ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

AI-Intelligent Culinary Device

Team #62

TONY LIU (zikunl2@illinois.edu)
JACKSON BROWN
(jcb10@illinois.edu)
GRIFFIN KELLEY
(gkelley4@illinois.edu)

TA: Aniket Chatterjee
Professor: Viktor Gruev

May 6th, 2026

Abstract

The AI-Intelligent Culinary Device, developed under NUCHEF™, elevates the cooking experience through a fusion of artificial intelligence (AI), virtual reality (VR), and a hardware seasoning dispenser. The Meta Quest VR headset serves as the user interface and sensing platform, using passthrough camera access and local inference tools to locate and classify 14 classes of ingredients through red-green-blue (RGB) and depth sensing [1], [2], [3], [4], [5]. The detected ingredient data, along with ingredient weights measured through a load-cell sensing circuit, are passed to the large language model (LLM) recipe planner to generate tailored step-by-step visual cooking guidance [6], [7].

During recipe guidance, the seasoning dispenser drives three stepper motors to dispense precise amounts of spices while monitoring dispensed mass through the load cell and tracking remaining spice storage using Time-of-Flight (ToF)/infrared (IR) sensing [6], [8], [9]. Communication between devices relies on two primary links: the VR headset connects to the Central Computing Hub, where the LLM planner resides, through Wi-Fi, and the Central Computing Hub continuously fetches sensor and actuator data from the ESP32 [10]. The complete system achieves a latency of $4\,\mathrm{ms}$ and seasoning dispensing error below $1\,\mathrm{g}$. Our team aims to provide a user-friendly culinary experience through this AI-powered ECE 445 project.

Contents

1 Introduction

A general section looks like this. There is usually a blurb introducing the top-level section here.

1.1 Problem

The processed food industry has become increasingly toxic due to chemical flavor additives (12%-32% higher cancer risk), yet getting into cooking for oneself often intimidates new chefs. Therefore, students and working professionals may rely on convenient but unhealthy meals. Most of the currently available 'smart cooking' tools do not provide a real-time experience that guides users through the process from raw ingredients to the finished dish. This can make it challenging to learn or even get started. It can also be inefficient to design recipes that can be adjusted to the user's available ingredients, requiring a prospective cook to hand catalog everything they have available. Users will waste food, and recipe creation is an expert skill that is difficult to customize. Spices are especially difficult to measure and control in a dish while often being the most important for flavor. A healthy and delicious diet is important for increased productivity and long-term health, but it can be difficult to accomplish.

1.2 Solution

We propose an AI-Nutritious Culinary Assistant that recognizes available ingredients and generates a personalized recipe with interactive, step-by-step guidance. Using the Meta Quest 3 as the user interface and sensor front-end, the system streams video and voice commands to an edge vision processor running an ingredient recognition pipeline. In addition to vision, the device integrates an environmental sensor module that measures ingredient weight for portion verification. Finally, the appliance includes a circular seasoning dispenser driven by stepper motors for proportional seasoning action, enabling closed-loop "dispense to target grams" assistance during cooking. The spice containers will also contain IR sensors, allowing users to see the percentage they have left and receive an alert when running low.

