HOPLINGS

A swarm of tiny, expressive cube robots that talk in beeps

Concept white paper and build specification, revision 0.1

Project components

- Hopling: a 48 mm 3D-printed cube robot with a face, a camera, a microphone, a speaker and vibration drive
- The Nest: a local hub that runs the shared brain, shared memory and charging mat
- Chirp: the beep-and-whistle language the Hoplings use with each other, readable by both radio and ear
- Roost: the software repository and firmware/hub codebase

September 2026

Contents

1. Executive summary

Hoplings are palm-sized cube robots designed to be watched rather than used. Three of them live together on a desk or floor, skitter and hop using vibration motors, show big animated eyes on a round screen, and talk to one another in a language of beeps and whistles called Chirp. Each one carries a camera and a microphone, so the group learns the room it lives in: which objects sit where, who the people are by voice, and where the good sunny spots are.

The robots stay deliberately small and cheap. Each is built around a single ESP32-S3 module with an integrated camera and microphone, and all heavy thinking runs on a local hub, the Nest. The Nest hosts on-device AI models through the Cactus inference engine, keeps a shared memory that every Hopling can query, and directs group behaviour so the trio acts like a small troupe rather than three separate toys. Nothing leaves the home network.

This document names the system, defines what it can do, specifies hardware and software down to part numbers and repository layout, and lays out a build plan that gets a first robot beeping within a weekend and three coordinated Hoplings running within about eight weeks of part-time effort.

Design goals

- As small as a square body allows while keeping camera, screen, mic, speaker, Wi-Fi and an hour of battery
- Under 60 USD in parts per robot; under 300 USD for the full three-robot system including hub
- Funny to watch: expressive faces, timing, group dynamics and a language with recognisable patterns
- Learns its environment: object memory, voice recognition, and a rough map, all shared across the group
- Fully local: Wi-Fi to the Nest only, no cloud dependency, optional cloud fallback disabled by default
- Hackable: every part is off the shelf, every file is open in a single Git repository

2. The concept

2.1 What a Hopling is

A Hopling is a 48 mm cube with one face. The face is a 1.28 inch round display that shows two large eyes, and above it sits a small camera lens. Inside, an inner sled carrying the electronics floats on four flexible TPU bumpers so the buzzing shell does not rattle the board. Two coin vibration motors bolted to the shell drive a set of angled flexible feet, giving the robot a bristlebot gait that reads as an excited skitter. A short full-power burst of both motors produces a small visible hop.

Hoplings are personalities first and machines second. Each has a name, a distinct voice pitch, a preferred mood and a role in the group that the Nest assigns and rotates. The comedy comes from the same place it does with pets: they react to each other, to people, and to things that appear or vanish in the room.

2.2 What the Nest is

The Nest is a Raspberry Pi 5 (or any small Linux box) that runs three services: a message bus, a memory service, and a director. It also serves as the charging base. Hoplings connect to it over Wi-Fi and never to the internet. Because the Nest does the vision and language work, each Hopling only needs enough compute to animate a face, synthesise beeps, detect a wake word and stream sensor data.

2.3 Why Chirp matters

Chirp is not sound effects layered on top of radio messages; it is the same message rendered two ways. Every message a Hopling sends over the bus has a token sequence, and each token has a fixed sound signature. Humans hear a consistent pattern for greeting, for alarm, for finding a cup, and start recognising it. The robots hear each other too: a Hopling out of Wi-Fi range or muted on the bus can still be understood by another Hopling listening with its microphone. This redundancy is also what makes the group feel alive rather than scripted.

3. Capabilities

3.1 Talking to each other

Hoplings communicate through the Nest message bus and audibly through Chirp. A conversation is a sequence of short exchanges: one Hopling emits a token phrase, the others may answer, and the director can inject prompts to keep the exchange going. Conversations have turn-taking rules to avoid three robots beeping at once, with a small random delay so timing feels natural.

