

EcoSense Smart Garbage Bin: Technical Documentation

EcoSense is a solar-powered, self-charging IoT smart garbage bin. It measures waste weight (load cell) and fill level (ultrasonic) with a single ESP32, runs on a solar panel on its lid that charges an internal battery, sends both readings to Firebase through a JioFi 4G hotspot, and alerts staff on WhatsApp when a bin crosses its threshold. One web dashboard monitors any number of bins at the same time.

1. System overview

Problem statement

- Bins are emptied on a fixed schedule, not on actual need: some overflow before pickup, others are collected half-empty.
- Staff have no visibility of how full or heavy a bin is until they physically check it.
- Overweight bins create handling and liner-tear risk, which a volume check alone does not catch.
- Smart bins that need a mains socket or a network cable can only go where that wiring exists, which rules out parks, roadsides and open campuses.

Key features

- **Solar powered and self-charging:** a 20 W panel on top of the lid charges an internal LiFePO4 battery, so the bin needs no mains socket or power cable.
- **Eco-friendly:** runs on renewable energy and cuts collection trips, because bins are emptied only when they are actually full or heavy.
- **Dual sensing:** weight from a load cell under the bottom weighing platform, fill level from an ultrasonic sensor under the lid.
- **Fully wireless:** 4G internet through a JioFi hotspot, so a bin can be placed anywhere with mobile coverage.
- **Monitor any number of bins:** every bin reports to its own node in Firebase and appears on one dashboard with live status and search.
- **Instant WhatsApp alerts:** one alert per threshold crossing for a full or overweight bin, sent to all configured staff numbers.
- **Single control box:** all electronics sit in one sealed box inside the bin, easy to service or swap.

