BUILD LOG & GUIDE · ROOMBA OPEN INTERFACE

Turning a Roomba 675 into a camera robot

An older Wi-Fi Roomba has a hidden serial port under its decorative ring. Through it, a computer can drive the robot and read its sensors using iRobot's own documented Open Interface — which makes the Roomba a ready-made robot base for a Raspberry Pi and a camera. This guide covers the whole arc: proving the link with a laptop first, then the Pi rider stage, then free roaming.

It is written as a build log, not a finished recipe. Each stage is labelled with what has actually been verified on the author's own robot, what is in progress, and what is still only a plan.

By **Caitria** · Bay Area · Status as of **1 November 2026**

WHERE THIS BUILD STANDS

● IN PROGRESS

Stage 1 — Tether test

Core link **verified 31 Oct 2026**: commands sent from a Mac were received by the Roomba and sensor data came back — two-way communication is proven. Still open: confirming beep/song playback through the test program (song playback needs Safe mode; a program update that reports the mode from partial replies was in testing).

● PLANNED

Stage 2 — Pi rider + camera

Not built yet. A Raspberry Pi and Pi Camera ride on top, wired to the same port through a real plug, powered independently, streaming video over Wi-Fi.

● PLANNED

Stage 3 — Free roaming

Not built yet. Untethered driving around the house under supervision, with dock-return behaviour worked out.

⚠ Read this first — safety

- **The battery pins are unregulated pack voltage** (roughly 14–20 V). Never connect them to any logic pin, computer, Arduino, or Pi. In this guide they stay empty, always.
- **Identify every hole with a multimeter before connecting anything to it.** Published pin numbering runs in opposite directions in different documents — the meter method in Stage 1 exists because of that.
- **Jumper wires are bench-test only.** They wiggle, and a slipped jumper can short neighbouring holes. The roaming build needs a real plug.
- **First drives happen on open floor, with a hand hovering.** Lifting the robot off the ground is the emergency stop.
- **End every session with the Stop command (173).** In Safe mode the Roomba will not charge, even sitting on its dock, and it does not sleep — skip this and the battery silently drains.

CONTENTS

1. Parts list
2. Stage 1 — Tether test
3. Finding the hidden port
4. Identifying the pins with a meter
5. Wiring: three wires only
6. Two ways to connect a computer
7. The protocol & the test program
8. Troubleshooting
9. Stage 2 — Pi rider + camera (planned)
10. Stage 3 — Free roaming (planned)
11. Credits & sources

Parts list

Prices are approximate, checked late October 2026. Nothing exotic — the expensive parts (the robot, the computer, the Uno, the Pi, the camera) were already on the author's shelf.

ITEM	USED IN	NOTES
iRobot Roomba 675	All stages	600-series, Wi-Fi model. Author already owned it. The model number is printed on the underside label.
Multimeter	Stage 1	Continuity + DC volts. Non-negotiable — the whole pin-identification method depends on it.
Male-to-male jumper wires + painter's tape	Stage 1	Three wires for the bench test; tape for labels and strain relief.
Mac or PC	Stage 1	Runs the Python test program. Commands in this guide are macOS Terminal commands.
Arduino Uno	Stage 1, route A	Free if you own one — works as a USB-serial bridge. Author already owned one.
5 V USB-TTL serial cable	Stage 1, route B	SparkFun FTDI Cable 5V (DEV-09718), ~$26–28, or the DTECH DT-6555, usually cheaper. Must be a genuine 5 V TTL cable — see routes.
7-pin mini-DIN plug or cable	Stage 2	Replaces the bench jumpers with a connection that survives motion.
Raspberry Pi + Pi Camera Module v1	Stage 2	Camera: OV5647 sensor, 5 MP, CSI ribbon cable. Which Pi model is still an open decision — see Stage 2.
Logic level shifter, or two resistors	Stage 2	For a voltage divider on the Roomba → Pi line (5 V down to 3.3 V).
USB power bank	Stage 2	Powers the Pi independently. Never power the Pi from the Roomba's serial-port battery pins.

