

Beach Mitra

An Autonomous AI-Powered Robot for Collecting Plastic Waste from Beaches

Varish Agarwal

Introduction

The first time I really looked at a beach properly looked, instead of just enjoying being there I was on a family trip and I had wandered past the busy stretch of sand toward the quieter end. There was a line of litter sitting exactly where the tide had dropped it: bottle caps, a torn crisp packet, a couple of water bottles half-buried in wet sand, and an astonishing number of small plastic fragments that had clearly once been something larger. A little further down, a group of volunteers in bright vests were working their way along the same line, bending down, picking up one piece, standing up, walking a few steps, and bending down again. They had been at it for hours. They were covering maybe a few hundred meters.

What struck me was not that the beach was dirty. It was how slow the solution was. The volunteers were doing something genuinely valuable, but the bottleneck in their work was not motivation or effort; it was the sheer amount of time a human spends simply searching. Finding the litter takes far longer than picking it up. That observation is what turned into this project.

The scale of the problem is bigger than any one beach. UNEP's global assessment of marine litter reports that between 19 and 23 million tonnes of plastic waste leak into aquatic ecosystems every year, and that plastics are "the largest, most harmful and most persistent fraction of marine litter, accounting for at least 85 per cent of total marine waste" (UNEP 2021). Almost all of it — around 98% — comes from land-based sources rather than from ships, which means beaches are not simply where the problem washes up; they are also one of the routes by which it gets in (Wikipedia contributors 2025a). A bottle left on the sand is not just untidy. It is a piece of plastic sitting one high tide away from the sea, where it will break down into fragments that are effectively impossible to recover.

Once plastic reaches the water, the damage stops being hypothetical. Studies collated in the marine plastic pollution literature identify at least 267 animal species affected by entanglement and ingestion, and a 2021 review identified 1,288 marine species known to ingest plastic (Wikipedia contributors 2025a). Roughly 90% of the plastic in the open marine environment is already microplastic too small to be collected by any realistic clean-up method. The window in which a piece of litter can still be picked up cleanly, as a single whole object, is short, and it closes on the beach.

So the obvious question is: why are we still doing this the slow way? There are, broadly, two existing answers. The first is volunteers, which is what I watched effective, low-impact, but limited by human time and physically demanding work. The second is mechanical beach cleaning: tractor-towed machines that rake or sieve the top layer of sand. These are fast, and they are widely used on busy tourist beaches, but they are indiscriminate. They lift seaweed and organic strandline material along with the rubbish, and that material is not waste it is the base of the beach food chain. Comparisons of groomed and ungroomed beaches have found macrofauna abundance almost nine times greater on ungroomed sand, with more than thirteen species present compared with fewer than three on groomed beaches (Wikipedia contributors 2025b). Mechanical cleaning solves the litter problem by removing a slice of the ecosystem along with it.

That gap is what I found interesting. There is a fast method that is environmentally crude, and a careful method that is slow. Nobody had given me a reason why the machine could not be careful. A machine that could identify a specific object, go to that specific object, and pick up only that object would have the speed of automation and the selectivity of a human volunteer. It would leave the seaweed alone because it would never have been told to look for seaweed.

That is the idea behind Beach Cleaner or as I have named it the "Beach Mitra". It is a four-wheel-drive robot built around a Raspberry Pi 4 Model B, which carries a camera, a 360 lidar, a display and a motor-control system. A custom-trained YOLOv8 object-detection model runs on the Pi and looks for target waste in the camera feed. When it finds something, the Pi works out where that object sits in the frame, converts that into a steering decision, and drives the robot toward it, while the RPLIDAR A1 watches the surroundings for obstacles the camera is not looking for. When the robot arrives, a front-mounted rotating roller — part of which I designed

and 3D printed in TPU, so it could be flexible enough to grip a bottle without being rigid enough to scrape sand — sweeps the object up and feeds it into an onboard collection bin.

The whole system works as a chain: **Detect** → **Locate** → **Navigate** → **Avoid Obstacles** → **Collect** → **Store**

I want to be honest about what that sentence means at this stage of the project. Each of those links has been built and tested individually. The camera detects bottles, the motors drive, the lidar returns clean scans, the roller turns and pulls objects into the bin, and the two power systems hold up. Recent beach testing has now demonstrated partial end-to-end autonomous operation. In four trials, the robot completed three successful detect–approach–collect cycles, although one intervention was required and the system has not yet demonstrated sustained autonomous operation over a longer beach-cleaning mission.

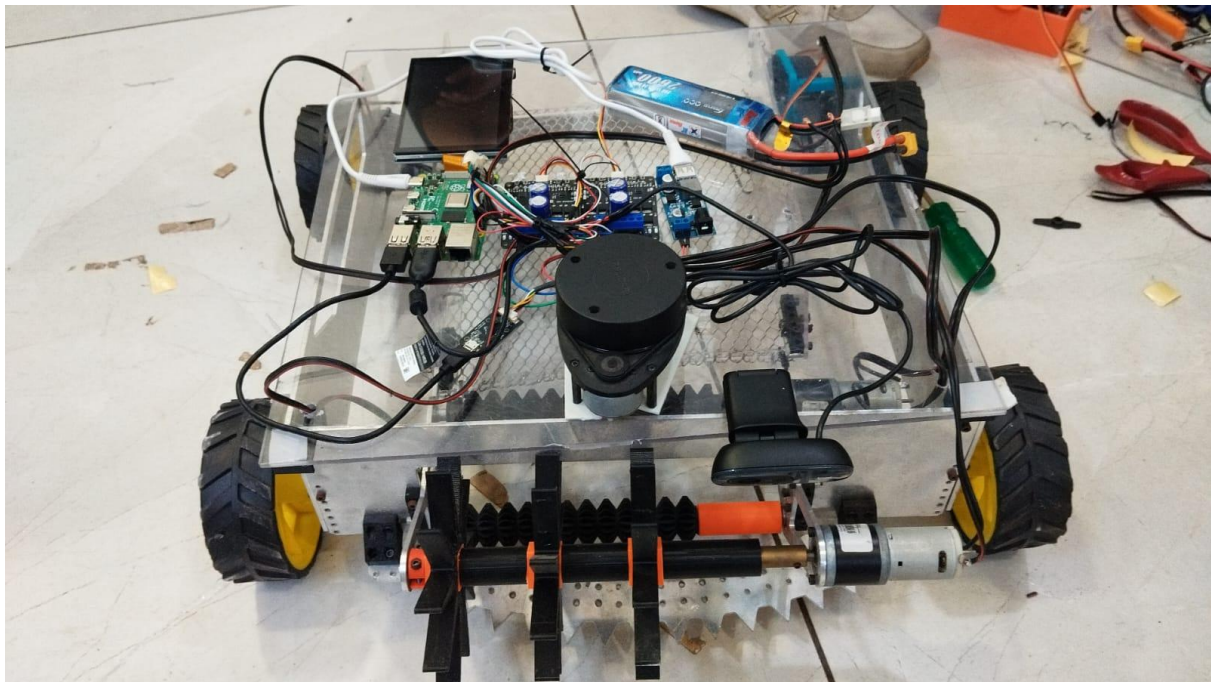


Figure 1 Robot

Aim & Objectives

The aim of this project is to design and build an autonomous robot that can find plastic waste on a beach using computer vision, drive itself to that waste while avoiding obstacles, and collect it mechanically selectively enough that it removes litter without stripping the natural material the beach ecosystem depends on.

I will have achieved my aim if:

- I build a working four-wheel-drive rover that can be commanded to move reliably on sand.
- A custom-trained object-detection model runs onboard the robot, in real time, and identifies target waste in a live camera feed rather than in pre-recorded images.
- The robot uses the position of a detected object to steer itself toward that object, without a human driving it.

- The lidar detects an obstacle in the robot's path and the robot changes its behaviour in response.
- The roller intake lifts a target object off the surface and deposits it in the collection bin.
- The complete detect → approach → collect cycle runs end to end on at least one target object without manual intervention.
- The robot leaves seaweed, driftwood, and other natural strandline material where it is, because it was never trained to target them.

My objectives were:

- Research the scale of the beach-litter problem and the environmental trade-offs of the cleaning methods currently used.
- Compare possible technical approaches—for the cleaning strategy, the perception method, and the collection mechanism and justify a choice for each.
- Learn the skills I did not have: training a deep-learning detector, working with lidar, CAD for a functional mechanical part, and 3D printing in a flexible material.
- Build a four-wheel-drive chassis capable of carrying the electronics, the battery mass and the collected waste.
- Train and deploy a custom YOLOv8 detector and get it running at a usable frame rate on a Raspberry Pi rather than on a laptop.
- Write control software that converts a detection into a motor command.
- Integrate the RPLIDAR A1 for obstacle avoidance.
- Design and manufacture a roller intake, including a 3D-printed TPU component, and iterate it until it lifts a bottle reliably.
- Separate the motor power supply from the compute power supply so that motor current draw cannot crash the Pi.
- Define a full testing plan covering detection, navigation, collection, power, and end-to-end autonomy and carry out as much of it as the build state allows.
- Evaluate honestly what works, what does not, and what I would do differently.