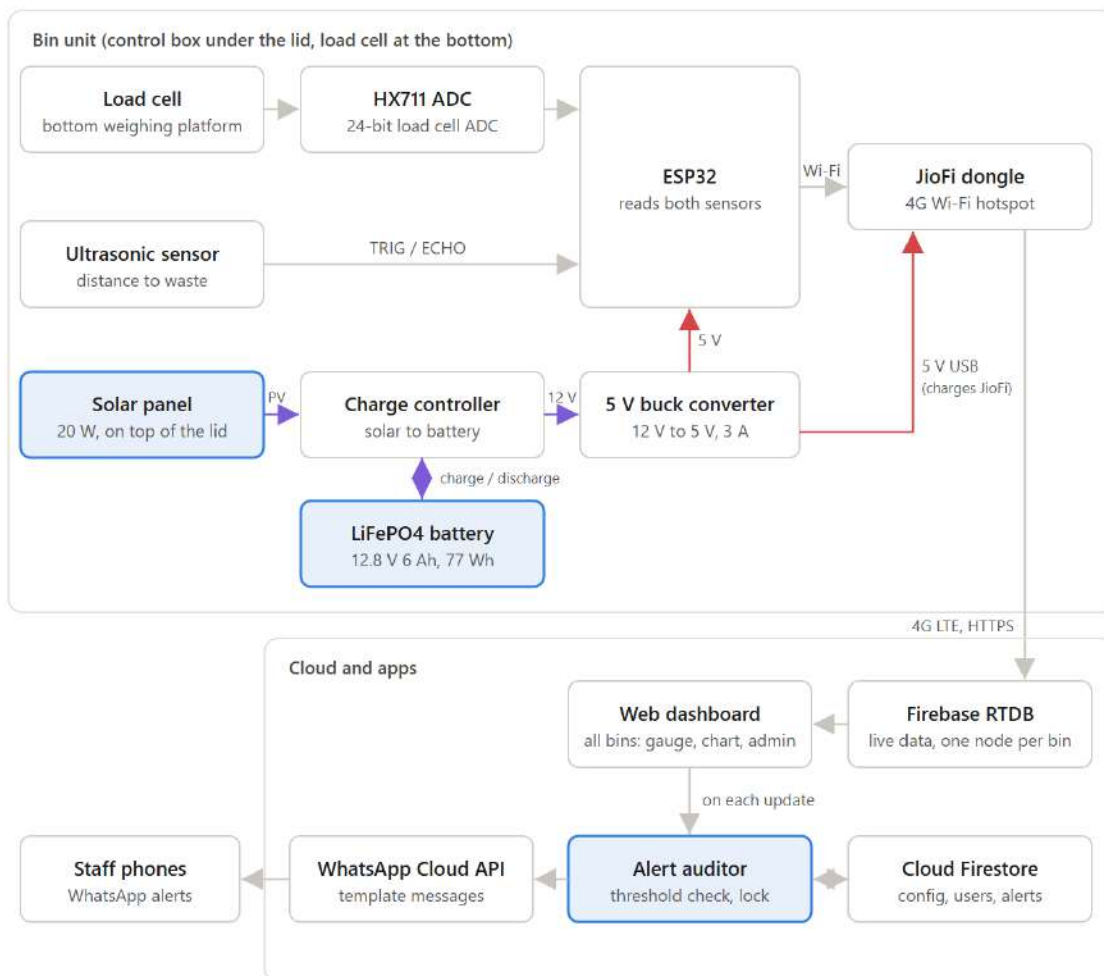
Overall functionality

- **Power:** the solar panel on the lid charges a 12.8 V LiFePO4 battery through a solar charge controller; a buck converter turns the battery voltage into the 5 V that runs the ESP32, the sensors and the JioFi, day and night.
- **Sense:** the ESP32 reads a load cell under the bottom weighing platform through an HX711 ADC for weight (0 to 40 kg range in the app) and an ultrasonic sensor under the lid for distance to the waste surface, converted to fill level 0 to 100%.
- **Transmit:** the ESP32 joins the Wi-Fi hotspot of a JioFi 4G dongle, which provides the internet link, and writes `weight`, `level` and a heartbeat value `IsON` to Firebase Realtime Database under the bin's ID.

- **Monitor:** a Next.js web dashboard lists every bin with its online/offline status (offline if no new heartbeat for 30 minutes) and, per bin, shows live weight, an animated level gauge (green below 75%, amber 75 to 90%, red above 90%) and a 30-point trend chart.
- **Alert:** when level reaches the threshold (default 90%) or weight reaches the threshold (default 35 kg), a server flow sends a WhatsApp template message (level_alert / weight_alert) to the configured numbers and logs it; dashboard warnings repeat at the set interval (default 1 hour).
- **Manage:** admins manage users and WhatsApp recipients from the admin panel; any logged-in user can set the alert thresholds and repeat interval, and rename, relocate or delete a bin from its settings. Bins are added as documents in the Firestore bins collection; any number can be added, each at `/bin/<binId>`.

2. System block diagram

Solar-powered bin streams to Firebase; crossing a threshold sends a WhatsApp alert



EcoSense block diagram · solar power, device, cloud, alerts

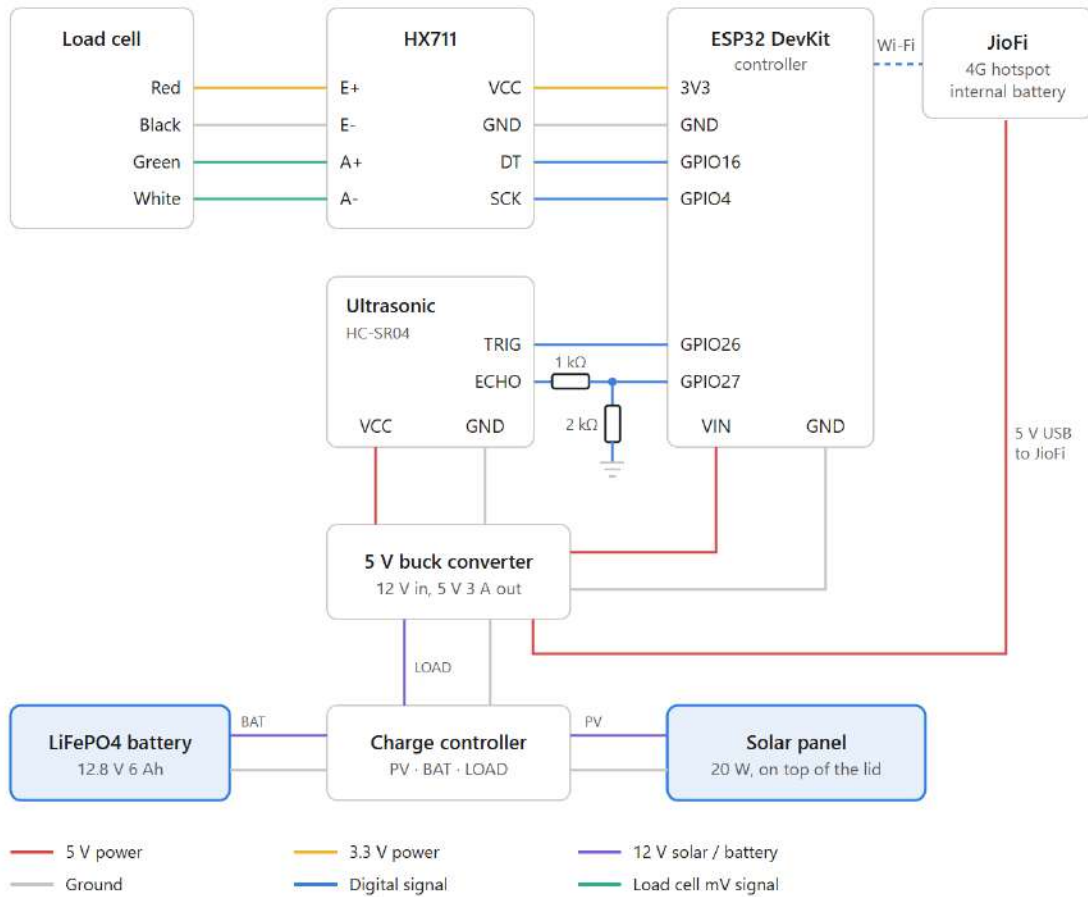
The solar panel on the lid charges the LiFePO4 battery through the charge controller, and a buck converter turns the battery's 12 V into 5 V for the ESP32, the ultrasonic sensor and the JioFi. The ESP32 reads both sensors and pushes the readings to Firebase RTDB through the JioFi 4G hotspot. The dashboard shows all bins live; each update runs the alert auditor (highlighted), which reads thresholds from Firestore and sends WhatsApp alerts.