Stage 1 — The tether test

● IN PROGRESS ● CORE LINK VERIFIED

Before bolting anything to the robot, prove the port works: connect a computer, send commands, get sensor data back. If the firmware on a Wi-Fi model refused drive commands, this is where you'd find out — cheaply, with parts already on the desk.

On the author's 675, the answer came back good: the Roomba accepted commands and replied with sensor data. The one loose end is audible confirmation — the speaker works fine in normal cleaning, but the test program's beep/song hasn't been confirmed yet, because song playback only works in Safe mode and the first version of the program couldn't show the mode from the slightly-short sensor replies the Uno bridge produces. A program update (v1.1) that parses partial replies was in testing when this was written.

1.1 Finding the hidden port

There is no hatch on the surface. The port hides under the thin silver decorative ring that circles the top plate. iRobot's specification puts it like this: *“The Mini-DIN connector is located in the rear right side of Roomba, beneath a snap-away plastic guard.”*

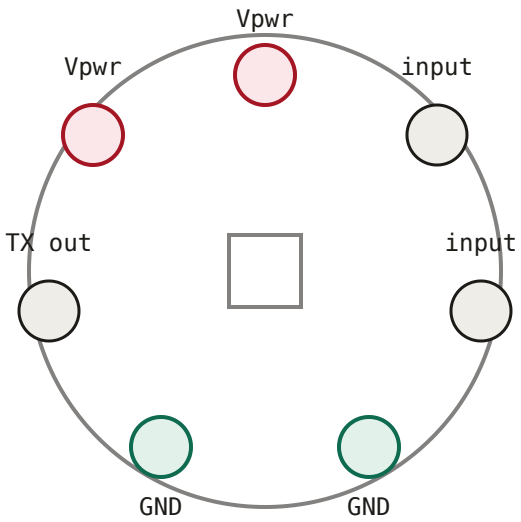
1. Pull the dust bin straight out. This exposes the back edge of the silver ring.
2. Starting at the bin opening, work fingertips or a plastic spudger under the back portion of the **thin silver ring** and lift, continuing around until it pops off. Do **not** pry the black top plate itself.
3. Under the ring, at the rear right, is a round socket with seven small holes around a square centre key: the 7-pin mini-DIN Open Interface connector.
4. Blow or brush dust out of the area around the socket before putting anything in the holes.

This matches how a Roomba 690 — the same Wi-Fi 600 generation — was opened up for the “Roomba RPi” project on Hackaday.io (project 178565), which is part of why this unit was worth trying at all: Wi-Fi models were rumoured to lack the port, and at least some demonstrably have it.

1.2 Identifying the pins — meter first, always

The specification's pinout is below, and it is real — but do not wire from it directly. The socket is drawn from opposite viewing sides in different documents, so the numbering runs clockwise in one and counter-clockwise in another; iRobot's own Create 2 cable

tutorial warns about the mirrored convention. Wire by a mirrored table and you can put battery voltage where a signal should go. So this guide identifies every hole empirically, with a meter, on your own robot.



Schematic only — the socket's hole pattern as a shape to recognise, **not** a wiring map. Orientation differs between robots and documents, which is exactly why the meter method below is the guide, and this picture is not.

Before measuring: charge first. A long-shelved Roomba's pack may be deeply discharged, and a flat pack makes every reading wander. The author's first session produced nonsense numbers (roughly 6.5 / 6.5 / 11 / 8.4 V across the holes) and a low-battery report from the robot. Put the Roomba on its dock until the CLEAN light is solid green, *then* map. Note that green means the charger finished — not that an aged pack is healthy. A healthy pack reads about 16–17 V; the author's charged-but-aged pack reads 14.9 V.

Step A — find the ground pair by continuity.

1. Prepare three jumper wires: wrap a painter's-tape flag near one end of each, and one snug layer of tape around each pin's base, leaving about 5 mm of bare pin. The tape stops neighbouring pins touching.
2. Set the multimeter to continuity (the sound-wave symbol). Wake the Roomba with one press of CLEAN.