I have set these criteria out before reporting any results, deliberately. It is very easy, after months of work, to quietly redefine success as whatever you happened to achieve. Writing the bar down first makes that harder.

Wider Purpose

The most important thing this project demonstrates, I think, is not that a robot can pick up a bottle. It is that an AI model output can be allowed to change something physical in the world, and that doing so responsibly requires you to think about far more than accuracy.

Most school-level AI work stops at the screen. You train a model, you get a number, and the number is the result. Here, the number is only the beginning. A detection must be turned into a position, a position into a heading, a heading into four motor commands, and all of that must survive loose sand, direct sunlight, battery sag and a roller that might jam. The moment a model's prediction causes a machine to move, the cost of being wrong changes completely. A misclassification on a screen is a wrong label. A misclassification on a beach is a robot driving toward a nesting bird. That shift — from prediction to consequence — is the thing I actually learned, and it is the reason the ethics section of this report is not a formality.

Beyond that, the project makes a specific argument about environmental technology: that selectivity is a feature worth engineering for. The current fast method of beach cleaning is fast precisely because it does not discriminate. It sieves. The reason mechanized cleaning damages beaches is not that machines are inherently bad for beaches — it is that we have only built machines that cannot tell the difference between a bottle and a piece of kelp. Object detection changes that. A robot that has been trained on plastic and only plastic has selectivity built into it at the perception layer, which is exactly where mechanical rakes have none. If this approach scales, it offers a way to clean beaches at machine speed without the ecological cost that currently makes conservation groups wary of mechanical cleaning in the first place.

There is also a democratization angle that matters to me. The components in this robot are not exotic. A Raspberry Pi, a USB webcam, a hobby lidar, four geared motors and a 3D printer are within reach of a school engineering club, a coastal community group, or a university society in a country where marine litter is a serious problem and research budgets are not large. Commercial autonomous beach-cleaning machines exist, but they are expensive enough that the beaches which most need them are often the least likely to get them. Showing that a functional prototype of this idea can be built at this scale is part of the point.

Finally, the approach generalizes. The perception-to-action loop here is not specific to beaches. The same structure detects a target class, locate it, navigate it, act on it — applies to litter in parks, debris on runways, waste sorting on a conveyor, or weeding between crop rows. Each of those would need its own dataset, its own mechanical end effector and its own safety analysis, and I would not claim otherwise. But the architecture transfers, and that is what makes it worth building carefully.

Timeline

The table below is taken from the log I kept while working. The hours are recorded rather than estimated afterwards, which is why some of them are unflattering. Phase 4 in particular took roughly twice as long as I had planned, almost entirely because getting YOLOv8 to run at a usable speed on the Pi turned out to be a much harder problem than training it in the first place.

Phase	Breakdown of time and explanation
Phase 1: Identifying the problem and brainstorming (7 hours)	• Researched the scale of beach and marine plastic pollution using UNEP and peer-reviewed sources. • Investigated how beaches are currently cleaned and the ecological drawbacks of mechanical raking. • Brainstormed possible solutions: drone survey, tethered vehicle, robotic arm rover, suction system, autonomous rover. • Choose the autonomous vision-guided rover and define the aim and success criteria.
Phase 2: Research and component selection (11 hours)	• Compared perception options: color thresholding, image classification, object detection, and depth segmentation. • Compared collection mechanisms: gripper arm, scoop, vacuum, rotating roller. • Read Raspberry Pi, SLAMTEC, and Ultralytics documentation to confirm the components could do what I needed. • Finalize the component list and sourced parts.
Phase 3: Chassis and drive system (16 hours)	• Built the four-wheel-drive chassis and mounted four 200 RPM Johnson motors. • Wired the motor driver and verified direction control on all four motors. • Worked out the two-battery power architecture after the Pi browned out during the first motor test. • Mounted the camera, lidar, and 5-inch display and checked clearances.

Phase	Breakdown of time and explanation
Phase 4: AI model training and deployment (14 hours)	- Assembled and reviewed the training dataset of waste images. - Trained the custom YOLOv8 model and reviewed the training output. - Fought a NumPy, CV library conflict on the Pi that prevented Ultralytics from importing at all. - Got live inference running from the USB camera on the Pi and measured the frame rate. - Wrote the detection-to-steering logic that converts a bounding box into a turn command.
Phase 5: Mechanical intake design (15 hours)	- Designed the roller intake in CAD and modeled the flexible sweeping element. 3D printed the flexible component in TPU; iterated the geometry across several prints. - Mounted the roller, aligned it with the bin opening, and tuned the roller speed. - Bench-tested pickup on bottles, caps, and wrappers on a hard surface, then on a sand tray.
Phase 6: Lidar and obstacle avoidance (10 hours)	- Interfaced the RPLIDAR A1 with the Pi and confirmed clean 360 scans. - Write the forward-arc obstacle check and the avoidance behavior. - Tested obstacle response with static objects at measured distances.
Phase 7: Integration and subsystem testing (13 hours)	- Brought detection, navigation, lidar, and intake into one control program. - Diagnosed and fixed conflicts between camera inference timing and motor control timing. - Ran the bench tests recorded in the Testing and Evaluation section. - Identified the outstanding issues that currently block a full autonomous field run.
Phase 8: Documentation and reflection (10 hours)	- Wrote up the design decisions, test results, and limitations. - Produced the CAD renders, block diagram, and flowchart. - Reflected on what I would change and defined the field-testing plan.

Total recorded: 96 hours

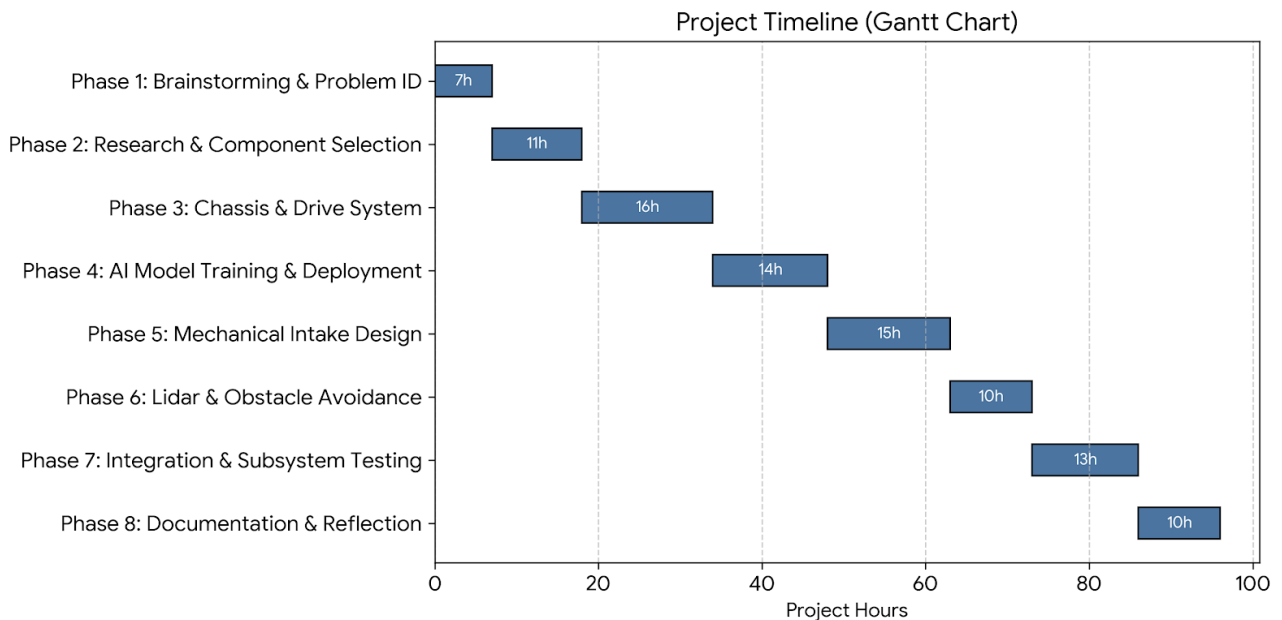


Figure 2 Development tracker showing hours logged per phase across the project.

Selection of Approach

Three decisions shaped this project. For each, I compared the alternatives and chose the option that best fit the problem.