Block	Function
Load cell + HX711	Converts waste weight on the bottom weighing platform to a 24-bit digital reading
Ultrasonic sensor	Measures lid-to-waste distance, converted to fill %
ESP32	Reads both sensors, computes fill %, writes readings and heartbeat to RTDB
JioFi 4G dongle	Wi-Fi hotspot that gives the ESP32 internet over 4G LTE; kept charged from the bin's 5 V rail
Solar panel (20 W)	Mounted on top of the lid; converts sunlight into charging power
Solar charge controller	Charges the battery safely and cuts the load off if the battery runs too low
LiFePO4 battery (12.8 V 6 Ah)	Stores solar energy; runs the bin at night and on cloudy days
5 V buck converter	Steps 12 V battery voltage down to 5 V for the ESP32, ultrasonic sensor and JioFi
Firebase RTDB	Live data, one node per bin, synced in real time to all viewers
Web dashboard	Bin list with status and search, live gauge, trend chart, admin settings
Alert auditor	Compares values to thresholds, locks, sends alert once, tracks heartbeat
Cloud Firestore	Bin config, thresholds, users, alert and message logs
WhatsApp Cloud API	Delivers template alerts to staff phones

3. Circuit diagram

WeightWise wiring: solar-charged battery powers the bin; ECHO is divided to 3.3 V

Single ESP32, internet through a JioFi 4G hotspot, all modules in the control box



EcoSense wiring · every module and pin

The solar panel charges the LiFePO4 battery through the charge controller. The controller's LOAD output feeds a 12 V to 5 V buck converter, whose 5 V rail powers the ESP32 (VIN) and the ultrasonic sensor and charges the JioFi over USB. The HX711 runs from the ESP32's 3.3 V pin so its DT / SCK logic matches the ESP32. The HC-SR04 ECHO output is 5 V, so a 1 kΩ / 2 kΩ divider drops it to 3.33 V ($5 \text{ V} \times 2 \text{ k}\Omega / 3 \text{ k}\Omega$) before GPIO27. The JioFi connects to the ESP32 over Wi-Fi only.

4. Pin connections, power and communication

Every sensor is wired directly to one ESP32 DevKit; there is no second microcontroller. All electronics sit in one sealed control box inside the bin under the lid; only the load cell cable runs down to the weighing platform at the bottom. Internet comes from the JioFi over Wi-Fi and power from the solar panel, so the bin needs no mains or network cable.

Pin-to-pin connections

From (device : pin)	To (device : pin)	Signal	Level
Load cell : Red (E+)	HX711 : E+	Excitation +	3.3 V
Load cell : Black (E-)	HX711 : E-	Excitation -	GND
Load cell : Green (A+)	HX711 : A+	Bridge output +	mV
Load cell : White (A-)	HX711 : A-	Bridge output -	mV
HX711 : VCC	ESP32 : 3V3	Power	3.3 V
HX711 : GND	ESP32 : GND	Ground	0 V
HX711 : DT (DOUT)	ESP32 : GPIO16	Serial data	3.3 V
HX711 : SCK	ESP32 : GPIO4	Serial clock	3.3 V
Ultrasonic : VCC	5 V buck : OUT+	Power	5 V
Ultrasonic : GND	5 V buck : OUT-	Ground	0 V
Ultrasonic : TRIG	ESP32 : GPIO26	10 μ s trigger pulse	3.3 V (accepted by HC-SR04)
Ultrasonic : ECHO	1 k Ω in series to ESP32 : GPIO27, 2 k Ω from GPIO27 to GND	Echo pulse width	5 V divided to 3.3 V
Solar panel : + / -	Charge controller : PV+ / PV-	Solar input	12 V nominal (up to 22 V open circuit)
Battery : + / -	Charge controller : BAT+ / BAT- (3 A fuse on +)	Battery charge / discharge	12.8 V
Charge controller : LOAD+ / LOAD-	5 V buck : IN+ / IN-	Protected battery output	12.8 V
5 V buck : OUT+	ESP32 : VIN	Power	5 V
5 V buck : OUT-	ESP32 : GND	Common ground	0 V
5 V buck : OUT+ / OUT- (USB cable)	JioFi : USB charging port	JioFi charging	5 V
ESP32 : GPIO2	Onboard LED	Status (optional)	3.3 V
ESP32 : Wi-Fi	JioFi hotspot	2.4 GHz Wi-Fi link	wireless

