yanco, H.A.: Reactive or Proactive? How Robots Should Explain Failures. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634963.
33. Wachowiak, L., Fenn, A., Kamran, H., Coles, A., Celiktutan, O., Canal, G.: When Do People Want an Explanation from a Robot?. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634990.
34. Ishii, H., Ullmer, B.: Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97). pp. 234–241 (1997). https://doi.org/10.1145/258549.258715.
35. Krestanova, A., Cerny, M., Augustynek, M.: Review: Development and Technical Design of Tangible User Interfaces in Wide-Field Areas of Application. Sensors (MDPI). (2021). https://doi.org/10.3390/s21134258.

36. Bae, S.S., Zheng, C., West, M.E., Do, E.Y.-L., Huron, S., Szafir, D.A.: Making Data Tangible: A Cross-disciplinary Design Space for Data Physicalization. In: CHI 2022, ACM Conference on Human Factors in Computing Systems (2022). https://doi.org/10.1145/3491102.3501939.
37. Colley, A., Väänänen, K., Häkkilä, J.: Tangible Explainable AI – an Initial Conceptual Framework. In: MUM 2022, 21st International Conference on Mobile and Ubiquitous Multimedia (2022). https://doi.org/10.1145/3568444.3568456.
38. Li, Y.L., Kuang, Q., Arslan, P.-P., Wang, Y.: Tangible Co-Ideation: Designing Embodied Prompting for Creative Thinking with Large Language Models. In: DIS 2026 Companion, ACM Designing Interactive Systems (Student Design Competition finalist) (2026). https://doi.org/10.1145/3802974.3810894.
39. Mahadevan, K., Chien, J., Brown, N., Xu, Z., Parada, C., Xia, F., Zeng, A., Takayama, L., Sadigh, D.: Generative Expressive Robot Behaviors using Large Language Models. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634999.
40. Wolfe, H., Su, Y., Wang, J.: Dimensional Design of Emotive Sounds for Robots. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634968.
41. Koike, A., Wehner, M., Mutlu, B.: Sprout: Designing Expressivity for Robots Using Fiber-Embedded Actuator. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634983.
42. Kumaran, S.C., Bechor, T., Erel, H.: A Social Approach for Autonomous Vehicles: A Robotic Object to Enhance Passengers' Sense of Safety and Trust. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634998.
43. Chang, E., Kuang, Q., Coelho, M.: The Stochastic Parrot: a physical AI Cohabitant. In: NeurIPS 2025, Creative AI Track (2025).
44. Baksh, F., Zorec, M.B., Kruusamäe, K.: Open-Source Robotic Study Companion with Multimodal Human–Robot Interaction to Improve the Learning Experience of University Students. Applied Sciences (MDPI). (2024). https://doi.org/10.3390/app14135644.
45. O'Connell, A., Banga, A., Ayissi, J., Yaminrafie, N., Ko, E., Le, A., Cislowski, B., Mataric, M.: Design and Evaluation of a Socially Assistive Robot Schoolwork Companion for College Students with ADHD. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634929.
46. Cherakara, N., Varghese, F., Shabana, S., Nelson, N., Karukayil, A., Kulothungan, R., Farhan, M.A., Nesset, B., Moujahid, M., Dinkar, T., Rieser, V., Lemon, O.: FurChat: An Embodied Conversational Agent using LLMs, Combining Open and Closed-Domain Dialogue with Facial Expressions. In: SIGDIAL 2023, 24th Meeting of the Special Interest Group on Discourse and Dialogue (2023). https://doi.org/10.18653/v1/2023.sigdial-1.55.
47. Lopez, N.E., Avila, M.I.C., Montalvo, J.U.V., Villela, V.J.G., Chagolla, F.M.: Design of an ESP32 Compatible Development Board for Mobile Robot Control in Social Robotics. In: LACCEI 2025, 23rd LACCEI International Multi-Conference for Engineering, Education and Technology (2025). https://doi.org/10.18687/LACCEI2025.1.1.2362.
48. Ullman, D., Malle, B.F.: Human-Robot Trust: Just a Button Press Away. In: HRI 2017 Companion, ACM/IEEE International Conference on Human-Robot Interaction (2017). https://doi.org/10.1145/3029798.3038423.

49. Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P.N., Inkpen, K., Teevan, J., Kikin-Gil, R., Horvitz, E.: Guidelines for Human-AI Interaction. In: CHI 2019, ACM Conference on Human Factors in Computing Systems (2019). https://doi.org/10.1145/3290605.3300233.
50. Chen, J.Y.C., Barnes, M.J., Harper-Sciarini, M.: Supervisory Control of Multiple Robots: Human-Performance Issues and User-Interface Design. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 41(4). (2011). https://doi.org/10.1109/TSMCC.2010.2056682.
51. Mayima, A., Clodic, A., Alami, R.: JAHRVIS, a Supervision System for Human-Robot Collaboration. In: RO-MAN 2022, 31st IEEE International Conference on Robot and Human Interactive Communication (2022). https://doi.org/10.1109/RO-MAN53752.2022.9900665.
52. Ye, X., Jo, W., Ali, A., Bhatti, S.C., Esterwood, C., Kassie, H.A., Robert, L.P.: Autonomy Acceptance Model (AAM): The Role of Autonomy and Risk in Security Robot Acceptance. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3635005.
53. Zhang, A.W., Queiroz, R., Sebo, S.: Balancing User Control and Perceived Robot Social Agency through the Design of End-User Robot Programming Interfaces. In: HRI 2025, 20th ACM/IEEE International Conference on Human-Robot Interaction (2025). https://doi.org/10.1109/HRI61500.2025.10974063.
54. Hagenow, M., Senft, E., Radwin, R., Gleicher, M., Zinn, M., Mutlu, B.: A System for Human-Robot Teaming through End-User Programming and Shared Autonomy. In: HRI 2024, 19th ACM/IEEE International Conference on Human-Robot Interaction (2024). https://doi.org/10.1145/3610977.3634965.
55. Espressif Systems: ESP32-S3-WROOM-1 and ESP32-S3-WROOM-1U Datasheet, https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.
56. DFRobot: DFPlayer Mini MP3 Player (SKU DFR0299), https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299.
57. W3C Audio Community Group: Web Speech API, https://webaudio.github.io/web-speech-api/.