1.3 Visual Aid

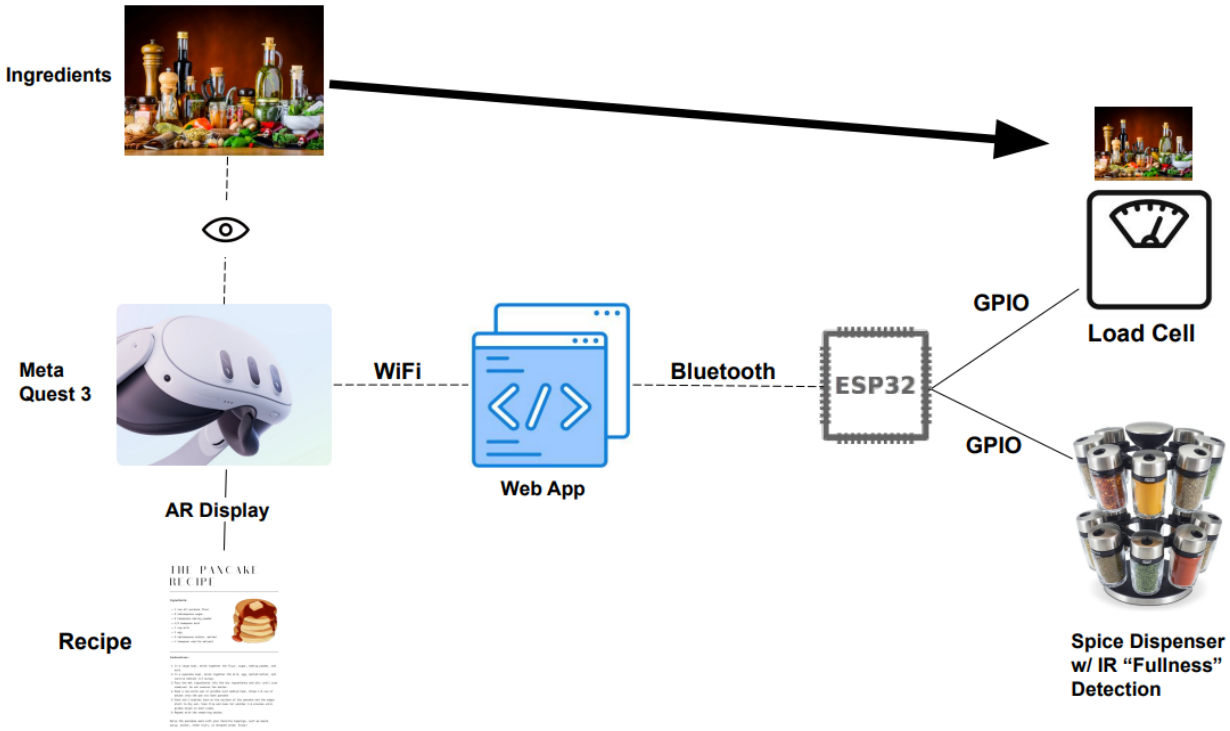


Figure 1: High Level Abstraction of The Project Pipeline

2 Design

The design is organized into the physical dispensing hardware, sensing hardware, power subsystem, communication layer, and local inference pipeline. The sections below describe the major design decisions, the implementation details, and the quantitative equations used to size or evaluate each subsystem.

2.1 Block Diagram

Figure 2 shows the top-level material, power, sensing, and communication flow for the NuChef culinary assistant.

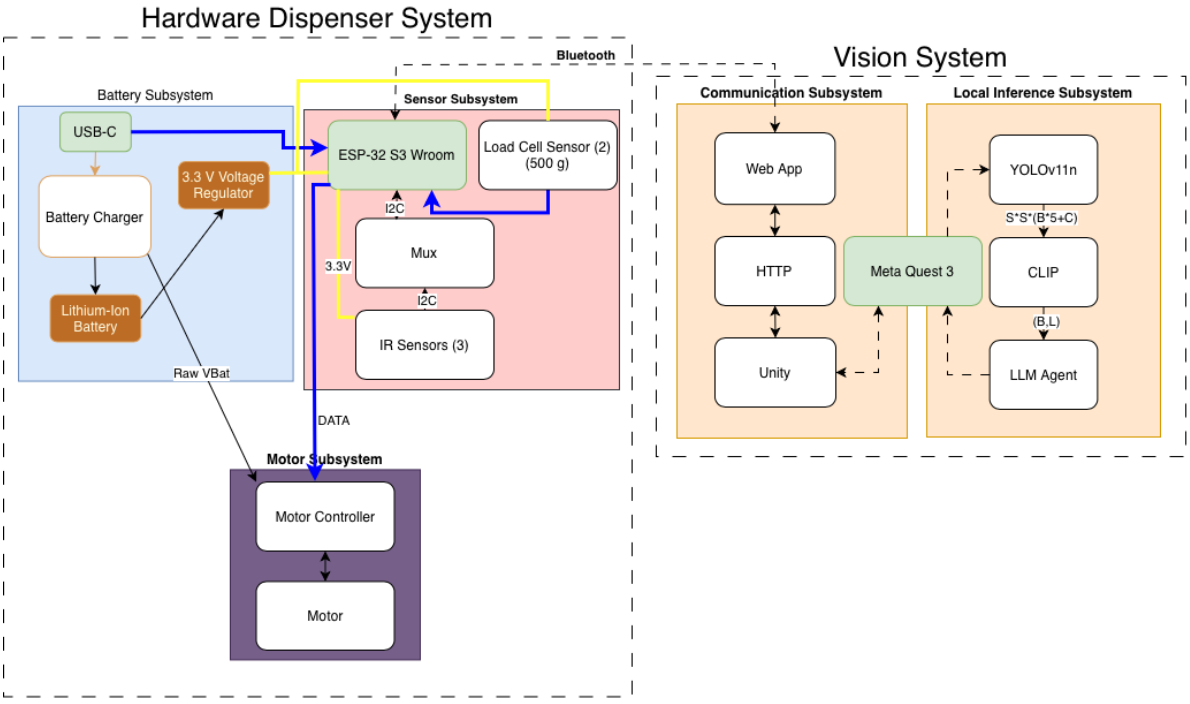


Figure 2: System block diagram for the NuChef culinary assistant.

2.2 Physical Design

The physical design, as shown in **??** combines a countertop spice dispenser with integrated sensing and control electronics. Three spice containers feed a central funnel through motor-driven rotating cylinders, and a load-cell platform verifies the mass of dispensed spices. The ESP32, motor drivers, battery charger, and voltage regulation circuitry are separated from the food path to simplify wiring and reduce contamination risk.

2.3 Battery Subsystem

2.3.1 Design Procedure

The battery subsystem powers the device from a rechargeable lithium-ion battery, supports USB-C charging, and generates stable rails for logic and actuation. It provides a regulated $3.3\,\mathrm{V}$ rail for the ESP32 and sensors while providing the raw lithium-ion battery voltage, approximately 3.0–$4.2\,\mathrm{V}$, to the stepper motors. This split supply prevents logic brownouts during motor current spikes.