Wire colours are the common convention for 4-wire load cells; confirm against the cell's datasheet. Run the load cell lead from the bottom platform to the control box as shielded 4-core cable along the inner wall, with a slack loop at the lid hinge so opening the lid does not pull on it.

Power distribution

Rail	Source	Feeds	Est. load
Solar input	20 W panel on top of the lid	Charge controller PV input	up to about 1.1 A at 18 V
12 V battery	12.8 V 6 Ah LiFePO4 through the charge controller	5 V buck converter (from the LOAD output)	about 0.16 A average
5 V	Buck converter	ESP32 VIN, ultrasonic sensor, JioFi USB charging	about 0.35 A average, 1.3 A peak
3.3 V	ESP32 onboard regulator (AMS1117)	ESP32 module, HX711, load cell	up to 0.25 A during Wi-Fi TX
GND	Common point at the buck converter output	ESP32, HX711, ultrasonic	n/a

Add a 470 μ F capacitor across 5 V at the ESP32 VIN to absorb Wi-Fi current spikes.

Solar energy budget

Item	Estimate
ESP32 with Wi-Fi on (upload every 5 s)	0.45 W, about 10.8 Wh/day
Sensors (HX711, load cell, HC-SR04)	0.1 W, about 2.4 Wh/day
JioFi 4G hotspot, always on	1.2 W, about 28.8 Wh/day
Buck converter losses (about 88% efficient)	0.24 W, about 5.7 Wh/day
Total demand	about 2.0 W, about 48 Wh/day
Solar harvest (20 W $\times$ 4.5 peak sun hours $\times$ 0.75)	about 67 Wh/day on a clear day
Battery reserve (77 Wh, 90% usable)	about 1.4 days with no sun

Estimates come from typical datasheet figures, not measurements on the unit. The 0.75 factor covers the flat mounting on the lid, dust and charging losses. The JioFi is about 60% of the load, so it decides the battery size: for long cloudy spells such as the monsoon, a 12.8 V 12 Ah battery gives about 2.9 days of reserve. Recommended addition: wire battery + through a 100 k Ω / 22 k Ω divider to ESP32 GPIO34

and send a `battery` value to RTDB, so the dashboard can show each bin's charge level (not yet in the app).

Communication links

Link	Between	Protocol	Settings
HX711 bus	HX711 to ESP32	2-wire clocked serial (HX711 proprietary)	Channel A, gain 128
Ultrasonic	Sensor to ESP32	Pulse width (TRIG / ECHO)	distance = echo μ s $\times$ 0.0343 / 2 cm
Local network	ESP32 to JioFi	Wi-Fi 802.11 b/g/n, 2.4 GHz	JioFi SSID and password set in firmware
Internet	JioFi to Firebase	4G LTE (Jio network)	Mobile data
Cloud uplink	ESP32 to Firebase RTDB	HTTPS / REST (Firebase-ESP-Client)	Node /<binId>
Alerts	Server to WhatsApp	WhatsApp Cloud API (Graph API v21.0), HTTPS	Template messages
Debug	ESP32 to PC	USB serial	115200 baud