- **Decision 1: Overall cleaning strategy**

Volunteer clean-ups are highly selective, but most of the time goes into searching, and there is far more coastline than there are volunteers. Drone surveys find litter quickly but can't collect it and miss small items. A remote-controlled vehicle still needs one person's full attention, so no time is saved. An autonomous vision-guided rover was the only option that is both selective and doesn't need constant supervision. It only targets what it was trained to recognise, so natural material is left alone. The trade-off is that everything depends on detection accuracy.

- **Decision 2: Perception method**

The robot needed to know where the waste is, not just whether it is there.

- **Colour thresholding** couldn't reliably tell litter from sand, foam and changing sunlight.
- **Image classification** only says litter is present, not where, so it can't guide steering.
- **A depth camera** works poorly in sunlight, is too heavy for the Pi to process and is expensive.
- **YOLOv8 object detection** (chosen) gives a label and a bounding box together (Ultralytics 2026), and the nano model runs on the Pi (Ultralytics 2023). The box's position tells the robot which way to turn, and its size shows how close the object is.

Decision 3: Collection mechanism

- **A gripper arm** needed centimetre-level accuracy from the vision, steering and odometry, and added extra weight and failure points.
 - **A fixed scoop** collected sand along with litter.
 - **Suction** risked sand damage, drew too much power and couldn't lift heavier bottles.
 - **A rotating roller with flexible TPU fingers** (chosen) sweeps a wide band, so precise alignment isn't needed. The TPU lifts objects rather than pushing them away. Its limitations are that some sand still gets in and buried or flattened items are hard to collect.
- **Changes from Original Plan**
 - **Beach testing:** delayed to allow safer, controlled testing first.
 - **AI detection:** expanded to use YOLO-based object detection.
 - **Collection mechanism:** modified after early tests.
 - **Autonomy:** took longer because navigation and obstacle avoidance needed more calibration.
 - **Testing:** more time spent on iterative testing than planned.

Final Selection

- The final system is a four-wheel-drive autonomous rover. A YOLOv8 detector on a Raspberry Pi 4 finds the waste, an RPLIDAR A1 avoids obstacles, and a TPU roller intake collects the waste. It runs on two separate LiPo supplies.
 - One model identifies and locates the waste.
 - The lidar handles obstacles separately, which keeps the control logic simple.
 - The roller tolerates small positioning errors.
 - Separate power supplies stop the Pi from browning out.
 - All the components are affordable and off-the-shelf, so the design can be reproduced.

Flow Chart

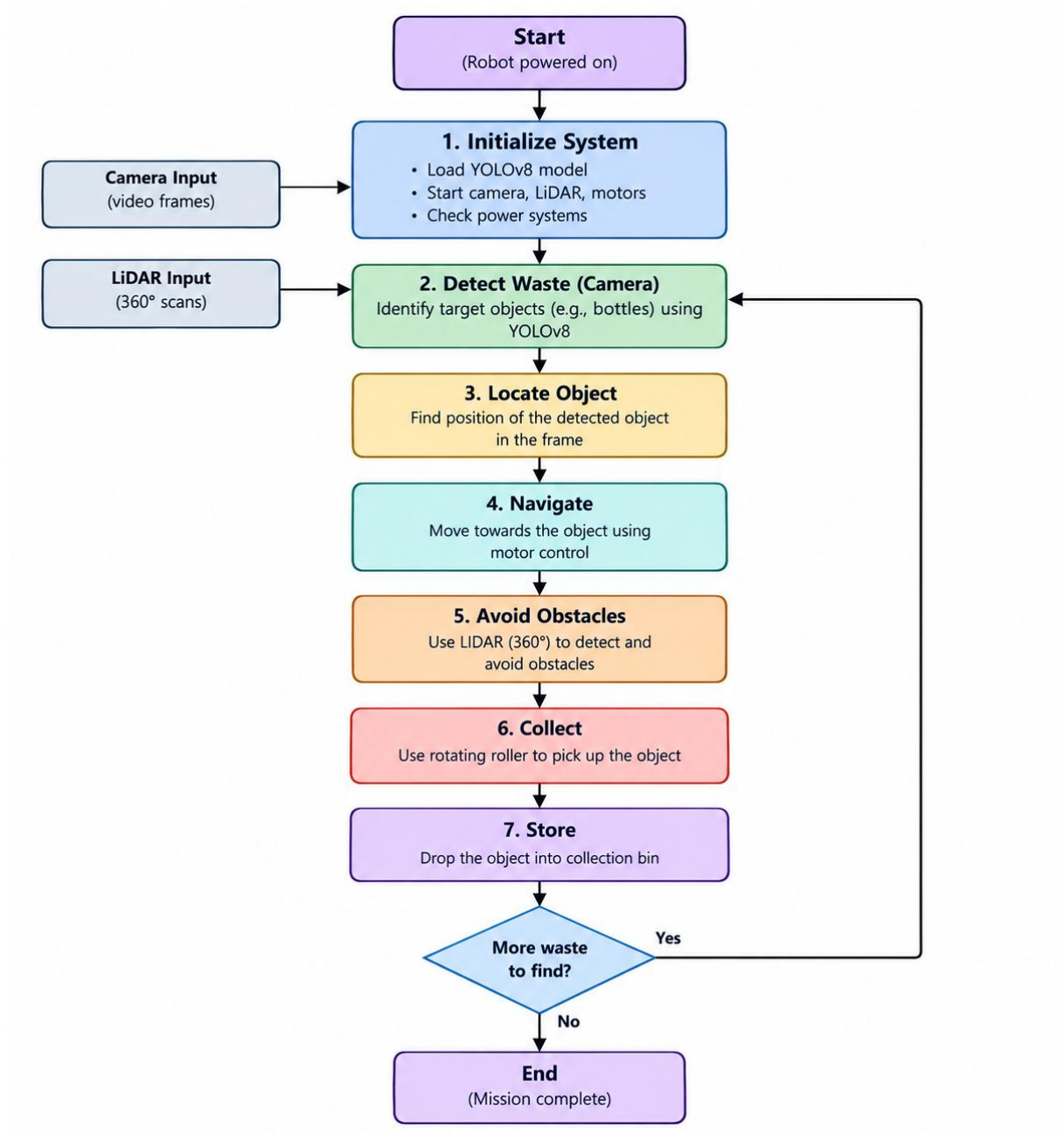


Figure 3 System block diagram

Implementation

The robot is built from six subsystems that meet at the Raspberry Pi: the drive platform, the vision system, the lidar, the intake mechanism, the power architecture and the control software. Below is how each of them was actually built, including the parts that did not work first time.

1. Drive platform

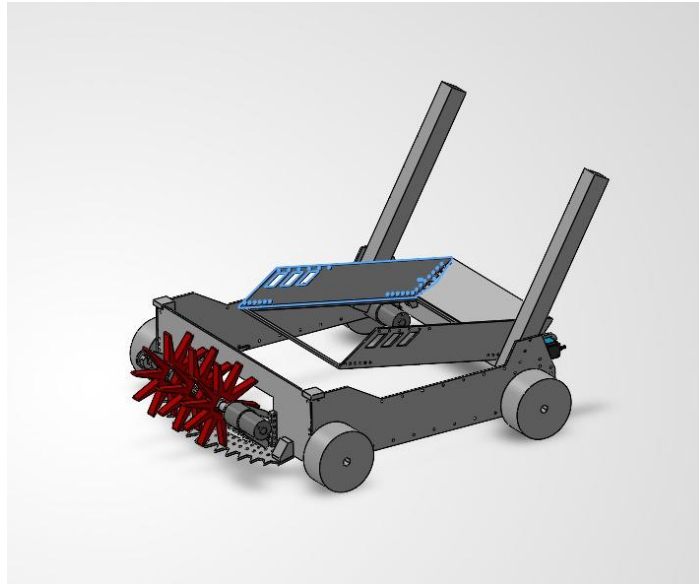


Figure 4 Four-wheel-drive chassis with the four Johnson motors mounted.

The chassis is a four-wheel-drive platform carrying four 200 RPM Johnson geared motors, one per wheel, driven through a motor driver controlled from the Pi's GPIO. I chose four-wheel drive rather than two because sand is a poor traction surface: with two driven wheels, any loss of grip on one side turns into a spin rather than a course correction. Driving all four wheels spreads the tractive effort and gives the robot a chance of pulling itself out of a soft patch rather than digging into it.

Steering is differential - there is no steering rack. To turn, the software drives the wheels on one side faster than the other; to turn on the spot, it drives them in opposite directions. This keeps the mechanical design simple and means the only thing the control software has to produce is a pair of left/right speed values. The 200 RPM gearing was chosen as a compromise: fast enough to cover ground, geared down enough to produce the torque needed to move a loaded robot through sand rather than spinning the wheels.