- Greeting and acknowledgement when a Hopling wakes, returns from charging or notices another nearby
- Announcements: found an object, recognised a person, low battery, stuck
- Requests: come here, follow me, look at this, be quiet
- Emotional colour: laugh, grumble, surprise, yawn, appended to any phrase
- Arguments: the director can seed a disagreement about who saw something first; the robots trade increasingly indignant phrases until one sulks

3.2 Following

A Hopling can follow another Hopling or a person. Following a person uses the camera at a low frame rate streamed to the Nest, where a lightweight person detector returns a bounding box; the robot steers to keep the box centred and stops at a comfortable distance. Following another Hopling is easier: the leader publishes its heading and the lead-follower pair keep a short conga line using camera and the leader's distinctive face pattern, which the vision model is trained to detect. Three Hoplings following each other in a loop is a reliable crowd pleaser.

3.3 Exploring

In explore mode a Hopling wanders with a random walk biased toward places it has not visited recently. It stops when the camera sees something new, chirps a found-object phrase, and asks the Nest what the object is. Exploration is cooperative: the director splits the reachable area into zones and assigns each

robot a zone so the group covers the room rather than bunching up. Bump detection comes from the IMU; a sharp deceleration triggers a startled face, a retreat and a grumble.

3.4 Learning the environment

The Nest maintains a shared world model that every Hopling contributes to and reads from:

- **Object memory.** When a Hopling reports a new object, the Nest runs a small vision-language model through Cactus to label it and stores an image embedding, a label, a timestamp and the reporting robot's approximate position. Repeat sightings update the record. Objects that move or disappear generate events the director can turn into drama.
- **People and voices.** A person is enrolled by saying a name to any Hopling. The Nest extracts a speaker embedding from the audio clip and stores it against the name. Later, any Hopling that hears speech sends a short clip to the Nest, gets a name back, and can react, for example by greeting that person by their assigned Chirp name-token.
- **Rough map.** Hoplings have no wheel encoders, so the map is topological rather than metric: a grid of cells over the play area, each with a visit count, a stuck count, a light level and a list of objects seen there. Dead reckoning comes from the IMU and known motor durations, corrected whenever a robot sees a landmark it already knows or is spotted by a fixed Nest camera if one is fitted.
- **Habits.** The director keeps simple statistics such as when people are usually around and which zones are most interesting, and uses them to schedule sleep, patrol and play periods.

3.5 Facial expressions

The face is the primary channel for a human audience. It is rendered locally by the ESP32-S3 at 30 frames per second from a small set of parameters rather than from stored images, so it can blend and transition smoothly. Eyes are two rounded shapes with adjustable size, height, tilt, pupil position, lid position and squash. A mood vector drives the parameters, and events fire short overriding animations.

Expression	Eye parameters	Trigger	Chirp colour
Neutral	Round, centred, slow blink every 3 to 6 s	Idle	None
Curious	One eye larger, pupils shifted toward stimulus	New object, new sound	Rising two-note query
Happy	Squashed upward crescents, faster blink	Recognised person, greeted back	Trill
Surprised	Very large, pupils tiny, no blink for 1 s	Bump, sudden loud sound	Sharp high spike
Grumpy	Lids lowered from top, inward tilt	Lost argument, interrupted	Low descending buzz
Sleepy	Half-closed, slow drift, occasional droop	Low battery, night schedule	Yawn glide
Alarmed	Wide, rapid side-to-side pupils	Picked up, unknown voice at night	Rapid stutter
Smug	One lid half down, slight tilt	Won argument, found object first	Short chuckle burst
Loading	Pupils replaced with orbiting dot	Waiting for Nest response	Soft tick

Expressions also drive the body. Curious pulses one motor gently so the robot rocks toward the stimulus; happy produces a short shuffle; alarmed triggers a retreat; sleepy slowly reduces motor duty to zero.