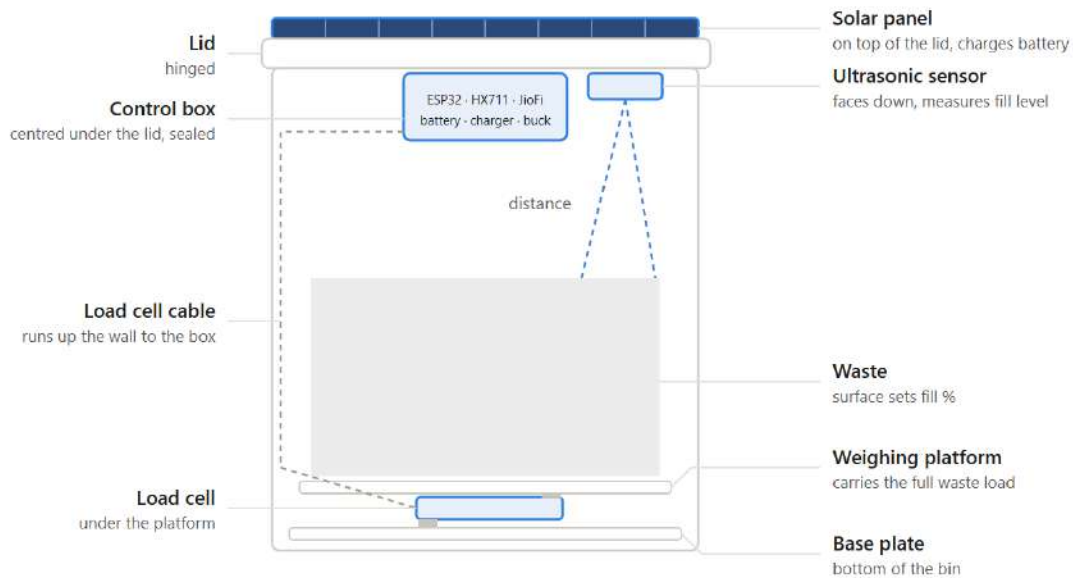
5. 3D CAD model

The design render shows the finished bin; the section view below it shows the internal layout, with the sensing and power parts highlighted.

EcoSense design render · level sensor, control unit, weighing platform

Section view: solar panel on the lid, control box under it, load cell at the bottom

Reference layout for the CAD model, not to scale



EcoSense mechanical layout · section view

The solar panel sits flat on top of the lid. The sealed control box is mounted at the centre of the lid's underside, inside the bin, and holds the ESP32, HX711, JioFi, battery, charge controller and buck converter; the ultrasonic sensor sits next to it facing down at the waste. The load cell sits under the weighing platform at the bottom, so it carries the full waste load.

CAD deliverable	Format	Notes
Full assembly	EcoSense.f3d (Fusion 360) EcoSense.step	Bin body, lid, 2 x 2 solar array and rails, control box, ultrasonic sensor, weighing platform, load cell, cable
Exploded view	PNG render	Every part labelled with the names used in section 6
Section view	PNG render	Load path from weighing platform to load cell to base
Sensor mount detail	PNG + dimensions	Ultrasonic height above the waste, allowing for the sensor's blind zone

CAD deliverable	Format	Notes
Control box	STEP + PNG	ESP32, HX711, JioFi, battery, charge controller and buck positions; cable glands for the solar and load cell cables; USB access; vent / IP rating
Solar panel mount	STEP + PNG	Panel fixing on the lid, cable entry through the lid, sealing against rain
2D drawings	PDF	Overall dimensions, bin volume, mounting holes