2.3.2 Design Details

Linear-regulator selection. A low-dropout (LDO) linear regulator is acceptable for the logic rail because the expected load current is small. At full charge, the lithium-ion battery voltage is $4.2\,\mathrm{V}$. During testing, the ESP32-S3, Bluetooth Low Energy (BLE) radio, and sensors drew no more than approximately $200\,\mathrm{mA}$ during peak communication and sensing activity. Equation (1) estimates the maximum LDO power dissipation.

$$P_{\mathrm{LDO,max}} = (V_{\mathrm{BAT,max}} - V_{\mathrm{OUT}})\, I_{\mathrm{max}} = (4.2\,\mathrm{V} - 3.3\,\mathrm{V})\,(0.200\,\mathrm{A}) = 0.18\,\mathrm{W} \tag{1}$$

For comparison, Equation (2) estimates the conversion loss of a 90% efficient buck converter under the same load.

$$P_{\mathrm{buck,loss}} = \frac{V_{\mathrm{OUT}} I_{\mathrm{max}}}{\eta} - V_{\mathrm{OUT}} I_{\mathrm{max}} = \frac{(3.3\,\mathrm{V})\,(0.200\,\mathrm{A})}{0.90} - (3.3\,\mathrm{V})\,(0.200\,\mathrm{A}) = 0.073\,\mathrm{W} \tag{2}$$

Equations (1) and (2) show that the LDO wastes approximately $0.11\,\mathrm{W}$ more than the buck converter at the estimated peak load. This difference is small relative to the $10\,\mathrm{Wh}$ battery capacity, so the simpler LDO design is justified.

A second consideration is dropout voltage. The AP2112K has a typical dropout of $0.125\,\mathrm{V}$ at $300\,\mathrm{mA}$, so the logic rail remains regulated for battery voltages down to approximately $3.425\,\mathrm{V}$. This operating range covers more than 90% of the lithium-ion battery voltage range. A voltage divider senses battery charge percentage so that the microcontroller and sensors can enter a software-controlled sleep mode before the cells are overdischarged.

The charger is configured for a $500\,\mathrm{mA}$ charge current, which matches the maximum rated charging rate for the MCP73831 battery charger. The programming resistor sets the regulation current according to Equation (3), where I_{REG} is expressed in amperes and R_{PROG} is expressed in ohms.

$$I_{\mathrm{REG}} = \frac{1000}{R_{\mathrm{PROG}}}, \qquad R_{\mathrm{PROG}} = \frac{1000}{0.5} = 2000\,\Omega \tag{3}$$

Figure 3 shows the battery charger, battery input, voltage-divider, and voltage-regulator schematics used in the power subsystem.

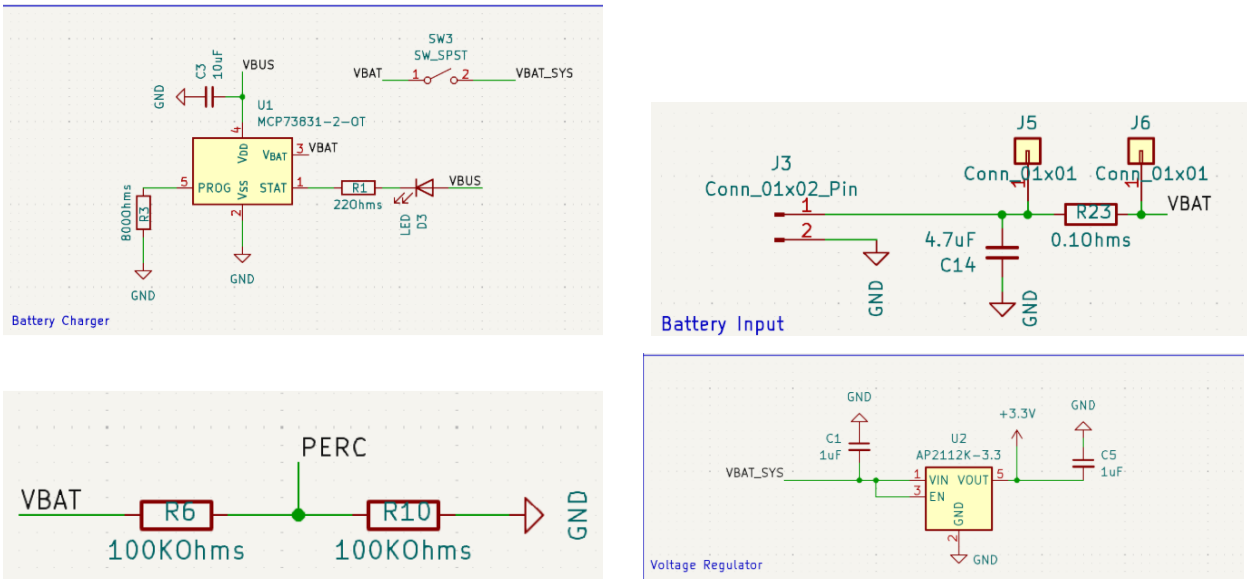


Figure 3: Battery charger, battery input, voltage-divider, and voltage-regulator schematics.

Cell choice. The cell was chosen for its $1.5\,\mathrm{A}$ peak output current and large capacity. These characteristics support continuous kitchen use without requiring the device to remain connected to an outlet.

Power connection. JST connectors prevent reverse battery insertion by allowing the battery to be connected in only one orientation. The rechargeable battery is intended to remain preassembled, so the user should rarely interact with the cell directly.

Interface. The battery subsystem supplies all power used by the sensor and motor subsystems. The regulated $3.3\,\mathrm{V}$ rail powers the microcontroller and sensors, and the raw battery voltage drives the motors.