3. Push a jumper into each of the two holes that sit side by side on the arc nearest the Roomba's outer rim. Touch the meter probes to the jumpers' free ends.
4. The right pair beeps, and the reading settles at **0 Ω** — both holes are the same piece of copper inside the robot. (A beep alone isn't proof: the meter also beeps for paths *through* the circuitry, which read in the tens of ohms. The number decides.) Label both flags **GND**, keep one jumper in, pull the other out.

Step B — sort the rest by voltage. Meter on DC volts (V⎓) from here on — never continuity on a live hole. The black probe stays on the free end of your ground jumper. Probe each remaining hole, one jumper at a time, and sort what you find:

- **Two holes at full battery voltage** — an identical matching pair (author's aged pack: 14.9 V and 14.9 V). These are the unregulated battery pins, Vpwr. **They stay empty for the whole test.** Mark them DO NOT USE.
- **One hole is the Roomba's serial output (TX)** — it idles at logic high and carries the replies. Flag it TX and leave its jumper in.
- **Two holes are inputs (RX and BRC)** — they can idle at the same voltage and *cannot be told apart with a meter*. Leave a jumper in one and flag it “RX?” — if commands later get no response, swap to the other hole. Both are inputs, so the trial is safe. That trial is exactly how the author's RX hole was identified: the first 9.3 V hole ignored everything; the second one answered.

> **The author's actual map, for honesty**
>
> On a charged-but-aged pack, probing around the socket in order gave: `GND, GND, 11.3, 14.9, 14.9, 9.3, 9.3` volts. The signal pins did **not** show the textbook 5 V / 0 V values on this pack — which is the whole point of this method. Identification is by continuity, pair-matching, and trial, never by matching a printed table and hoping.

The specification's pinout — for reference and cross-checking, with the warning above attached:

PIN	NAME	WHAT IT IS	IN THIS BUILD
1	Vpwr	Roomba battery +, unregulated	**Stays empty**
2	Vpwr	Roomba battery +, unregulated (same rail as pin 1)	**Stays empty**
3	RXD	0–5 V serial input to the Roomba	One of the two meter-found input holes — found by trial
4	TXD	0–5 V serial output from the Roomba	The reply line
5	BRC	Baud Rate Change input	The other input hole — unused here
6	GND	Battery ground	The continuity-proven pair
7	GND	Battery ground (same node as pin 6)	Either ground hole works; use one

Pin numbers per iRobot's Roomba 600 Open Interface Specification (see Credits). Numbering direction differs between documents — trust the meter, not the numbers.

1.3 Wiring: three wires only

Only three wires ever connect between the Roomba and anything else: **ground, the Roomba's TX (its talk-out), and the Roomba's RX (its listen-in)**. The battery holes stay empty. End state before connecting the computer end, checked twice:

- [ ] Exactly three jumpers in the socket: GND, TX, RX?
- [ ] Battery holes empty
- [ ] No bare pins touching a neighbour
- [ ] A loose loop of the three wires taped to the Roomba's top, so any tug pulls tape, not pins

Using individual jumper wires in this socket has precedent — a builder in the Synthiam community ran a Roomba 530 this way: *"I'm actually using jumper cables instead of a mini din and the connection is good."* Good for a bench test. It is not a rolling connection: jumpers wiggle loose and can short, which is why Stage 2 starts with a real plug.

1.4 Two ways to connect a computer

The Roomba speaks 5 V TTL serial. A computer speaks USB. Something has to bridge them, and it must be a 5 V device — not an RS-232/DB9 serial cable (wrong voltages entirely), and not a CP2102-based cable (those signal at 3.3 V and are the “sometimes gets heard” option).

Route A — Arduino Uno as a bridge (free if you own one). This is the route the author's verified link ran on.