4. System architecture

The system has two tiers connected over the local Wi-Fi network. A Hopling is a thin client responsible for real-time work: face rendering, sound synthesis, motor control, wake-word detection and sensor streaming. The Nest is the shared brain responsible for perception, memory, language decisions and group direction.

Layer	Runs on	Responsibilities	Latency budget
Reflex	Hopling firmware	Face animation, beep synthesis, motor PWM, bump detection, wake word	Under 20 ms
Perception	Nest	Object detection and labelling, person detection, speaker ID, Chirp audio decoding	100 to 800 ms
Memory	Nest	World model, object and voice database, map grid, event log	Under 50 ms query
Direction	Nest	Role assignment, conversation seeding, scheduling, personality via LLM	1 to 3 s, asynchronous

4.1 Message bus

All communication uses MQTT over the Nest's Mosquitto broker. Topics are namespaced per robot and per function. Payloads are compact JSON for control and memory traffic; camera frames and audio clips are sent as binary MQTT payloads or over a plain HTTP POST to the Nest's ingest endpoint, whichever proves simpler on the ESP32.

```
hoplings/<id>/status        battery, mode, mood, heading, cell   (retained, 1 Hz)
hoplings/<id>/chirp         outgoing Chirp phrase                (event)
hoplings/<id>/event         bump, pickup, wake, object_seen, voice_heard
hoplings/<id>/cmd           move, hop, face, say, mode            (Nest to robot)
hoplings/<id>/frame         JPEG frame, 160x120, on request or 2 fps in explore
hoplings/<id>/audio         16 kHz mono clip, up to 3 s, after wake or loud sound
nest/world/objects          shared object memory updates          (retained)
nest/world/people           recognised person events
nest/director/<id>/role     current role and zone assignment      (retained)
nest/broadcast              all-robot messages
```

4.2 Data flow example: finding a new object

1. Hopling B in explore mode notices high change in its low-resolution frame and stops.
2. It shows the loading face, chirps a soft tick and publishes a 160x120 frame to hoplings/b/frame.
3. The Nest perception service crops the region of interest, computes an image embedding and compares it against object memory. No match above threshold, so it runs the vision-language model through Cactus to label it: a blue mug.

4. The Nest writes the record to the object store, publishes to nest/world/objects and sends Hopling B a cmd with face curious and a Chirp phrase: found, object-token(mug), here.
5. Hopling B renders the face and beeps the phrase. Hoplings A and C hear it on the bus and, if idle, the director may send A a come-here role so it trundles over and the two of them inspect the mug together.

5. Hardware specification

5.1 Hopling (per robot)

Subsystem	Part	Notes	Approx. cost USD
Controller	Seeed XIAO ESP32-S3 Sense	Dual-core 240 MHz, 8 MB PSRAM, Wi-Fi, USB-C, LiPo charger, snap-on OV2640 camera and PDM mic	14
Display	1.28 in round IPS, GC9A01, 240x240, SPI	Alternative: 1.54 in square ST7789 240x240 fills the cube face	5
Audio out	MAX98357A I2S amplifier plus 20 mm 8 ohm speaker	3 W class D, mono, drives beeps well above room noise	5
Motion	2 x 10 mm coin ERM vibration motors, 3 V	Left and right, mounted to shell not sled	2
Motor driver	DRV8833 dual H-bridge breakout	Only one direction used per channel, but allows braking pulses for crisper hops	2
IMU	MPU6050 6-axis, I2C	Bump, pickup, heading estimate, tilt	2
Fuel gauge	MAX17048 breakout, I2C	Battery percentage without using a spare pin	2
Battery	500 mAh 1S LiPo, 36x20x8 mm, with protection	About 60 to 90 min active; charges via XIAO onboard charger	6
Charging contacts	2 x pogo pins or 2 x 6 mm neodymium magnets with wires	Base pads mate with Nest charging mat	2
Feet	Printed TPU fin strip, 8 fins angled 15 degrees	Tune stiffness by print line width	under 1
Shocks	4 x printed TPU bumpers, 12 mm	Hold inner sled; 95A TPU works well	under 1
Shell	PLA or PETG, 2 parts plus inner sled, 48 mm cube	About 40 g filament	1
Misc	Slide switch, wires, M2 screws, heat-set inserts		3
		Total per Hopling	about 43 to 55