2.4 Sensor Subsystem

2.4.1 Design Procedure

The sensor subsystem uses a $1\,\mathrm{kg}$ load cell to measure ingredient and spice mass for portion verification. It also uses infrared (IR) time-of-flight (ToF) sensors inside the spice containers to estimate remaining spice level. The ESP32 reads the sensors, applies tare and scale-factor calibration, and forwards measurements to the motor controller and web application.

2.4.2 Design Details

The HX711 load-cell analog-to-digital converter (ADC) was selected because it is well supported by existing libraries and is designed for bridge-style load-cell measurements.

Using Bogde's HX711 library, the ESP32 reads the load-cell voltage through general-purpose input/output (GPIO) pins. When an object is placed on the suspended half of the load cell, strain changes the bridge resistance, which creates the differential voltage shown in Figure 4. Equation (4) gives the calibrated conversion from load-cell voltage to mass.

$$m = K_{\text{cal}} \left(V_{\text{out}+} - V_{\text{out}-} \right) + m_{\text{tare}} \tag{4}$$

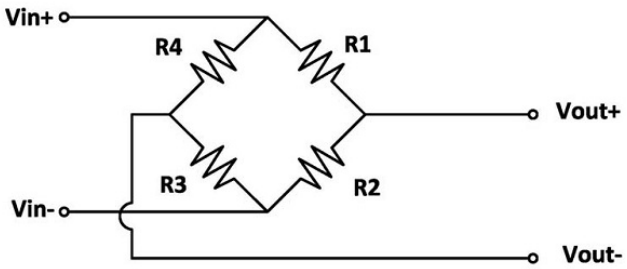


Figure 4: Load-cell bridge voltage generated by mechanical strain.

The ESP32-S3-WROOM-1 was selected for its available GPIO count, built-in antenna, crystal, and BLE capability. The ESP32-C3 Mini was considered initially, but it did not provide enough GPIO pins for the number of sensors used in the project.

The VL53L4CD ToF sensor was selected for its turnkey readiness and 1–$1200\,\text{mm}$ sensing range [9]. Its 18° field of view (FoV) required large, square spice containers to avoid wall interference. The sensor reports a distance measurement, and the controller converts that measurement into a spice-fill percentage using full-container and empty-container calibration distances, as shown in Equation (5).

$$F_{\text{spice}} = 100 \, \frac{d_{\text{empty}} - d_{\text{meas}}}{d_{\text{empty}} - d_{\text{full}}} \tag{5}$$

In Equation (5), F_{spice} is the estimated container fill percentage, d_{meas} is the measured distance, d_{full} is the full-container calibration distance, and d_{empty} is the empty-container calibration distance. Figure 5 shows the load-cell amplifier and IR sensor schematics.

2.5 Motor Subsystem

2.5.1 Design Procedure

The motor subsystem uses three stepper motors to dispense three spices into a central funnel and cup. Each motor turns a cylindrical gate that normally blocks the bottom opening of a spice container. A notch in the cylinder releases a repeatable volume of spice during each rotation. The ESP32 controls the motors and uses the load-cell measurement to stop dispensing when the target mass is reached.