1. Uno unplugged from everything: push one jumper from **RESET** to a **GND** socket on the Uno's power row (the row printed IOREF, RESET, 3V3, 5V, GND, GND, VIN). This holds the Uno's own processor off for the whole test, so its USB chip passes the computer's bytes straight through to sockets 0 and 1.
2. Connect the three Roomba wires: Roomba **GND** → any Uno **GND**; Roomba **RX** → Uno socket **0**; Roomba **TX** → Uno socket **1**. (The labels line up without crossing on this route — normal, because the Uno's processor is asleep.)
3. Plug the Uno into the computer by USB. On a Mac, find today's port name — it changes with the USB socket, so look it up fresh every session:

   ```
   arduino-cli board list
   ```
   Copy

   Use the `/dev/cu.usbmodem…` port from the Arduino UNO row.

> **Known weakness of the Uno route**
>
> At 115200 baud the Uno bridge drops the tail of longer replies. The author's sensor reads consistently arrived as **8 of 10 bytes** — mode, voltage, current, and charge came through; capacity fell off the end. If your sensor readback is flaky on this route, that is the bridge, not the Roomba. Commands going in are unaffected, which is why the two-way link counts as proven despite the short replies.

Route B — a proper 5 V USB-TTL cable. The clean version: everything, including full sensor readback, should just work.

- **SparkFun FTDI Cable 5V** (DEV-09718), ~$26–28.

- **DTECH DT-6555** — genuine FTDI FT232RL chip, 5 V logic per its manual; usually cheaper. Manual: manuals.plus/asin/B07RBK2P47.
- iRobot's own Create 2 communication cable (item 4466502, $29.99) is the official part — it was out of stock when this was written.

Wire these cables by colour and function, never by position — the two brands run their wires in opposite order along the plug. On both:

CABLE WIRE	FUNCTION	CONNECTS TO
Black	GND	Roomba GND hole
Yellow	Cable RX (listens)	Roomba TX hole
Orange	Cable TX (talks)	Roomba RX hole
Red	5 V power out	**Nothing — tape it off**
Green / brown	Handshake lines	Nothing

1.5 The protocol & the test program

Everything the robot understands is in iRobot's Roomba 600 Open Interface Specification (last updated 10 August 2016). The serial settings are **115200 baud, 8 data bits, no parity, 1 stop bit, no flow control**. Commands are single-byte opcodes, some followed by data bytes. The ones this build uses:

OPCODE	COMMAND	WHAT IT DOES
128	Start	Starts the Open Interface. Always sent first. Beeps once — but only when starting from Off.
131	Safe	Safe mode: full control, but cliff and wheel-drop protection stays on.
132	Full	Full mode: protection off. This guide never uses it.
137	Drive	Drive at a velocity and turning radius.
139	LEDs	Sets the button LEDs.

OPCODE	COMMAND	WHAT IT DOES
140	Song	Defines a song (note numbers + durations).
141	Play	Plays a defined song. Works only in Safe or Full mode — ignored in Passive.
142	Sensors	Requests one sensor packet.
149	Query List	Requests several sensor packets in one reply.
173	Stop	Stops the Open Interface. Ends the session — see the rule below.

The sensor read uses Query List to ask for six packets in one go: **35** OI Mode, **21** Charging State, **22** Voltage (mV), **23** Current (mA, signed), **25** Charge (mAh), **26** Capacity (mAh). That reply is 10 bytes — the one the Uno bridge tends to deliver as 8. Useful detail: the mode is the very first byte, so even a short reply tells you whether you're in Safe mode.

One more number from the spec worth knowing before Stage 2: the battery pins are fused at **200 mA continuous / 500 mA peak**. That's a sensor budget, not a power supply — one more reason nothing gets powered from them.