Power budget

Typical active draw is about 180 mA at 3.7 V: ESP32-S3 with Wi-Fi about 90 mA average, display about 30 mA, two motors at 50 percent duty about 50 mA, audio bursts a few mA averaged. A 500 mAh cell gives roughly 2.5 hours of mixed use or about 75 minutes of continuous play. The robots return to the mat when the Nest reports them under 20 percent.

Pin allocation on the XIAO ESP32-S3 Sense

Function	Pins	Bus
Display	SCK D8, MOSI D10, CS D1, DC D2; RST tied to EN, backlight tied to 3V3	SPI2
Speaker amp	BCLK D7, LRC D6, DIN D9	I2S0
Motors	Left D0, Right D3 (PWM), DRV8833 nSLEEP tied high	LEDC PWM
IMU and fuel gauge	SDA D4, SCL D5, MPU6050 at 0x68 and MAX17048 at 0x36	I2C
Camera and mic	Onboard via Sense daughterboard (GPIO 10 to 18, 38 to 48)	DVP, PDM

The XIAO exposes eleven general pins once the camera is fitted, and the layout above uses all of them, which is why the display reset and backlight are hard-wired and battery voltage is read over I2C by a MAX17048 fuel gauge rather than an ADC divider. If a conflict appears during bring-up, the fallback is a 24-pin ESP32-S3-WROOM devboard at the cost of about 6 mm more length.

5.2 Mechanical design

- Outer shell: two halves, front and back, joining on a lip with two M2 screws at the rear. The front half carries the display recess, the camera aperture and the speaker grille. The back half carries the switch and USB-C access slot.
- Inner sled: a printed frame holding the XIAO, amp, driver, IMU and battery in a stack, with four corner posts that snap into the TPU bumpers. The display connects by a 40 mm ribbon so the sled can move about 1.5 mm in any direction without straining the cable.
- Motor mounts: two pockets in the shell floor with adhesive pads, one each side, so vibration goes into the shell and feet, not the electronics.
- Feet: one TPU strip glued to a shallow channel in the base; fins angled backward. Print a set at 0.4, 0.6 and 0.8 mm width and test which gives the best forward bias.
- Charging pads: two exposed contacts on the base, 30 mm apart, matching parallel copper tape strips on the Nest mat, so any orientation along the strip charges. Reverse polarity protection with a Schottky diode.

5.3 The Nest

Component	Part	Notes	Approx. cost USD
Compute	Raspberry Pi 5, 8 GB, active cooler	Runs Cactus models, perception, memory and director	95

Component	Part	Notes	Approx. cost USD
Storage	64 GB A2 microSD or NVMe HAT	Object and voice database, event log, model files	12
Optional camera	Pi Camera Module 3 wide, mounted high	Gives a fixed overhead view for position correction	35
Charging mat	2 x 5 V 2 A supply, copper tape strips on 3 mm acrylic	Charges all three Hoplings in parallel	10
Enclosure	Printed housing combining Pi, mat and camera mast		3
		Total	about 120 to 155

The Pi 5 CPU is adequate for the models listed in section 6; adding a Hailo or Coral accelerator HAT roughly halves vision latency but is optional. Any x86 mini PC or an old laptop also works and simply runs the same containers.

6. Software specification

6.1 Hopling firmware

Firmware is written in C++ on ESP-IDF via PlatformIO. It is structured as FreeRTOS tasks pinned to the two cores: core 0 handles Wi-Fi, MQTT and the camera pipeline; core 1 handles the face renderer, audio synthesis and motor control so animation never stutters when the radio is busy.