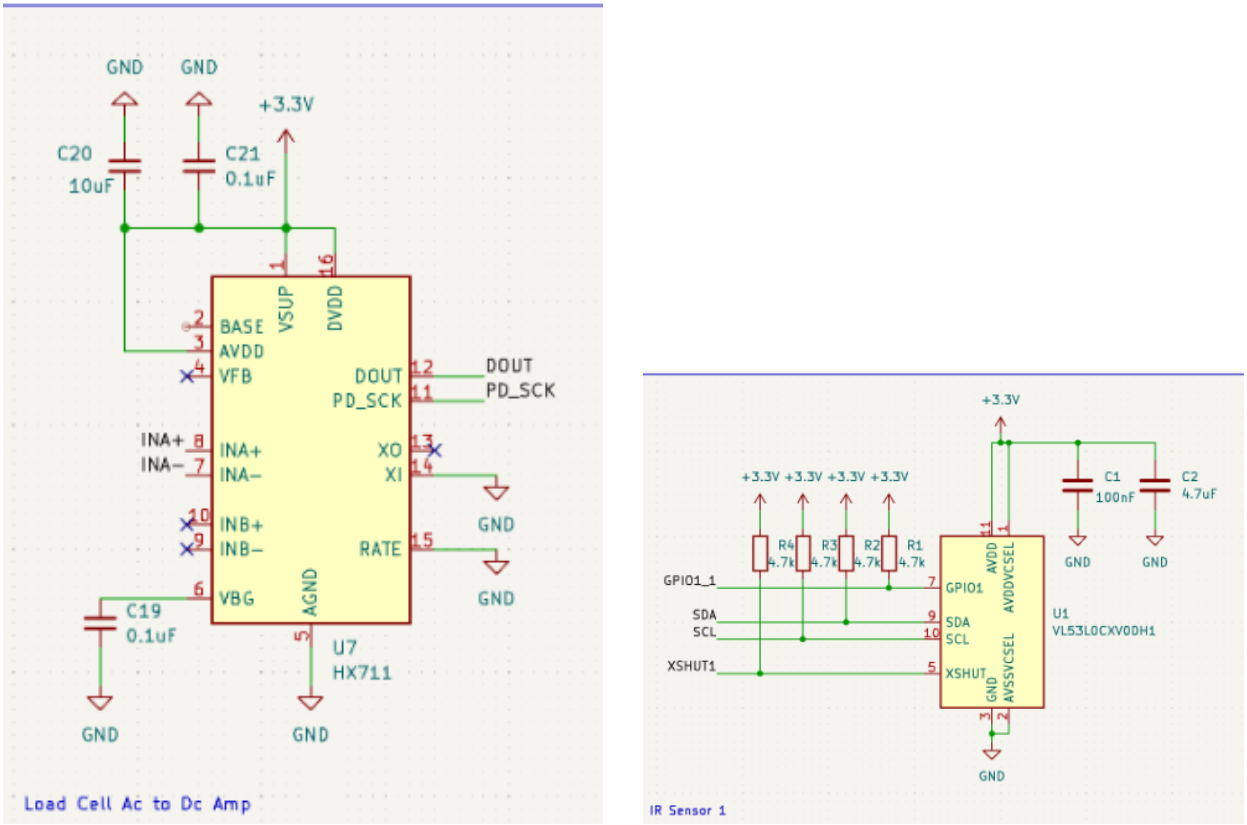


Figure 5: Load-cell amplifier and IR ToF sensor schematics.

2.5.2 Design Details

Stepper motors were selected because their motion is repeatable without an additional position sensor. Each motor has 200 full steps per revolution, so Equation (6) gives the nominal full-step angle.

$$\theta_{\text{step}} = \frac{360^\circ}{200} = 1.8^\circ \tag{6}$$

The driver energizes the two motor coils through the `A+`, `A−`, `B+`, and `B−` terminals described in the 17HE15-1504S datasheet [8]. The changing coil currents rotate the internal magnetic field and step the rotor. The design uses a TB6612FNG motor driver and the AccelStepper library so that the ESP32 can command each motor through GPIO pins.

2.6 Vision Communication Subsystem

2.6.1 Design Procedure

The communication subsystem is a lightweight local web application that acts as the real-time communication hub between the Meta Quest 3 interface, the ESP32 scale module, and the downstream vision processor and recipe planner. It combines user interaction

events, such as spice selection, tare commands, confirmations, and step navigation, with continuous weight telemetry from the ESP32. These data are stored in a timestamped event stream and a continuously updated cooking state. The consolidated state is forwarded to the vision processor and recipe planner so the system can enforce step logic, such as stopping at a target mass or requesting an additional $2\,\mathrm{g}$ of spice, while keeping the augmented-reality/virtual-reality (AR/VR) overlay synchronized with physical actions.

This subsystem is intentionally minimal and fast. It prioritizes low-latency forwarding, basic validation, and robust reconnection rather than heavy computation.

Design decisions. Three design decisions shaped the final user experience.

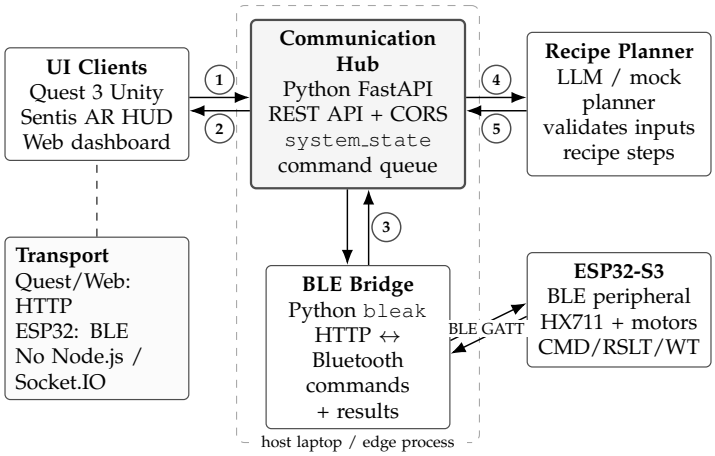
a. **Unity development environment.** Unity was selected as the Meta Quest 3 development environment instead of the Android Camera API because it provides a clearer path for annotations and future interaction improvements.

b. **Architecture ownership change.** The original architecture sent the headset video stream to a backend computer for pretrained computer-vision inference. The final architecture runs inference locally on the VR device, reducing communication latency. The tradeoff is an observed 6% accuracy reduction caused by selecting YOLOv11-FT, a compact model suitable for the limited GPU resources on the Meta Quest 3.

c. **State-machine design.** The backend serializes operation with a state machine. This makes the user flow more deterministic and creates explicit states for errors, recovery, and reset behavior.

2.6.2 Design Details

The communication framework connects three entities: the Meta Quest 3, the central web application, and the ESP32. The framework emphasizes low-latency forwarding, basic validation, and robust reconnection across those devices.



1 **UI Events:** select spice, next/previous step, confirm step, tare.
2 **UI Feedback:** target reached, add more/less, warnings, current step display.
3 **Telemetry:** weight in grams and sensor status; raw HX711 readings are converted into filtered readings with a stability flag.
4 **Forwarded Stream + Snapshots:** timestamped UI/hardware events plus the current cooking state.
5 **Planner Commands:** step updates, target grams, and UI messages to display in the headset/dashboard.

Note: The Communication Hub is currently represented as a Python FastAPI service, but the block is intentionally abstract so the web-service implementation can be changed later without changing the subsystem interfaces.

Figure 6: Updated communication subsystem. The Node.js/WebSocket layer is removed; the FastAPI backend maintains cooking state, the Quest and dashboard communicate over HTTP, and the ESP32 dispenser is reached through a host-side BLE bridge.