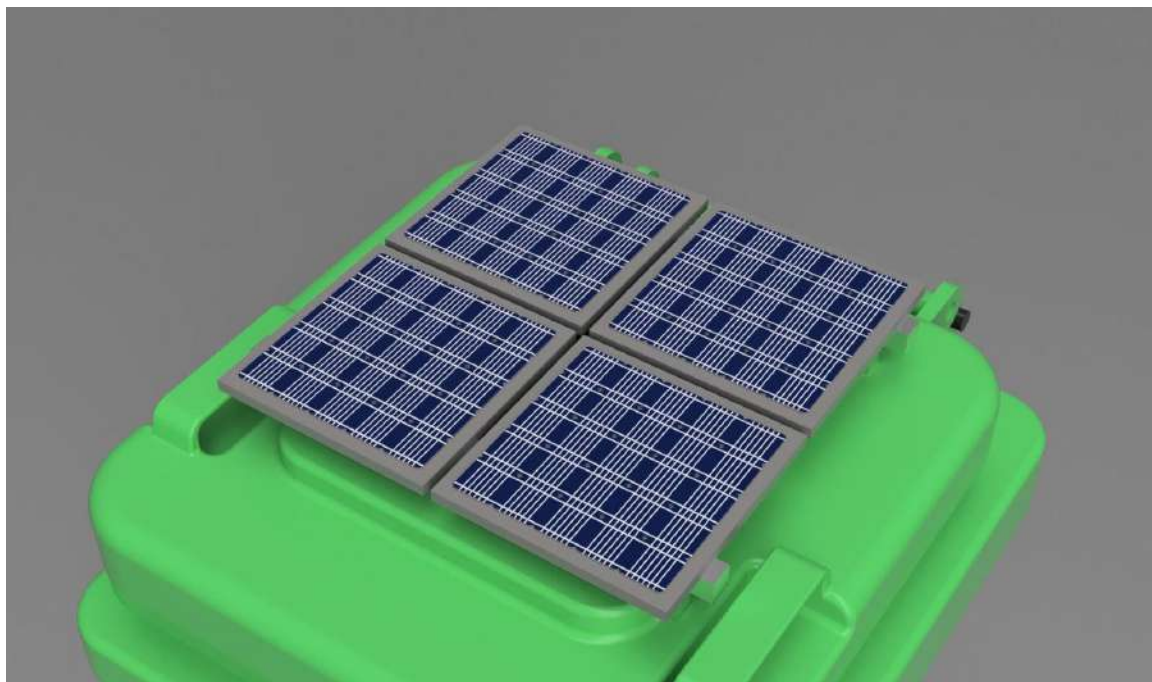
Rendered views of the working parts



Complete device: 2 x 2 solar array on rails over the lid, control box and weighing system inside the wheelie bin



Lid open: the control box and ultrasonic sensor are fixed under the lid and swing up with it on the hinge



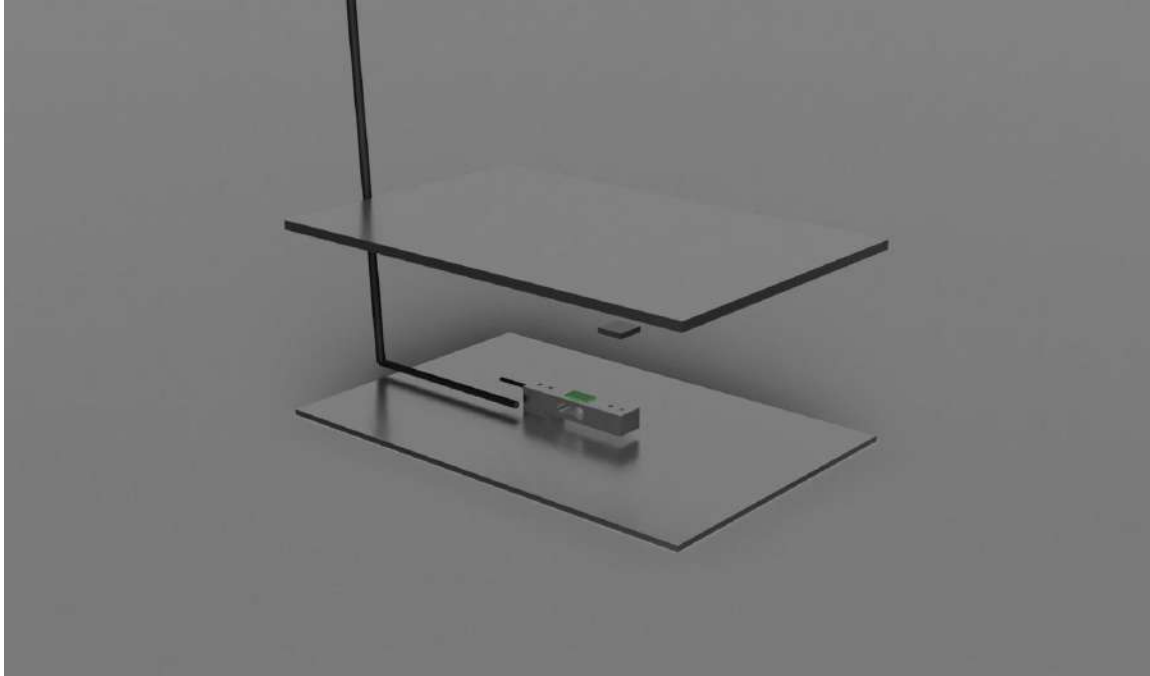
Solar array: four 280 mm panels on aluminium rails and legs, clear of the lid surface



Control module: IP65 enclosure (ESP32, HX711, JioFi, LiFePO4 battery, charge controller, 5 V buck), cable glands facing the hinge, HC-SR04 sensor beside it



Internal layout with the bin walls hidden: electronics under the lid, weighing platform on the floor, load-cell cable up the side wall



Weighing system, exploded: platform, load-end spacer, bar load cell on its fixed-end spacer, base plate



Bar load cell: the dumbbell hole concentrates bending strain under the potted strain gauges; the 4-wire cable runs to the HX711

6. Component specifications

Values are typical datasheet figures (approximate, not measured on the unit).