Module	Purpose	Key libraries
face	Parametric eye renderer at 30 fps with tweening and event overlays	LovyanGFX for GC9A01, custom drawing
chirp	Token to sound synthesis, 16 kHz, sine and square oscillators with pitch envelopes, vibrato and noise bursts	ESP-IDF I2S driver
ears	Wake-word detection, loud-sound trigger, Chirp audio decoding via goertzel filters	Espressif ESP-SR or microWakeWord
motion	Motor PWM patterns: skitter, turn, hop, rock, shuffle; bump and pickup detection from IMU	LEDC, I2C
link	Wi-Fi provisioning, MQTT client, frame and audio upload, OTA updates	esp-mqtt, esp_https_ota
brain	Local state machine: mode, mood vector, reflex reactions when the Nest is unreachable	Custom

6.2 Nest services

The Nest runs as a docker-compose stack of five services so any of them can be swapped or scaled.

Service	Language	Purpose	Models and libraries
broker	Mosquitto	MQTT message bus	
perception	Python	Frame ingest, object detection and embedding, person detection, speaker ID, Chirp audio decode	YOLO-nano or MobileNet-SSD for boxes; SmolVLM-256M via Cactus for labels; ECAPA-TDNN (SpeechBrain) or Resemblyzer for voice; DINO-small embeddings for object matching
memory	Python + SQLite	World model, object and people tables, map grid, event log, REST query API	sqlite-vec for embedding search
director	Python	Roles, zones, conversation seeding, scheduling, personality generation	Needle 2 or Qwen-0.6B via Cactus for personality text, converted to Chirp tokens with a rule table
console	Web (HTML + JS)	Dashboard: live faces, map, object list, voice enrolment, Chirp log with translation	FastAPI backend, plain JS frontend

6.3 Cactus integration

Cactus is used on the Nest as the inference runtime for both the vision-language labelling model and the personality language model. It runs GGUF-quantised models on ARM CPUs efficiently and exposes an OpenAI-compatible local API, which keeps the director and perception code independent of the model choice. Cloud fallback is available in Cactus but is disabled in this project's configuration so the system stays fully local. If a future Hopling revision moves to a board with an NPU, the same Cactus SDK runs the small models on the robot itself, and the Nest simply becomes a memory and coordination service.

6.4 The Chirp language

Chirp is a token language with a fixed vocabulary of about forty tokens. A phrase is one to five tokens. Each token has a sound signature built from pitch (relative to the robot's base pitch, which is its voice), duration, contour (flat, rising, falling, wobble) and timbre (sine, square, noise). Because signatures are relative to the base pitch, three Hoplings say the same phrase in recognisably the same way but in different voices.

Token	Meaning	Sound signature
HI	Greeting	Two rising sine notes, 80 ms each
YO	Acknowledge, yes	Single short square blip
NAH	No, refuse	Two falling square blips
HUH	Query, what is that	Rising sine sweep with vibrato, 200 ms
FOUND	I found something	Trill of four alternating notes
OBJ:<n>	Object class n from shared memory	Three-note melody derived from the object id
WHO:<n>	Person n	Signature jingle assigned at enrolment
HERE	At my location	Flat low sine, 150 ms

Token	Meaning	Sound signature
COME	Come to me	Repeated rising pair, up to three times
FOLLOW	Follow me	Rising sweep then two blips
LOOK	Look at this	Sharp high spike then silence
MINE	Claim, I saw it first	Descending square with wobble
HA	Laugh	Fast stutter of six short sine notes
GRR	Grumble	Low buzzing square, 300 ms
ZZZ	Sleepy, going to charge	Long falling glide with fade
EEK	Alarm	Rapid high noise-and-square stutter
SSH	Be quiet	Breathy noise burst, short
BYE	Leaving	Two falling sine notes

Phrase grammar is simple: an optional opener (HI, HUH), a core (FOUND OBJ:12 HERE), and an optional colour (HA, GRR). The Nest console shows a live translation so people can learn to read the exchanges. A cheat sheet card for the fridge is part of the repository.