Figure 7 shows the backend state machine that controls the operation sequence. The states ensure that scanning, recipe generation, dispensing, user cooking steps, and completion occur in a deterministic order.

Figure 7: Backend state machine for scanning, recipe generation, dispensing, and completion.

- **IDLE**: The system waits for the user to begin operation.
- **SCANNING**: The Meta Quest 3 runs YOLO inference and records candidate ingredients.
- **INGREDIENTS CONFIRMED**: The user confirms the detected ingredients before recipe generation.
- **RECIPE READY**: The recipe is generated and formatted into ordered steps.
- **DISPENSING STEP**: The ESP32 controls the stepper motor and reads feedback from the load cell.

- **USER COOK STEP**: The system waits for the user to complete the current cooking instruction.
- **COMPLETE**: The cooking workflow has finished.

2.7 Local Inference Subsystem

2.7.1 Design Procedure

The local inference subsystem performs NuChef's core perception by converting Meta Quest 3 red-green-blue (RGB) frames into a structured, real-time ingredient representation for the recipe planner. Earlier multistage designs, such as "detect, crop, classify" pipelines, introduced extra latency and compounded errors. The final design uses a single YOLOv11 model to jointly perform detection and classification in one forward pass, following the original YOLO architecture's use of GoogLeNet **YOLO**. The model recognizes two top-level ingredient categories, protein and vegetable, and outputs a bounded list of detections containing class labels, confidence scores, and locations. The planner uses this structured ingredient list to generate step-by-step guidance in the headset overlay.

For each processed frame, the subsystem outputs the detection set in Equation (7).

$$\begin{aligned} &\mathcal{D} = \{(c_i, p_i, \mathbf{b}_i)\}_{i=1}^{N}, \\ &c_i \in \{\text{protein}, \text{vegetable}\}, \\ &p_i \in [0,1], \\ &\mathbf{b}_i = (x_i, y_i, w_i, h_i) \end{aligned} \tag{7}$$

In Equation (7), c_i is the ingredient class, p_i is the confidence score, and $\mathbf{b}_i$ is the bounding box. If a segmentation variant is used later, $\mathbf{b}_i$ can be replaced by a pixel mask $\mathbf{m}_i$.

Main formulas and technology stack. The inference pipeline relies on three quantitative tools: intersection over union, temporal smoothing, and transfer learning.

a. *Intersection over Union and greedy detection matching.*

Detection evaluation is based on Intersection over Union (IoU). Given two axis-aligned bounding boxes A and B, Equation (8) defines IoU.

$$\text{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \in [0,1] \tag{8}$$

The system converts raw detections into instance-level true positives (TP), false positives (FP), and false negatives (FN) using confidence-sorted greedy matching. Predictions are sorted by confidence. Each prediction p_i is matched to the unmatched ground-truth box g_j that maximizes $\text{IoU}(p_i, g_j)$. If $\text{IoU}(p_i, g_j) \geq \tau$ and the class labels agree, the prediction is marked as a TP and g_j is removed from the candidate pool.

Otherwise, the prediction is marked as an FP. Remaining unmatched ground-truth boxes are counted as FN.

Equation (9) defines the precision, recall, and F_1 metrics used after matching.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad F_1 = \frac{2PR}{P + R} \tag{9}$$

The detection-style accuracy proxy in Equation (10) penalizes both missed objects and false alarms.

$$\text{Acc} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{10}$$

b. *Temporal smoothing: EMA boxes and confidence-weighted voting.*

A lightweight IoU-based tracker associates detections across frames to reduce video jitter. Two smoothing mechanisms stabilize each track k.

Exponential Moving Average on bounding boxes. Let $\mathbf{b}_k^{(t)}$ denote the smoothed box for track k at frame t, and let $\hat{\mathbf{b}}_k^{(t)}$ denote the newly matched detection box. Equation (11) defines the exponential moving average (EMA) update.

$$\mathbf{b}_k^{(t)} = \alpha\, \mathbf{b}_k^{(t-1)} + (1 - \alpha)\, \hat{\mathbf{b}}_k^{(t)}, \qquad \alpha \in (0, 1) \tag{11}$$

A larger α, such as 0.7, favors the historical position and suppresses frame-to-frame noise.

Confidence-weighted class vote. Rather than trusting a single frame's class prediction, the tracker aggregates the last K observations. For each candidate class c, Equation (12) defines the vote score.

$$S(c) = \sum_{i \in \mathcal{H}_K} w_i\, \mathbf{1}[\hat{c}_i = c]\,, \qquad \hat{c}_k^{(t)} = \arg\max_c\, S(c) \tag{12}$$

In Equation (12), $\mathcal{H}_K$ is the set of the last K observations and w_i is the detector confidence at frame i. This vote down-weights low-confidence detections and improves temporal consistency.

c. *Transfer learning with fine-tuned backbones.*

Both model families in the bake-off use transfer learning. The YOLOv11n detector is initialized from COCO-pretrained weights and fine-tuned end-to-end on the ingredient dataset by minimizing the multitask loss in Equation (13).

$$\mathcal{L} = \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} \tag{13}$$

In Equation (13), $\mathcal{L}_{\text{box}}$ is the CIoU regression loss, $\mathcal{L}_{\text{obj}}$ is the binary cross-entropy objectness loss, and $\mathcal{L}_{\text{cls}}$ is the cross-entropy classification loss.