Component	Model number	Operating voltage	Current consumption	Communication protocol	Measurement range	Purpose in EcoSense
Controller	ESP32-WROOM-32 (ESP32 DevKit V1)	5 V in (VIN/USB), 3.3 V logic	about 80 mA avg, 240 mA peak on Wi-Fi TX	Wi-Fi 802.11 b/g/n, GPIO, USB serial	n/a	Reads both sensors, computes fill %, sends data to Firebase
Load cell	50 kg bar-type strain gauge (e.g. YZC-133 / YZC-131)	3.3 V excitation from HX711	about 3.3 mA (1 k Ω bridge)	Analog mV (Wheatstone bridge)	0 to 50 kg (app scaled to 40 kg max)	Measures waste weight under the bottom weighing platform
Load cell ADC	HX711 module	2.6 to 5.5 V (run at 3.3 V)	under 1.5 mA	2-wire clocked serial (DT/SCK)	24-bit, ± 20 mV at gain 128	Amplifies and digitises the load cell signal
Level sensor	HC-SR04	5 V	about 15 mA	Trigger / echo pulse	2 to 400 cm	Measures distance to waste surface, converted to fill %
Internet gateway	JioFi 4G dongle	Internal battery, 5 V USB charging	about 1.2 W average	4G LTE to internet, Wi-Fi hotspot to ESP32	n/a	Provides internet access; kept charged from the bin's 5 V rail
Solar panel	20 W 12 V monocrystal line panel	about 18 V at max power, 22 V open circuit	up to about 1.1 A output	n/a	0 to 20 W	Mounted on top of the lid; charges the battery

Component	Model number	Operating voltage	Current consumption	Communication protocol	Measurement range	Purpose in EcoSense
Charge controller	12 V 10 A solar charge controller with LiFePO4 setting (PWM or MPPT)	12 V battery, PV input up to about 25 V	about 10 mA self-use	n/a	10 A max	Charges the battery; low-voltage disconnect protects it
Battery	12.8 V 6 Ah LiFePO4 with built-in BMS	10 to 14.6 V	supplies about 0.16 A average	n/a	77 Wh	Stores solar energy for night and cloudy days
Buck converter	LM2596 module set to 5.0 V	4.5 to 40 V in, 5 V out	up to 3 A out	n/a	n/a	Powers the ESP32, ultrasonic sensor and JioFi charging
Level shifter	Resistor divider 1 kΩ / 2 kΩ	5 V to 3.3 V	negligible	n/a	n/a	Protects ESP32 GPIO27 from the 5 V ECHO signal

Fill level formula (for the sensor MCU): $\text{level \%} = (\text{bin depth} - \text{measured distance}) / (\text{bin depth} - \text{blind zone}) \times 100$, clamped 0 to 100.

7. Software components

Three layers: C++ firmware on the device, Firebase as the cloud backend, and a TypeScript Next.js web app with a server-side alert flow.

Layer	Component	Language / tech	Role
Device	ESP32 firmware	C++ (Arduino core for ESP32), Arduino IDE	Joins JioFi Wi-Fi, reads HX711 and ultrasonic, computes fill %, writes JSON to RTDB

Layer	Component	Language / tech	Role
Device	Firebase-ESP-Client (Mobizt), WiFi.h, HX711 library	C++ libraries	Firebase RTDB client, auto Wi-Fi reconnect, load cell reads
Network	JioFi 4G dongle	4G LTE, Wi-Fi hotspot	Internet access for the bin
Cloud	Firebase Realtime Database	NoSQL JSON, real-time sync	Live data per bin: weight, level, IsON, levelAlarmSent, weightAlarmSent, lastSeen
Cloud	Cloud Firestore	Document DB	bins, system-settings, users, alerts, alert-locks, whatsapp-logs, whatsapp-status-events
Cloud	Firebase Authentication	10-digit mobile number login (mapped to a Firebase account)	Dashboard login; admin / user roles stored in Firestore users
Web app	Dashboard	Next.js 15, React 18, TypeScript	Bin list, live weight, level gauge, trend chart, admin panel
Web app	UI	Tailwind CSS, shadcn/ui (Radix), Recharts, lucide-react	Components, charts, icons
Server	binDataAuditor flow	TypeScript, Genkit, firebase-admin	Checks thresholds, takes atomic lock, sends alerts, updates heartbeat
Server	whatsapp-service.ts + webhook route	WhatsApp Cloud API (Graph v21.0)	Template alerts, 3 retries with backoff, 50 msg/s rate limit, delivery status tracking
Hosting	Vercel (live site), Firebase App Hosting config	Node.js	Serves the web app and API routes

The .ino currently in the repo is a serial-input test build (it accepts weight, level typed over USB serial); commit the sensor-reading version so the repo matches the unit.