2. Vision system and detection-to-action



Figure 5 USB webcam used for the vision system.

A USB webcam is mounted at the front of the robot, angled down so that its field of view covers the ground immediately ahead the region the roller will sweep rather than the horizon. The camera feeds frames to a custom-trained YOLOv8 model running on the Pi.

The important part is what happens after a detection. YOLO returns, for each detected object, a bounding box, a class label, and a confidence score (Ultralytics 2023). The control software takes the box for the highest-confidence target and computes the horizontal offset of the box's centre from the centre of the frame. That offset is the steering error:

- If the target center is to the left of the frame center, the robot turns left.
- If it is to the right, the robot turns right.
- If it is within a central dead-band, the robot drives straight ahead.
- The size of the bounding box grows as the robot closes on the object, which is used as a proxy for proximity and triggers the roller.

This is deliberately a simple control law. I considered proportional steering, where the turn rate scales with the size of the offset, and it is the obvious improvement, but the inference rate on the Pi is low enough that a smooth proportional controller running on stale frames oscillated more than the crude three-state version did. That is a limitation of the platform rather than the idea, and it is on the improvement list.

3. Lidar and obstacle avoidance



Figure 6 RPLIDAR A1

An RPLIDAR A1 is mounted on top of the robot with a clear 360° view. It scans at 5.5 Hz over a 0.15 m to 12 m range with angular resolution of $\leq 1^\circ$, communicating over UART (SLAMTEC 2026). The software does not build a map; full SLAM would be well beyond what the Pi can manage alongside inference. Instead, it takes each scan, extracts the readings within a forward-facing arc, and checks whether the minimum distance in that arc has dropped below a threshold.

Keeping the camera's job and the lidar's job strictly separate was one of the better decisions I made. The camera answers "where is the target?" The lidar answers "is something in my way?" They are different questions with different failure modes, and when I first tried to have the camera also handle obstacles — by treating large, unrecognized objects as things to avoid — the logic became tangled and the robot started avoiding the litter it was supposed to be collecting. Two sensors, two questions, two independent code paths.

4. Roller intake and the TPU component

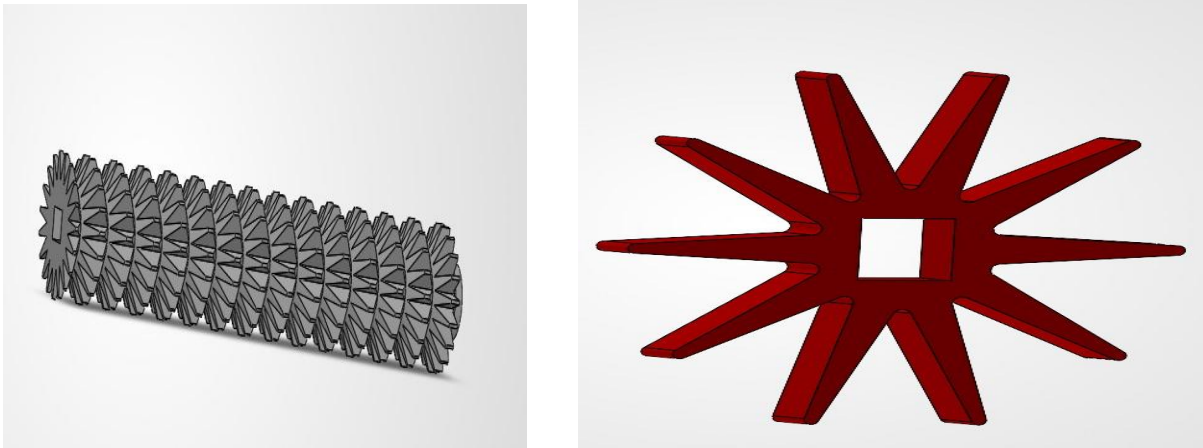


Figure 7 Intake

The front intake is a powered roller mounted across the width of the robot, just above the ground, rotating so that its lower surface moves rearward the same direction a conveyor would run to pull something in. Objects that the roller contacts are swept up the ramp behind it and drop into the collection bin.

The part that took the most iteration is the flexible element on the roller, which is 3D printed in TPU. My first version was printed in PLA, which was a mistake I should have predicted: a rigid roller contacting a rigid bottle just pushes the bottle away. The bottle skitters forward, the robot chases it, and the whole cycle repeats until the bottle reaches something it can be pinned against. TPU solves this because it deforms. The flexible fingers wrap slightly around the object instead of striking it, which converts a push into a grip.

Print settings took several attempts. TPU is notoriously awkward to print — it is flexible enough to buckle in a Bowden-style extruder, so it needs slow speeds and careful retraction settings. I also had to tune the stiffness by adjusting wall thickness rather than material: too soft and the fingers just fold over the object without lifting it, too stiff, and I was back to the PLA problem. The exact print parameters and roller geometry are recorded in the engineering log and should be read from the CAD model rather than estimated.

One design constraint I built deliberately: the roller is set to skim the surface rather than dig into it. A scoop or a rake lifts sand; the roller is positioned so that it contacts objects sitting on the sand without penetrating the surface layer. This is the mechanical half of the selectivity argument — the perception layer decides what to go to, and the intake geometry decides how much of the beach comes with it.

5. Power architecture



Figure 8 LiPo Battery

The robot runs on two separate LiPo batteries: one supplying the four drive motors and the roller motor through the motor driver, and one supplying the Raspberry Pi and its peripherals.

This was not the original design. The first build ran everything from a single pack, and the first time all four motors started simultaneously, the Pi rebooted. Four geared motors under load draw a large, sharp current spike, and the resulting voltage sag on the shared rail was enough to drop the Pi below its brownout threshold — the Pi 4 needs a minimum of 3 A at 5 V (Raspberry Pi Ltd 2026), and it is unforgiving about dips. Losing the computer at the exact moment the robot starts moving is about the worst possible failure mode, because the motors can keep running while nothing is telling them to stop.

Separating the supplies fixed it completely. The motor pack can sag as much as it likes under load without the compute side noticing. It costs extra weight and a second charging routine, and it is worth both. The exact battery chemistry, capacity, regulation and measured runtime are recorded in the engineering log; runtime under realistic load is one of the measurements still outstanding.

6. Control software and interface

All of the coordination happens on the Pi, in Python. The main loop pulls a frame from the camera, runs inference, extracts the best detection, converts it to a steering command, checks the latest lidar scan for obstacles, and issues motor commands accordingly. Obstacle avoidance takes priority over target approach: if the lidar reports something inside the threshold distance, the avoidance behaviour overrides the steering command from the camera, and target approach resumes once the path is clear.

Getting the Ultralytics stack running on the Pi at all was a significant piece of work. The Pi runs 64-bit Raspberry Pi OS, and the environment threw a numpy / OpenBLAS symbol conflict — an undefined symbol error on import — that prevented the library from loading. Resolving it meant rebuilding the Python virtual environment and force-reinstalling numpy against the correct BLAS build. It is the kind of problem that is invisible in a tutorial and consumes an entire afternoon in practice.

A 5-inch display mounted on the robot shows the live camera feed with detection overlays and the current state of the control loop. This is genuinely useful rather than decorative: when the robot behaves oddly, being able to see what the model is seeing at that moment is the fastest way to work out whether the problem is perception, decision-making or mechanics.

Hardware Components

This section sets out each major component, why it was chosen over the alternatives, and what it contributes to the system.