The test program. The author's tool is a small Python program (`roomba_tether_test`, built on pyserial) with a numbered menu: **1** Start + Safe · **2** beep test (LEDs + a three-note song) · **3** sensor read · **4 / 5 / 6** drive nudges (forward and back at 100 mm/s for 1 s; a turn at radius 1) · **7** stop + quit. Setup on a Mac, from zero:

```
mkdir -p ~/roomba-bot
cd ~/roomba-bot
python3.12 -m venv .venv
source .venv/bin/activate
pip install pyserial
```

Copy

Prove the software before any hardware — the program has a dry-run mode that talks to a pretend Roomba:

```
python roomba_tether_test_v1.py --dry-run
# menu: 1, then 2, then 3, then 7 — watch the bytes it would send
```

Then the real run, with the Roomba awake (one CLEAN press, wait 3 seconds), on open floor with a metre of clear space:

```
python roomba_tether_test_v1.py --port /dev/cu.usbmodemXXXX
# swap in today's port from: arduino-cli board list
```

The actual byte frames, so anyone can rebuild the tool in any language:

```
# Start, then Safe mode
[128]
[131]

# Define song 0: three notes — C (72), E (76), G (79)
# durations are in 1/64 s units: 16, 16, 32
[140, 0, 3, 72, 16, 76, 16, 79, 32]

# Play song 0 (only works in Safe or Full mode)
[141, 0]

# Ask for sensor packets 35, 21, 22, 23, 25, 26 in one reply
[149, 6, 35, 21, 22, 23, 25, 26]

# Drive forward: velocity 100 mm/s, radius 32768 = straight
[137, 0, 100, 128, 0]

# End of session — always
[173]
```

The session rule

In Safe mode the Roomba **will not charge, even on its dock, and it does not sleep**. Every session ends with menu 7 / Stop (173) — or by sending Passive — before the robot goes back on the dock. Skip it and the battery drains silently overnight.

1.6 Troubleshooting

SYMPTOM	WHAT IT USUALLY MEANS	DO THIS
Sensor read returns 8 of 10 bytes	The Uno bridge dropping the reply tail at 115200 baud. Mode, voltage, current and charge are in the 8 you got.	Reseat the TX wire once; if the count is a flat 8 every time, accept it on the Uno route (the link is proven) or switch to the FTDI cable for clean reads.
Commands sent, no response at all	The wire is in the BRC input hole instead of RX — the two inputs can't be told apart by meter.	Move the RX wire to the other input hole and retry. Both are inputs, so this trial is safe.
No beep on Start	Usually nothing wrong: Start only beeps when the Roomba is starting from Off, and you woke it by hand first.	Judge control by the sensor reply's OI Mode byte instead — if it reads Safe, your commands are landing.
Song doesn't play, other commands work	Play is ignored in Passive mode.	Confirm the mode reads Safe (menu 3) after menu 1, then retry the song. Watch the CLEAN light too — the beep test sets it green.
“Could not open port” on a port that listed fine a minute ago	The USB-serial port name changed — it re-registers under a new number when the connection remakes.	Run `arduino-cli board list` again and use the exact port it shows now, without unplugging anything in between.
Voltage readings wander and match nothing	The battery is too flat to hold its rails up; floating pins read floating numbers.	Stop measuring. Jumpers out, Roomba on the dock until solid green (hours, possibly overnight for a shelved pack), then map again.

Expected state at the end of Stage 1: commands in, sensor data back, short drive nudges working under supervision. On the author's robot the first two are verified; the song confirmation was still open at the time of writing.

Stage 2 — The Pi rider

● PLANNED — NOT BUILT

Everything in this stage is a plan, written down before the build so the decisions are visible. Nothing here has been tested yet.

The idea: the laptop goes away. A Raspberry Pi rides on top of the Roomba permanently, wired to the same Open Interface port, with a camera looking forward. The Pi runs the same kind of Python + pyserial control code as the tether test, and streams the camera over Wi-Fi (MJPEG-style) to a web page with a joystick — drive the robot around the house from a phone, seeing what it sees.