Data flow

1. ESP32 reads weight from the HX711 and distance from the ultrasonic sensor, then computes level %.
2. ESP32 writes weight (grams), level (%) and IsON to RTDB at the bin's node through the JioFi hotspot (every 5 s in the reference example).
3. Dashboard listens to that node (onValue), updates the gauge and chart, and keeps the last 30 points.
4. Each update triggers binDataAuditor: it refreshes lastSeen when the heartbeat changes, and compares values to thresholds in Firestore.
5. On first crossing, it acquires a Firestore lock, sends WhatsApp alerts, sets the alarm flag, and logs to alerts. The flag resets when the value drops below threshold.
6. The WhatsApp webhook records sent / delivered / read / failed status in Firestore.

Monitoring multiple bins

- Each bin's firmware carries its own bin ID and writes to its own RTDB node /<binId>, so bins never overwrite each other. Adding a bin means flashing a new ID and adding a document (name, device ID, location) to the Firestore bins collection; the first bin ("Tree Side") is created automatically on first run.
- The dashboard sidebar lists every bin with live online/offline status and a search box; each bin opens at /bin/<binId> with its own gauge, chart and history.
- Alarm flags and alert locks are kept per bin, so the alert auditor handles each bin independently: a full bin at one site alerts without affecting the others.
- Thresholds, repeat interval and WhatsApp recipients are system-wide settings that apply to all bins.

Dashboard and admin features

- **Login:** users sign in with their 10-digit mobile number; admins see an extra Admin Panel entry in the sidebar.
- **Bin page:** live weight, level gauge and weight-over-time chart, each card opens an enlarged view; an Online / Awaiting connection indicator and "Last updated ... ago" show device freshness.
- **On-screen warnings:** when a bin is over threshold the page shows a red "URGENT: High Bin Level / Weight" banner, plays a warning beep and pops up a notice that a WhatsApp alert was also sent, repeated at the set interval; an offline bin shows a "Device Offline, showing last known values" banner.
- **Active warnings bell:** the sidebar (and the mobile header) has a bell that lists every bin currently in alarm, with a View Bin link to each.
- **Bin search:** the sidebar filters bins by name, location or device ID and shows how many match.

- **Bin settings:** the gear icon on a bin page edits its name and location (device ID is read-only) and deletes the bin after a confirmation.
- **Global settings:** level threshold 50 to 100%, weight threshold 0 to 40 kg in 1 kg steps, and dashboard warning repeat of 30 minutes, 1, 2 or 4 hours.
- **Admin panel:** add users by 10-digit number with an optional name and admin role, search users, rename them, make or remove admin, and delete them (numbers are shown masked); edit the WhatsApp recipient list (comma-separated, country code 91 added automatically); send a test `level_alert` or `weight_alert` to all recipients, logged as a test.
- **Mobile layout:** on phones the sidebar becomes a slide-out menu opened from a header bar.

8. Challenges and how they were solved

The software rows come from the code and fix notes in the repo; the last row is the hardware power redesign.

Challenge	Root cause	Solution
Duplicate WhatsApp alerts (one per open dashboard)	The alert check ran from every browser viewing the bin; all saw <code>alarmSent = false</code> before any set it	Firestore transaction lock per bin and alert type (<code>alert-locks</code>); first caller sends, others skip; lock expires after 30 s (fix note)
Repeat warnings never fired	Sensor updates every few seconds re-ran the React effect and reset the 1-hour timer	Removed live data from the effect dependencies and wrapped handlers in <code>useCallback</code> (fix note)
"Test message failed" though message was sent	Firestore rejects <code>undefined</code> fields in log documents	<code>sanitizeForFirestore()</code> strips <code>undefined</code> fields before every write (fix note)
Scaling beyond a few bins	Everything in Realtime Database; hard to query config, users and alert history	Hybrid: live sensor data stays in RTDB, config / users / alerts moved to Firestore; no firmware change (summary)
Unreliable alert delivery	Network errors and WhatsApp rate limits	Retry with exponential backoff (1 s, 2 s, 4 s), 50 msg/s throttle, delivery webhook (summary)
Detecting an offline bin	A static last value looks the same as a live one	ESP32 sends a changing heartbeat <code>IsON</code> ; server updates <code>lastSeen</code> on change; bin shows offline after 30 min without one

Challenge	Root cause	Solution
Page crash after framework upgrade	Next.js 15 made route params async	Unwrapped params with <code>React.use()</code> in <code>/bin/[binId]</code> (fix note)
Testing without hardware	Dashboard work blocked when no device was online	Demo mode generates simulated readings when a bin has no data
Bins tied to a power outlet	A 5 V mains adapter meant each bin could only stand next to a socket	20 W solar panel on the lid charging a 12.8 V LiFePO4 battery through a charge controller, sized for about 1.4 days without sun (section 4)