The ResNet-18 crop classifier is loaded with ImageNet-pretrained weights. Its final fully connected layer is replaced with the task-specific layer in Equation (14).

$$f_\theta(\mathbf{x}) = W_{\text{new}}\,\phi(\mathbf{x}) + \mathbf{b} \tag{14}$$

In Equation (14), $\phi(\mathbf{x})$ is the backbone embedding, W_{new} has size $C \times 512$, and C is the number of ingredient classes. Training uses AdamW with cross-entropy loss.

2.7.2 Design Details

Several additional techniques are used throughout the local inference notebook. Class-frequency analysis counts ground-truth annotations per class, selects the most frequent K ingredients, and maps the remaining labels to an "unknown" class to keep the task tractable. Data augmentation, including random resized crop, horizontal flip, and color jitter, is applied during convolutional neural network (CNN) training to improve generalization. Macro-averaged precision, recall, and F_1 are used for the crop classifier so that rare classes contribute equally to the aggregate score. Finally, the detect-every-N-frames strategy skips expensive inference on intermediate video frames and relies on the IoU tracker to interpolate between detections, trading a controlled amount of accuracy for lower effective inference time.

3 Verification

3.1 Battery Subsystem Verification Result

We carried out the verification experiments as stated in Table 1. Our low-level requirements were validated using a multimeter on printed testpoints on our PCB, validating our current and voltage requirements for our entire system. We were able to verify our battery charge by giving it time plugged in and recording the battery voltage over time. We can see in Figure 8 below that our battery was gaining voltage over time.

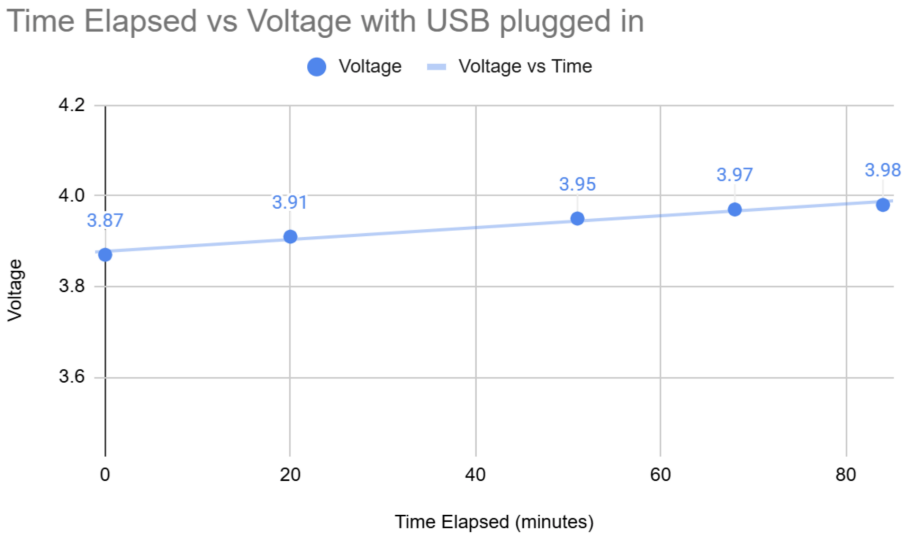


Figure 8: Battery Voltage vs Time

We can see that the battery gained about 0.1V over the course of one hour. While this is slow, we can attribute this to the fact that we planned our system for a smaller battery, but supply and timing issues resulted in our use of a larger capacity battery. While this did not affect the overall design, especially as we pass the voltage through a regulator before sending it through the system, the larger battery does result in slower charging. Nevertheless, we are happy with the result that our battery is able to be charged.

3.2 Sensor Subsystem Verification Subsystem

We carried out the verification experiments as stated in Table 2. For the load cell, as specified in the table, we were looking for spice accuracy to be within one gram, and ingredient accuracy to be within 2-3 grams, as the spices are much more precise measurements and the focus of our project. We ran a number of trials to analyze the Load Cell accuracy, as seen in Figures 9 and 10. For each trial, we tared the load cell and placed the item on it, recording the weight that is displayed on our web app after scaling on the backend. We the compare this weight to a commercial kitchen scale reading of the same ingredient.

Item	Load Cell (g)	Actual (g)	Difference (g)	%
Carrot	177.24	177	−0.24	−0.14%
Phone	213.32	213	−0.32	−0.15%
Zucchini	225.96	225	−0.96	−0.43%
Measurer	157.91	158	+0.09	+0.06%
Solder Spool	446.87	448	+1.13	+0.25%
Avg Err		**0.548 g**	**0.20%**	

Figure 9: Load Cell Ingredients: Measured Weight vs Actual Weight

Item (MSG)	Measured (g)	Actual (g)	Diff (g)	%
Trial 1	18.78	19	+0.22	+1.16%
Trial 2	34.58	34	-0.58	-1.71%
Trial 3	58.15	58	-0.15	-0.26%
Trial 4	7.18	7	-0.18	-2.57%
Trial 5	3.54	4	+0.46	+11.50%
Avg Error		**0.318 g**	**3.44%**	

Figure 10: Load Cell Spices: Measured Weight vs Actual Weight

From these results, we can see that our load cell remained within our desired accuracy for every test. Our ingredients had an average error of 0.548 grams, with the largest discrepancy being a difference of 1.13 from expected. Our spices had an average error of 0.318 grams, with the largest discrepancy being only 0.58 grams.