What changes from the bench setup

- **A real plug.** The three bench jumpers are replaced by a proper 7-pin mini-DIN plug or cable with strain relief, so connections survive motion and vibration.
- **The camera.** A Pi Camera Module v1 (OV5647, 5 MP) on its CSI ribbon cable, mounted facing forward. Roomba-height video is chair legs and baseboards; a short mast sees more of a room but no longer fits under furniture. Mounting height is a trade-off to settle in the build.
- **Level care on the serial line.** The Roomba's TX is 5 V logic; the Pi's GPIO pins are 3.3 V and are not 5 V-tolerant. The Roomba → Pi line needs a voltage divider (two resistors) or a level shifter. The other direction is fine as-is: the Roomba reads the Pi's 3.3 V as a valid high.
- **Independent power.** The Pi gets its own USB power bank strapped on top. It is never powered from the serial port's battery pins — those are fused at 200 mA and a Pi with a camera wants far more than that. (A regulated tap off the Roomba's battery is a possible later refinement, not the first version.)
- **Software.** Python + pyserial on the Pi for drive control — Start, Safe, Drive, sensor polling — plus the camera stream and a small web control page.

Open decision

Which Pi? A Zero-class board suits the power and weight budget of a rider much better than a full-size Pi 3 B+, and the known prior art (Roomberry, below) used a

Pi Zero W. The author's full-size Pi is also earmarked for another project. Not settled yet.

Prior art: the **Roomberry** project put a Pi Zero W + Pi Camera on a 600/700-series Roomba doing essentially this — streaming video over Wi-Fi while the Pi drives through the serial port. This stage follows that recipe, on a 675.

Stage 3 — Free roaming ● PLANNED — NOT BUILT

With the rider working, the cable to the laptop is already gone — “free roaming” means driving the robot around the house from the web page, untethered, and learning what that takes in practice. Also unbuilt and untested:

- **Safe mode's protections stay in charge.** A cliff or wheel-drop event stops the motors and drops the robot back to Passive mode mid-drive. The driving software has to notice mode changes and re-enter Safe deliberately, not just keep sending Drive commands into the void.
- **Drive supervision.** First runs on open floor, hand hovering, exactly like the bench nudges — lifting the robot is still the emergency stop.
- **Pets.** The author's household has two small dogs; they are in another room for first runs, and how the robot behaves around them is an open question, not an assumption.
- **Dock return.** Getting back to the dock — by driving to it under camera view, or handing control back to the Roomba's own dock-seeking — is untested.
- **Honest limit:** an older Roomba has no lidar and no map. Remote-driving by camera is the achievable win here; “go to the kitchen on your own” is a different, much harder project.

Credits & sources

This build stands on other people's documentation and prior builds. In order of how much weight they carry here:

- **iRobot** — the Roomba 600 Open Interface Specification (last updated 10 August 2016): the protocol, pinout, opcodes, and safety limits this entire guide is built on. Also the Create 2 cable documentation, and the Create 2 communication cable listing.
- **Synthiam community** — the decorator-ring port location for this Roomba generation (Roomba 595 answer) and the jumper-wire bench precedent (Roomba 530 build: “using jumper cables instead of a mini din… the connection is good”).
- **Hackaday.io project 178565, “Roomba RPi”** — proof that a Wi-Fi 600-series unit (a 690) carries the port and can be Pi-hacked the same way.
- **Roomberry** — the Pi-camera-on-a-Roomba prior art that Stage 2 follows: Pi Zero W + Pi Camera on a 600/700-series Roomba, streaming over Wi-Fi.
- **SparkFun** — product documentation for the FTDI Cable 5V (DEV-09718). **DTECH** — the DT-6555 manual (FTDI FT232RL, 5 V logic): manuals.plus/asin/B07RBK2P47.

Where a source has no stable public link, it is credited by name rather than given an invented address.

A personal build log by **Caitria** (Bay Area), documenting her own Roomba 675. Status as of 1 November 2026: Stage 1 in progress with the two-way serial link verified; Stages 2 and 3 planned.

Roomba, iRobot, and Create are trademarks of iRobot Corporation. This is an independent hobbyist build log, not affiliated with or endorsed by iRobot. Hacking your own robot is at your own risk — meter first.