Component	Key specification	Why it is in this robot
Raspberry Pi 4 Model B	Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8 GHz; LPDDR4 SDRAM; 2 × USB 3.0 and 2 × USB 2.0; 40-pin GPIO header; 2-lane MIPI CSI camera port; 5 V DC via USB-C, minimum 3 A (Raspberry Pi Ltd 2026)	The central controller. It is the only single board in this price range that can run a neural network, read a lidar over serial, drive GPIO motor control and output to a display simultaneously. A microcontroller could not run inference; a laptop would not fit on the robot.
Logitech USB webcam	USB plug-and-play webcam, front-mounted and angled downward to cover the roller's sweep area (exact model to be confirmed from the build log)	The robot's eyes. A plain USB webcam was chosen over a Pi camera module or a depth camera because it is robust, needs no ribbon cable running to a moving front assembly, and works in bright outdoor light where depth cameras struggle.
RPLIDAR A1	360° scan; 0.15 m–12 m measuring range; 8K samples/second; 5.5 Hz scan rate; ≤1° angular resolution; UART serial at 3.3 V; 5 V system, 0.5 W (SLAMTEC 2026)	Obstacle detection. Gives distance information across the full circle regardless of lighting, which a camera cannot do. Its job is deliberately narrow: detect obstructions, not targets.
4 × Johnson geared motors	200 RPM geared DC motors, one per wheel, in a four-wheel-drive configuration with differential steering	Traction and steering. Four driven wheels rather than two because sand is a low-grip surface where a single slipping wheel turns into a spin. The 200 RPM gearing trades top speed for the torque needed to move a loaded robot through sand.
Motor driver	Dual-channel H-bridge driver controlled from the Pi's GPIO, supplied from the motor battery	The interface between logic and power. GPIO pins cannot supply motor current; the driver translates low-current direction and speed signals into the high-current switching the motors need.
Roller intake with TPU element	Powered roller across the robot's width, with a flexible 3D-printed TPU sweeping element; positioned to skim the surface rather than penetrate it	The collection mechanism. TPU rather than a rigid plastic because a rigid roller pushes objects away instead of gripping them. Skimming rather than digging is what keeps sand out of the bin.
5-inch Raspberry Pi display	5-inch screen mounted on the robot showing the live camera feed with detection overlays and control-loop state	Onboard diagnostics. When the robot misbehaves, seeing what the model sees at that moment is the quickest way to tell a perception failure from a mechanical one.
2 × LiPo batteries	Separate packs: one for the motor system, one for the Raspberry Pi and peripherals (capacities and measured runtime recorded in the engineering log)	Power isolation. A single shared pack caused the Pi to brown out and reboot when all four motors started at once. Splitting the supplies removed that failure mode entirely.

Cost and Scalability

I would keep it very factual:

One of the goals of Beach Mitra is to explore whether autonomous beach-cleaning technology can be made more accessible than highly specialised commercial systems. However, affordability cannot be assessed without considering the actual cost of construction.

The current prototype consists of commercially available computing, sensing, motor-control and power components together with custom 3D-printed mechanical parts. The total prototype cost will therefore be calculated from the component-level bill of materials, separating reusable components from prototype-specific expenses.

The cost analysis will consider:

- electronics and computing;
- camera and lidar;
- motors, wheels and chassis;
- batteries and power-management components;
- collection mechanism;
- 3D-printed components and materials;
- wiring and other consumables.

The more important question is the **cost per deployable machine**, rather than simply the cost of the first prototype. If four machines can be built using the same core design, the project will also provide an opportunity to determine whether the design can be reproduced without a proportional increase in development effort.

Estimated cost per additional machine: ₹20,000

Estimated cost for four machines: ₹60,000

Artificial Intelligence and Computer Vision

Why YOLOv8

YOLOv8 is an object-detection model family released by Ultralytics in January 2023, available in five sizes from nano to extra-large, and supporting detection, segmentation, classification, pose estimation and oriented detection (Ultralytics 2023). The nano variant, YOLOv8n, has 3.2 million parameters — small enough to be a realistic proposition on a CPU-only single-board computer, which was the hard constraint here.

What makes object detection the right family of model for this problem, as set out in the approach comparison, is that its output is a bounding box rather than a label. A box has a position, and a position can be turned into a motor command. That single property is what allows a neural network to steer a robot.

Training the custom model

A general-purpose model trained on a standard dataset is not much use here. Common object-detection benchmarks contain a "bottle" class, but those bottles are photographed indoors, upright, on tables, under artificial light. The objects this robot has to find are lying on sand, often crushed, often partly buried, in harsh directional sunlight with strong shadows. So the model was trained on a purpose-assembled waste dataset.

The dataset used for training comprised approximately 3,500 training images and 300 testing images, roughly a 92/8 split across about 3,800 images in total. Training was carried out off-robot, on a machine with more compute than the Pi, and the trained weights were then exported and deployed to the Pi for inference. This division of work (training the model on a powerful computer, then running the finished model on the robot itself) is standard practice and was essential here, because the Pi does not have enough computing power to train a model but can run a trained one in real time.

The project-reported accuracy of the trained model is approximately 75%. I want to be precise about what that figure does and does not mean, because it is the number most likely to be over-interpreted. It is the accuracy figure reported for the trained model; it has not been decomposed into the standard object-detection metrics. In particular it should not be relabelled as mAP50-95, precision or recall without going back to the training output and checking, because those metrics measure genuinely different things and are not interchangeable. Ultralytics reports YOLOv8 performance as mAP at 50–95% IoU on COCO (Ultralytics 2023), and reporting my model on the same basis is a specific item on the outstanding work list.

75% is a reasonable result for a first custom model on a hard visual problem, and it is also clearly not good enough for deployment. One missed detection in four is one piece of litter in four left on the beach. The honest framing is that the perception layer currently works well enough to prove the concept and not well enough to do the job, and improving it is the highest-value next step in the whole project.

From detection to action

The chain from pixel to wheel is short and worth stating explicitly, because it is the core of the project:

1. The camera captures a frame of the ground ahead of the robot.
2. YOLOv8 runs inference on the frame and returns any detections, each with a bounding box, a class label and a confidence score (Ultralytics 2026).
3. The software discards detections below a confidence threshold and selects the highest-confidence remaining target.
4. The horizontal offset between the box centre and the frame centre becomes a steering error.
5. That error is converted into left and right wheel speeds.
6. The bounding box growing beyond a size threshold indicates the robot has closed on the object, and the roller is engaged.

Step 4 is the interesting one. It is the point where a statistical prediction about an image becomes a physical instruction to a machine, and everything downstream of it - every safety consideration, every ethical question in this report follows from the fact that this step exists.

Testing and Evaluation

Testing so far has been at subsystem level, on a bench and on a prepared sand tray. Preliminary supervised testing has also been carried out on the beach (Figure 9). I have separated the two clearly below: the first table records tests I have run and what happened, and the second sets out the field-testing plan with its success criteria defined in advance.

Test conditions

Testing was conducted in three environments: indoor testing, a prepared sand tray, and supervised beach testing. Tests were conducted using bottles, wrappers and bottle caps as representative plastic targets. Beach testing was conducted under clear outdoor conditions at Silver Beach, Mumbai. The robot was supervised throughout all field testing and an operator remained able to intervene.



Figure 9 Beach Mitra on the sand at Silver Beach, Mumbai, during supervised beach testing.

Bench and sand-tray tests completed

Test	What I did	Result and what it told me
Live inference on the Pi	Ran the trained model on the live USB camera feed on the Pi and confirmed detections appeared on the onboard display.	Works. Detections appear reliably on clearly visible bottles at close range. Frame rate is low enough that it constrains how fast the robot can safely drive this is the main compute bottleneck, and it is a hardware limit, not a code fault.
Detection under varied lighting	Presented the same bottles indoors, in shade, and in direct sunlight.	Noticeably works in direct sunlight and against strong shadows. Glare on transparent plastic is the specific failure a clear bottle catching the sun sometimes disappears from the model's view entirely. This is a dataset problem and confirms that training images need to come from real beach conditions.
Motor direction and drive control	Commanded each motor individually, then in drive and differential-turn combinations.	All four motors respond correctly in both directions. Differential turning works on hard ground and on the sand tray.
Power isolation	Started all four motors simultaneously with a single shared battery, then repeated with the split supply.	Single supply: the Pi browned out and rebooted. Split supply: no effect on the Pi at all. This test is the reason the power architecture is what it is.
Lidar scan integrity	Ran the RPLIDAR continuously and inspected the scan output with obstacles placed at measured distances.	Clean, consistent 360° scans. Obstacle distances match measured values within the accuracy the datasheet claims. No dropped scans over the test period.

Test	What I did	Result and what it told me
Obstacle response	Placed a static obstacle in the robot's forward path during a drive command.	The robot detects the obstacle and changes behaviour before contact. The avoidance manoeuvre itself is crude it does not yet resume the original heading elegantly, which is on the improvement list.
Roller pickup on hard ground	Drove the robot at bottles, caps and wrappers on a hard flat surface.	Reliable on bottles and wrappers. Small flat items like caps are less consistent because they can slip under the roller rather than being lifted by it.
Roller pickup on sand	Repeated the pickup test on a prepared sand tray.	Works, but noticeably worse than on hard ground: partly buried objects are frequently pushed rather than lifted, and some sand enters the bin. This confirmed that burial depth is a real limiting variable.
TPU vs PLA roller element	Ran identical pickup attempts with the rigid PLA element and the flexible TPU element.	Decisive. PLA pushed objects away in most attempts; TPU gripped and lifted them. This is the single clearest before-and-after result in the project.

Five approaches were tested with the robot starting approximately 25m from the target. Four approaches successfully reached the target without manual steering, giving an 80% success rate. Mean approach time was 5 seconds.