6.5 Group behaviour and personality

The director assigns each Hopling one of a small set of roles that rotate on a schedule or on events: Scout explores and reports, Buddy follows a person, Heckler comments on what the others do, Napper charges and yawns. Roles are simple rule sets, and the language model is used only to add flavour: given a compact description of recent events, it chooses which Chirp phrases a Hopling with a particular personality would say next. Keeping the LLM out of the real-time loop makes behaviour predictable and the comedy timing consistent.

Each Hopling has a personality file: base pitch, default mood, curiosity, stubbornness and sociability values from 0 to 1, and a name. The three shipped personalities are Pip (curious, high voice), Bort (grumpy, low voice) and Mo (sleepy, mid voice, laughs easily).

6.6 Privacy and safety

- All audio and video stays on the local network; frames and clips are deleted after processing unless a person explicitly saves them from the console
- Voice enrolment is opt-in and names are stored locally; deleting a person removes their embedding
- Motors are current-limited and the shell has no pinch points; the battery has a protection circuit and charging only happens through the XIAO's onboard charger
- Wake word is required before any speech clip is uploaded; loud-sound triggers upload only a 0.5 s clip for classification, not transcription

7. GitHub repository setup

Everything lives in one monorepo named roost so firmware, hub, hardware and documentation stay versioned together.

7.1 Repository layout

```
roost/
  README.md
  LICENSE                      MIT for code, CERN-OHL-P for hardware
  docs/                        this white paper, build guide, Chirp cheat sheet
  firmware/                    PlatformIO project for the Hopling
    platformio.ini
    src/main.cpp
    src/face/  src/chirp/  src/ears/  src/motion/  src/link/  src/brain/
    include/pins.h             single source of truth for pin allocation
    include/personality.h      generated from personalities/*.yaml
    test/                      unit tests for chirp encoder and face tweening
  nest/                        hub services
    docker-compose.yml
    broker/mosquitto.conf
    perception/  memory/  director/  console/
    models/                    downloaded at first run, gitignored
  shared/
    chirp_tokens.yaml          the vocabulary, used by firmware and Nest
    mqtt_topics.md
    personalities/pip.yaml bort.yaml mo.yaml
  hardware/
    cad/hopling.scad           parametric shell, sled, bumpers, feet
    cad/nest.scad
    stl/                       exported prints
    bom.csv
    wiring.md
  tools/
    flash.sh                   build and flash one robot with its personality
    enroll_voice.py
    chirp_play.py              play any Chirp phrase on your laptop for testing
  .github/workflows/
    firmware.yml               build firmware on every push
    nest.yml                   lint and test the hub services
    cad.yml                    regenerate STLs from SCAD on release tags
```

7.2 First-time setup

```
# clone and enter
git clone https://github.com/<you>/roost.git && cd roost

# Nest: on the Raspberry Pi 5
cd nest
cp .env.example .env           # set WIFI_SSID, MQTT password, play-area size
docker compose up -d           # pulls images, downloads Cactus models on first run
open http://nest.local:8080    # console dashboard
```

```
# Firmware: on your laptop with PlatformIO installed
cd firmware
cp ../shared/personalities/pip.yaml personality.yaml
pio run -t upload               # XIAO in bootloader mode over USB-C
pio device monitor              # first boot prints the Wi-Fi provisioning QR

# Enrol yourself
python tools/enroll_voice.py --name alice   # speak for 10 s to any Hopling
```