Testing video and behind the scene images:

<https://drive.google.com/drive/folders/1ALuVnbmgb9BAHOBvAcG8gFeOxBY66Hht?usp=sharing>

Evaluation Against Original Success Criteria

Overall evaluation

The testing shows that the individual components of Beach Mitra are functioning, but that the system is not equally reliable across all parts of the chain. The drive platform, lidar and basic collection mechanism have produced repeatable results under controlled conditions. The main limitation remains the perception layer, particularly in direct sunlight and with transparent or partially buried plastic. The field trials are therefore important not simply to demonstrate that the robot works, but to measure how reliably the complete system works under the conditions it was designed for.

What the results actually say

Perception is the bottleneck, not the mechanics

I expected the hard part of this project to be mechanical - that making something physically pick up a bottle would be the wall I hit. It was not. The roller worked, after the TPU change, more reliably than anything else on the robot. The limiting factor throughout has been perception: 75% accuracy, degraded further by glare on transparent plastic in sunlight, running slowly enough that the control loop has to make decisions on frames that

are already out of date. Almost every behavioral weakness in the robot traces back to the vision system rather than to the hardware it drives.

The two-battery decision was the highest-value fix

Splitting the power supplies took an afternoon and eliminated an entire class of failure. Before it, the robot was unreliable in a way that looked like a software bug — it would sometimes just stop responding, and I spent time looking at the code for a fault that was in the power rail. This is the single most transferable lesson from the project: when a computer behaves strangely under load, suspect the power supply before the program.

Choosing forgiving mechanisms compensates for imperfect control

The roller's tolerance for positioning error is doing a lot of quiet work. Because it sweeps a band rather than needing a precise grasp point, the oscillating approach, which would completely defeat a gripper arm — still usually ends with the object being collected. Designing the mechanism to be forgiving was what allowed the rest of the system to be imperfect and still function, and I would apply that principle again.

Separating sensor responsibilities simplified everything

Giving the camera and the lidar strictly different questions to answer made the control logic dramatically easier to reason about and debug. When I briefly tried to make the camera handle both target-finding and obstacle-detection, the two behaviours interfered and the robot started avoiding the litter it was meant to collect. One sensor, one question.

Resources: Materials and People

A project that spans deep learning, embedded Linux, mechanical design, 3D printing and power electronics is not one that a single person walks into already knowing. Being realistic about which parts I could not do alone, and asking early rather than after weeks of being stuck, was probably the most useful project-management decision I made.

Sourcing and materials

Component sourcing shaped the build more than I expected. Several parts were either out of stock locally or only available from suppliers with long lead times, which meant the build order was partly determined by what arrived when rather than by what was logically next. TPU filament in particular took some searching — flexible filament is a specialist item compared to PLA, and the specific shore hardness matters a great deal for whether the roller fingers grip or fold.

I also had to be honest about budget. Every component in the robot was chosen partly on the basis that it is affordable and widely available, which is also the reason the design is reproducible. A depth camera would have simplified the approach-and-stop logic considerably, and I rejected it partly on technical grounds (depth sensing performs badly in direct sunlight but also because it would have roughly doubled the sensing cost of the robot for a benefit I could work around in software).

People I learned from

- Mentors with embedded systems experience helped me diagnose the brownout problem. I had been treating it as a software fault; the suggestion to measure the supply rail under motor load, rather than to keep reading the code, was what actually solved it.
- Guidance on 3D printing flexible filament saved me a considerable number of failed prints. The advice to tune stiffness through wall thickness rather than by hunting for a different material was what made the roller work.
- Advice on deploying machine-learning models to constrained hardware clarified the train-off-device, infer-on-device split and set my expectations about what frame rate was realistically achievable on a CPU-only board, which stopped me from assuming a low frame rate meant I had done something wrong.
- Documentation and open-source communities were a resource in their own right. The numpy /CVconflict that blocked the entire project for a day was resolved through a community-reported fix, not through anything I worked out myself.

What I took from the mentoring was not only the technical answers. The more valuable part was watching how experienced engineers approach a problem: they isolate before they theorise. Every time I described a symptom and jumped to a cause, the response was some version of "how do you know?" —which is a habit I did not have at the start of this project and do now.

External Feedback

Beach-cleaning volunteer / NGO representative " *During a clean-up, finding the waste is often more time-consuming than picking it up* " — Khushiyaan Foundation, 24th May 2025

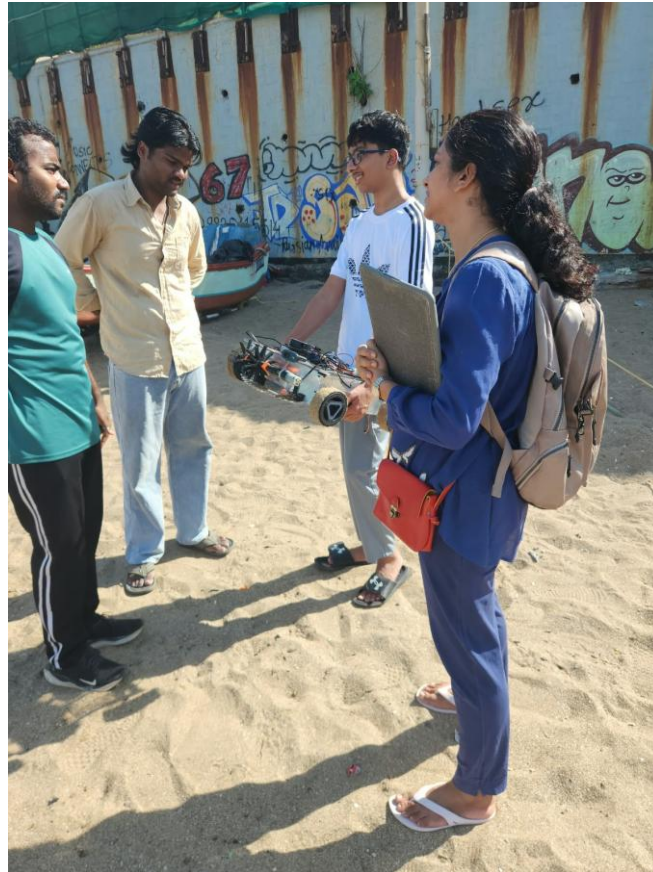


Figure 10 Showing Beach Mitra to local beach cleaners and discussing how it could support their clean-up work.

Ethical, Environmental and Safety Considerations

A robot designed to protect an environment has an obligation not to damage it, and a robot with a powered rotating mechanism has an obligation not to injure anyone. Neither of these is a formality here both changed design decisions.

Environmental responsibility

The central ethical claim of this project is selectivity: that a vision-guided robot removes litter without stripping the strandline material that beach ecosystems depend on. That claim places a duty on me to make sure it is true rather than just plausible.

- Wildlife is the most serious risk. A roller cannot tell the difference between a bottle and a crab, so the responsibility falls entirely on the perception layer and on the operating rules. The robot must never be run unsupervised near nesting areas, and early trials are supervised throughout with an operator able to stop it immediately.
- Strandline material—seaweed, driftwood, and shells—is not waste. Beach wrack acts as a cross-boundary subsidy linking the marine and terrestrial systems, feeding the invertebrates that shorebirds depend on and supplying the nutrients that let dune vegetation establish, so removing it limits both habitat and dune formation (Wikipedia contributors 2025c). The model is trained only on waste classes, and the roller is set to skim rather than dig, so the intended behavior is to leave natural material in place. The selectivity test in the field plan exists specifically to verify this rather than assume it.
- Sand removal should be minimized. Anything lifted into the bin that is not litter is beach material being relocated, and the roller height is set with that in mind.

- Protected dunes and environmentally sensitive areas are excluded from operation unless specifically permitted. Dune vegetation is exactly what mechanical beach cleaning damages, and a robot with wheels can damage it too.
- Disturbance matters as well as contact. A machine moving through a habitat is a disturbance even when it touches nothing, which is another reason for supervised, time-limited trials rather than continuous unattended running.

I think it is worth stating the limit of the claim plainly. This robot is a targeted collection prototype, not a general-purpose machine for removing everything from a beach. Describing it as the latter would overstate what it does and would invite exactly the indiscriminate use that makes mechanical cleaning a problem in the first place.

Safety

- The rotating roller is a pinch hazard. It is guarded during demonstrations and maintenance, and the robot is never handled with the roller powered.
- An emergency stop and power-disconnect method must be available and tested before the robot operates anywhere the public can reach it. This is a prerequisite for field trials, not an optional extra.
- LiPo batteries require respect: appropriate charging, secure mounting so they cannot shift or be punctured in a collision, and proper storage. They are the highest-energy component on the robot.
- A four-wheel-drive robot with geared motors has enough torque to hurt fingers and to keep driving into an obstacle if the software fails. The lidar's obstacle threshold is the first line of defence and the manual stop is the second.
- Early field trials are supervised at all times, with a defined area and an operator watching. Autonomy is a design goal, not an excuse to walk away from a moving machine.
- Wiring is secured and the electronics enclosed, both for the robot's protection against sand and moisture and to remove exposed conductors.

Hazard	Likelihood	Severity	Mitigation
Rotating roller/pinch point	Medium	High	Guard roller; power off during handling
Robot collision	Medium	High	LiDAR obstacle detection + manual stop
LiPo battery damage	Low	High	Secure mounting, controlled charging/storage
Wildlife interaction	Low/Medium	High	Supervised operation; avoid nesting/sensitive areas
Sand/moisture entering electronics	Medium	Medium	Enclosed electronics and secured wiring
Public entering test area	Medium	High	Controlled test area and continuous supervision

An emergency-stop/power-disconnect mechanism will be required before public deployment.

Data and transparency

The camera records images of a public space. In practice the camera points down at the ground immediately ahead of the robot, so it captures sand rather than people, but this is a design property worth stating rather than a happy accident and it is a reason to keep the camera angled down rather than forward. No imagery collected during testing is published in a form that identifies any individual.

There is also a transparency obligation around the results themselves. A reported accuracy of 75% that has not been decomposed into standard metrics is a number that could easily be quoted as though it means more than it does. Stating clearly what has been measured, what has not, and which claims remain expectations is part of doing this responsibly.

Reflection and Improvements

Nothing in this project went the way I planned it, which I am told is normal and which I found deeply irritating at the time. These are the problems that changed the design.

The Pi kept rebooting and I blamed the wrong thing

The first time all four motors ran together, the Pi shut down and restarted. I spent a genuinely embarrassing amount of time looking for a software fault reading through the control loop, adding print statements, suspecting the motor library before someone suggested measuring the supply voltage under load. It was a power problem the whole time. Four geared motors starting simultaneously pull a current spike large enough to sag a shared rail below the Pi's brownout threshold, and the Pi 4 wants a minimum of 3 A at 5 V (Raspberry Pi Ltd 2026). Two separate batteries fixed it permanently. The lesson I actually took away is about diagnosis, not power: I had assumed the cause was in the part of the system I understood best, which is exactly the wrong instinct.

A rigid roller pushes bottles away

My first intake was printed in PLA because PLA is easy to print. It did not work, and in hindsight the reason is obvious: a rigid cylinder meeting a rigid bottle transfers force forward, so the bottle simply rolls ahead of the robot. Watching this happen repeatedly was what made me think about compliance as a design property rather than an inconvenience. Printing the sweeping element in TPU changed the interaction completely - the flexible fingers deform around the object and lift it instead of striking it. Getting the stiffness right took several iterations, tuned by changing wall thickness rather than by changing material.

The software would not even import

Deploying the model to the Pi was blocked by a numpy symbol conflict that caused the Ultralytics import to fail outright. This is the kind of problem that does not appear in any tutorial, has nothing to do with machine learning, and consumes an entire working day. Rebuilding the virtual environment and force-reinstalling numpy against the correct build resolved it. It taught me that in applied ML, the environment is part of the engineering, and that "it works on my laptop" is not a deployment.

Sunlight breaks the model in a specific way

Detection is noticeably worse outdoors than indoors, and the failure is specific rather than general: transparent plastic catching direct sunlight can disappear from the model's view completely. Glare turns a bottle into a bright shapeless region that does not resemble anything in the training data. This is a dataset problem, not a model-architecture problem, and it means the single most valuable thing I can do next is collect training images on actual beaches in actual sunlight rather than train on more of the same kind of image.

Small flat objects slip under the roller

Bottle caps and flat fragments are collected far less reliably than bottles, because they can pass beneath the roller instead of being lifted by it. Since fragments are a large proportion of real beach litter, this is a meaningful gap. It is a geometry problem roller height, finger length and the ramp angle into the bin all interact and it needs a deliberate iteration rather than a guess.

Skills Acquired

Some of these I set out to learn. Others I learned because the project refused to work until I did.

1. Training and deploying a deep-learning model. I went from having no practical experience of object detection to assembling a dataset, training a custom YOLOv8 detector, exporting the weights, and getting live inference running on a single-board computer. The deployment half turned out to be harder and more interesting than the training half, which is not what I expected going in.
2. Embedded Linux and dependency management. The numpy/CV conflict forced me to understand what a virtual environment actually does, why a library compiled against one numerical backend fails against another, and how to read an import error properly instead of copying it into a search box and hoping. This is unglamorous, and it is the difference between a model that runs on a robot and one that does not.
3. Electronics and power system design. I learned the practical difference between a supply that provides enough average current and one that survives a transient, why motor and logic supplies are separated in real designs, and how to diagnose a brownout. Before this project I would not have thought of the power rail as a place bugs come from.
4. CAD and design for 3D printing. Designing the roller taught me to think about a part in terms of the forces it will experience and the material it will be printed in, not just its shape. Working in TPU specifically taught me that material properties are a design variable compliance was the solution to a problem that geometry alone could not fix.
5. Sensor integration. Interfacing a lidar over serial, parsing scan data, and turning a 360° point cloud into a single actionable decision was new to me. The broader lesson was architectural: give each sensor one clear question to answer and the software stays comprehensible.
6. Control logic and real-time constraints. Writing a loop that has to capture, infer, decide, and actuate within a time budget—and watching what happens when it does not—taught me more about latency than reading about it would have. The oscillating approach is latency made visible.
7. Systematic debugging. This is the skill I improved most. I started the project assuming the fault was wherever I had most recently changed something, and I finished it isolating subsystems and measuring before theorising. The brownout was the turning point.
8. Honest technical reporting. Deciding to write "not yet tested" rather than quietly rounding an expectation up into a result was uncomfortable and is, I think, the most important thing in this report. Overstating a result is not just dishonest, it is useless it removes the information about what still needs doing.
9. Research and evaluation of sources. Building the case for why this robot should exist meant reading actual assessments of marine litter and the ecological effects of beach grooming, and distinguishing between what a source measured and what it inferred. That habit fed directly into how I reported my own results.
10. Project and time management. 96 hours across eight phases, with component lead times that did not respect my plan, taught me to sequence work around what can be done now rather than around what comes next logically.

The meta-skill underneath all of these is that autonomous robotics is an integrated discipline. A brilliant detector on a robot that browns out is worth nothing. A perfect intake mechanism that the vision system never guides to a target is worth nothing. The system is only as good as its weakest link, and the weakest link is rarely the part you were most excited to build.

What I Learned

The technical skills are the easiest things to list. I can point to the model I trained, the code I wrote, the roller I designed and the problems I eventually fixed. The less obvious things I learned are probably more important.

I learned that building something for a real environment is different from building something that works in isolation. On a bench, I can control the lighting, the surface, the objects and the distance from the robot. On a beach, I control almost none of those things. The sand changes, the sunlight changes, the litter changes and there are things moving that I did not put there. Designing for the real world means accepting that uncertainty rather than trying to eliminate it.

I also learned to become more comfortable with being wrong. My first roller design was wrong. My first power architecture was wrong. My assumption about the source of the Pi's crashes was wrong. None of those failures were particularly enjoyable while they were happening, but each one gave me information that I could use. I started the project wanting to get the design right. I finished it understanding that getting to the right design is more important than getting the first design right.

The project also changed how I think about asking for help. Earlier, I tended to see asking someone else as admitting that I had reached the limit of what I could do. Working on this project showed me that experienced engineers often make progress faster not because they know every answer, but because they know what question to ask. The advice to measure the power rail, for example, saved far more time than another afternoon of debugging code.

I learned patience as well. Some of the least interesting problems consumed the most time: a dependency conflict, a failed print, a component that was unavailable, a battery that was not suitable. None of these felt like progress when I was dealing with them. Looking back, they were part of the project rather than interruptions to it.

Most importantly, I learned why I enjoy building things. There is a particular satisfaction in watching something that existed as an idea become a physical machine, and then watching that machine do something useful. The robot is not finished, and there are still obvious things it cannot do reliably. But it can detect something, make a decision and physically act on that decision. Seeing that happen for the first time made the hours of failed prints, debugging and redesign feel worthwhile.

The biggest lesson I am taking forward is that engineering is not just about making something work. It is about understanding why it works, knowing when it does not, and having enough curiosity to keep improving it. Beach Mitra has given me a starting point for that.