7.3 Workflow conventions

- Branches: main is always flashable; feature branches per module, squash-merged with a changelog line
- Versioning: firmware and Nest share a semantic version; the Nest refuses connections from robots more than one minor version behind and offers OTA
- CI: GitHub Actions builds firmware for the XIAO on every push and runs the Chirp encoder tests; Nest services get ruff and pytest; release tags regenerate STLs with OpenSCAD in CI so hardware and code versions match
- Issues: labels for firmware, nest, hardware, chirp and behaviour; a good-first-issue label for new Chirp tokens and new expressions, which are the easiest contributions
- Secrets: Wi-Fi and MQTT credentials live only in .env and in NVS on the robot, never in the repo

8. Build plan

Phase	Weeks	Deliverable	Done when
0. Bench	1	XIAO on a breadboard with display and speaker	Eyes blink and a HI phrase plays from a serial command
1. Voice	1	Chirp synthesiser and token vocabulary	All tokens sound distinct in a blind test with two listeners
2. Nest	1	Broker, memory and console running on the Pi	Two breadboard robots exchange HI and YO over MQTT and beep them
3. Body	2	Printed shell, feet, bumpers, motors, battery	One robot skitters forward, turns and hops on a table without tipping
4. Eyes and ears	1	Frame upload, object labelling, wake word, voice enrolment	Robot chirps FOUND OBJ for a mug and greets an enrolled person
5. Trio	1	Three robots, director roles, following, exploring	Conga line and a seeded argument run for five minutes unattended
6. Polish	1	Charging mat, sleep schedule, cheat sheet, OTA	Robots self-charge overnight and wake with the morning

8.1 Bill of materials for three robots plus Nest

Item	Qty	Unit USD	Total USD
XIAO ESP32-S3 Sense	3	14	42
GC9A01 1.28 in round display	3	5	15
MAX98357A amp and 20 mm speaker	3	5	15
Coin ERM motor 10 mm	6	1	6
DRV8833 breakout	3	2	6
MPU6050	3	2	6
MAX17048 fuel gauge	3	2	6
500 mAh LiPo with protection	3	6	18
Pogo pins or magnets, switch, wire, screws, inserts	3 sets	5	15
PLA, PETG and TPU filament	1	20	20
Raspberry Pi 5 8 GB with cooler and PSU	1	105	105
microSD 64 GB	1	12	12
Charging mat materials	1	10	10
Spares and contingency			30
		Total	about 300

9. Risks and open questions

- **Vibration drive controllability.** Bristlebots are famously unpredictable on carpet and slightly tilted surfaces. Mitigation: design for a smooth mat of about one square metre as the default play area, and treat wandering off it as an event the Nest reacts to rather than a failure.
- **Pin count on the XIAO.** The camera consumes most GPIOs. Bring-up may show a conflict between the display and the IMU. Mitigation: an I2C port expander, or the larger ESP32-S3-WROOM devboard with a 52 mm cube.
- **Wi-Fi power draw.** Continuous frame streaming will halve battery life. Mitigation: frames only on change detection or on request, and modem sleep between publishes.
- **Speaker ID accuracy in a small room with beeping robots.** Mitigation: robots go silent for one second after a wake word so the clip is clean; the Nest rejects low-confidence matches and asks the person to repeat their name.
- **Model latency on the Pi 5 CPU.** Object labelling may take close to a second. This is acceptable because the loading face covers it and, frankly, a robot squinting at a mug for a second is part of the joke. An accelerator HAT is the upgrade path.
- **Does Chirp stay funny?** The vocabulary is small on purpose so humans learn it; the personality model adds variety in which tokens get chosen, not in the tokens themselves. Expect to tune phrase timing and the argument scripts more than anything else.

10. Next steps

6. Order the parts for one Hopling and the Pi 5; the rest of the BOM can wait for phase 3.
7. Create the roost repository from the layout in section 7 and commit chirp_tokens.yaml first, since firmware and Nest both depend on it.
8. Build phase 0 on a breadboard and record the first HI; it makes a good README video.
9. Print the feet strip in three stiffnesses and a test shell before finalising the CAD.
10. Enrol two voices and one mug, and let the first robot loose.