Next Steps and Deployment

The next stage of Beach Mitra is to move it from a prototype that I have built and tested into a system that can be tested in the environment it was designed for. The important question now is not whether each individual subsystem can work. I have demonstrated that they can. The question is whether the complete system can operate reliably on a real beach, repeatedly, without damaging the environment it is intended to protect.

1. Complete real-world field testing

The first priority is to complete supervised testing on an actual beach. The recent tests have already shown that the robot behaves differently on real sand and in outdoor conditions than it does on a controlled surface. Direct sunlight, glare from transparent plastic, partially buried objects and the unevenness of sand all affect the system in ways that are difficult to reproduce indoors.

The next tests will therefore measure the complete system rather than individual components. I want to record detection rate, successful approaches, collection rate, missed objects, false detections, human interventions,

battery runtime and the number of complete *detect* → *approach* → *collect cycles* completed without intervention.

The testing also needs to answer the environmental question that motivated the project in the first place. The robot should collect the plastic it is looking for without collecting seaweed, driftwood or other natural strandline material. This needs to be measured rather than assumed. The field trial will therefore include a deliberate test of selectivity, with the robot operating around natural beach material under supervision.

The biggest improvement to the AI model will come from the field itself. The current model performs noticeably worse with transparent plastic in direct sunlight, which means the next dataset should contain more images from actual beaches, including different lighting conditions, viewing angles, levels of burial and types of plastic. Improving the model is therefore not simply a matter of training for longer; it requires better data representing the environment in which the robot will actually operate.

2. Working with an NGO

Once the basic field testing is complete, I want to work with an NGO involved in beach clean-up to test whether Beach Mitra is useful outside my own testing environment. This is an important step because I have designed the robot based on a problem I observed, but the people who regularly clean beaches understand practical constraints that I cannot learn from a workshop or a sand tray.

The aim would be to begin with supervised deployment rather than treating the robot as a finished autonomous product. The NGO would be able to identify suitable areas for testing, provide feedback on the types of waste that matter most to them, and help evaluate whether the robot actually reduces the time people spend searching for litter.

I also want the deployment to generate better requirements for the next version. A machine that works technically but is difficult to transport, clean, charge, maintain or operate is not a useful field machine. Feedback from people using it for an actual clean-up will therefore be treated as part of the engineering process rather than as an afterthought.

3. From one prototype to four machines

The longer-term goal is to build four Beach Mitra machines and test them as a small fleet. Building one working prototype proves that the idea is possible. Building four will test something different: whether the design is repeatable.

The four machines should use the same core architecture — *Raspberry Pi, camera, lidar, four-wheel drive and roller intake* — while incorporating the improvements identified during field testing. I would like to compare their performance rather than assuming that a successful prototype automatically produces four equally reliable machines.

This will also introduce a new set of engineering problems. Components will need to be mounted consistently, wiring will need to be robust, batteries will need to be managed safely, and the software will need to behave predictably across multiple machines. Maintenance and cleaning will become more important when the robots are operating regularly on sand rather than being tested one at a time.

The goal is not simply to have four robots. It is to find out whether Beach Mitra can become a repeatable system that can actually contribute to a beach-cleaning operation.

4. What success would look like

I do not want to define the next stage simply as "*making the robot better*." I want to be able to answer a few specific questions.

Can the robot reliably find the types of plastic that matter in real beach conditions? Can it collect them without taking significant amounts of sand or natural material with them? Can it operate for a useful period without human intervention? Can the same design be reproduced four times and still perform consistently? And, most importantly, does it actually reduce the amount of time people need to spend searching for litter?

If the answer to some of these questions is no, that will not mean the project has failed. It will tell me what the next engineering problem is.

Beach Mitra has reached the point where the next lessons will come less from building another component in isolation and more from putting the complete system into the environment it was designed for. The next stage is therefore about moving from **"Can I build it?"** to **"Can I make it useful?"**

Conclusion

This project set out to test whether an autonomous robot could find plastic waste on a beach, drive to it, and pick it up selectively, without stripping the natural material that beach ecosystems depend on. The answer, as far as I have been able to test it, is a qualified yes.

Every subsystem works. The four-wheel-drive platform drives and steers on sand. A custom-trained YOLOv8 model runs live on the Raspberry Pi and detects target waste in the camera feed. The detection-to-steering logic turns a bounding box into a heading and the robot moves toward what it sees. The RPLIDAR A1 detects obstacles and the robot changes behaviour before contact. The roller intake, once its sweeping element was printed in flexible TPU rather than rigid PLA, lifts bottles and wrappers into the collection bin. The split power architecture keeps the compute side stable under motor load.

What has not yet happened is the run that ties all of it together: a complete, uninterrupted detect → approach → avoid → collect → resume cycle on a real beach, with no human intervening. That, and the selectivity test that would demonstrate the robot leaves seaweed alone, are the two outstanding milestones. Until they are done, the central claim of this project is a well-supported expectation rather than a demonstrated result. I would rather leave that distinction visible than claim success that the testing does not support.

I am more interested in what the project revealed than in what it achieved. I expected the mechanical problem to be the hard one, and it was not — the roller was among the most reliable things on the robot. The bottleneck was perception, and specifically the gap between a model that scores acceptably on curated images and a model that works on wet sand in direct sunlight. The next stage is therefore not simply to improve a number on a screen, but to understand how the whole system behaves in the environment it was designed for.

The broader point holds regardless. The reason mechanical beach cleaning damages beaches is not that machines are bad for beaches; it is that we have only built machines that cannot tell a bottle from a piece of kelp. Object detection changes that. A machine that is selective at the perception layer can, in principle, have the speed of automation and the discrimination of a human volunteer at the same time — and that combination does not currently exist at a price that coastal communities can afford. This prototype does not prove that it can be built well. It does show that it can be built.

There is also something more personal that I will take from the project. I enjoyed the process of turning an idea into something physical far more than I expected. Seeing the TPU roller pick up a bottle after the earlier versions simply pushed it away, solving the power problem after the Pi repeatedly rebooted, and watching the robot respond to something it had detected were small moments, but they were the moments that made the project feel real. The machine was no longer just a collection of components and code. It was something I had designed, built, broken and improved.

That is probably why I do not see Beach Mitra as a finished project. I see it as a first step towards building more complicated systems. The next stage is to take the robot to a real beach, work with an NGO, learn from people who actually deal with the problem, and build four machines that can be tested together. Building one prototype showed me that I can take a real-world problem and turn it into a working piece of engineering. Building and deploying several machines will show me whether I can turn that prototype into something genuinely useful.

Acknowledgements

I would like to thank **Legrand India and Khushiyaan Foundation** for organizing the beach-cleaning activity, which inspired us to work towards addressing the problem of beach pollution.

I am also grateful to **The Innovation Story** for providing laboratory facilities, technical mentorship, and guidance throughout the development of this project.

Works Cited

Dugan, Jenifer E., and David M. Hubbard. "Loss of Coastal Strand Habitat in Southern California: The Role of Beach Grooming." *Estuaries and Coasts*, vol. 33, no. 1, 2010, pp. 67–77. <https://doi.org/10.1007/s12237-009-9239-8>.

Dugan, Jenifer E., David M. Hubbard, Michael D. McCrary, and Mark O. Pierson. "The Response of Macrofauna Communities and Shorebirds to Macrophyte Wrack Subsidies on Exposed Sandy Beaches of Southern California." *Estuarine, Coastal and Shelf Science*, vol. 58, suppl., 2003, pp. 25–40. [https://doi.org/10.1016/S0272-7714\(03\)00045-3](https://doi.org/10.1016/S0272-7714(03)00045-3).

Hyndes, Glenn A., et al. "The Role of Inputs of Marine Wrack and Carrion in Sandy-Beach Ecosystems: A Global Review." *Biological Reviews*, vol. 97, no. 6, 2022, pp. 2127–2161. <https://doi.org/10.1111/brv.12886>

Lindegren, Martin, et al. "Impacts of Beach Wrack Removal via Grooming on Surf Zone Water Quality." *Environmental Science & Technology*, vol. 48, no. 4, 2014, pp. 2304–2311. <https://doi.org/10.1021/es405536q>

Raspberry Pi Ltd. *Raspberry Pi 4 Model B Documentation*. Raspberry Pi Ltd., 2026.

SLAMTEC. *RPLIDAR A1 Documentation*. SLAMTEC, 2026.

Ultralytics. *YOLO Documentation*. Ultralytics, 2026.

Ultralytics. *YOLOv8 Documentation*. Ultralytics, 2023.

United Nations Environment Programme. *From Pollution to Solution: A Global Assessment of Marine Litter and Plastic Pollution*. United Nations Environment Programme, 2021.