Some of this error can be attributed to noise, as smaller-grade load cells require more precise measurements and therefore result in more impactful noise. One way we found to counteract this was to take many samples and average them to get the result. This resulted in a greater delay of about two-three seconds between load cell readings but a lower overall error.

A large portion of the spice error can be attributed simply to measurement precision in our verification scale, which could only measure to the nearest gram. With a more precise scale we would anticipate the average error decreasing even more.

For our IR sensors, we were looking for a display accuracy within 1% of the spice container size. We took measurements at multiple different levels of container fullness. We then compared this to the spice weight fullness percentage for each reading. This was measured by taking the weight of the spice in the container and calculating $Weight_{\%} = Weight_{measured}/Weight_{Maximum}$. Our results are shown in Figure 11.

Weight (g)	IR Distance	IR %	Weight %	% Difference
0	153	0%	0.00%	0.00%
44	132.5	19%	19.01%	+0.37%
89.48	113	38%	38.66%	-0.85%
124.48	91.9	58%	53.78%	+3.97%
164.48	73.2	75%	71.06%	+4.37%
231.48	47.2	100%	100.00%	0.00%
Average Difference				**1.59%**

Figure 11: IR Sensor Distance vs Weight Verification

We can see from our results that our IR sensor never strayed more than 4.5% from the expected spice level, resulting in an average differece of 1.59%. This was a bit higher than we desired, which can be attributed to IR beam interference with the walls of the container or non-uniform peaks of spice within the container. We originally had a very large error in our testing that we were able to greatly reduce by increasing the size of our spice containers. We increased the size from 2″x0.5″ to 2″x2″ to account for the 18^o IR FoV. Overall, the IR sensors are used for a general idea of the spice level as opposed to an accurate measurement, so we are happy with our results.

3.3 Motor Subsystem Verification Result

We carried out the verification experiments as stated in Table 3. As seen in Figure 12 below, our motor remained overall consistant in its angle when completing full rotations. Stepper motors are designed to be precise, and in our case, 200 steps corresponds to one rotation.

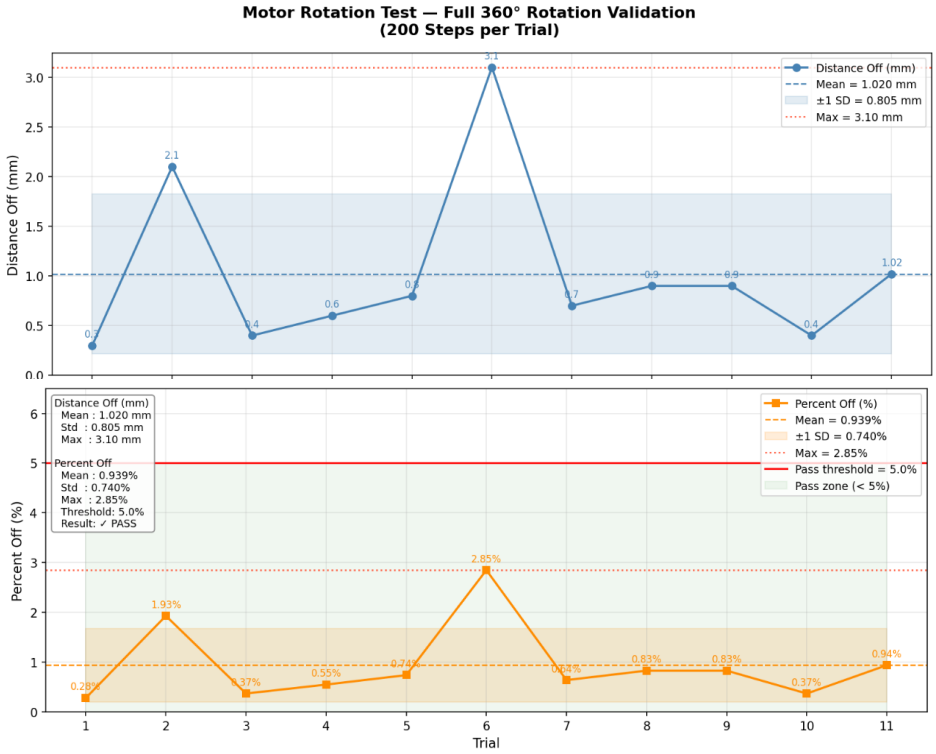


Figure 12: Motor Rotation Verification

Outliers in this data can be overall attributed to friction. It is possible that our motor and attached cylinder cup have friction with the wooden sides of the spice holder as well as the clear plastic covering. Any spice leakage can also cause friction when wedged between the spinning component and the wall of the container. Regardless, the motors on average were able to stay within 2% of the specified rotation. In order to account for small inaccuracies, a manual mode is implemented on the frontend of the web app. This allows the user to make small adjustments, such as 3-5 steps of the motor in either direction, to ensure that it remains accurate and avoids spillage.

3.4 Vision Communication Subsystem

This subsystem highlights the AI and VR interaction components are indeed functional and fast. Table 4 showed that Meta Quest 3 fits seamlessly into our project ecosystem by providing:

a. Consistency

b. Low latency

The communication subsystem was tested by sending timestamped events between the Quest interface, web application, ESP32, and planner. As supported in the evaluation result in Table 6, during burst testing, the system forwarded events at the required rate of 10 events/sec without dropping unique event IDs. The measured 95th-percentile event la-

tency was below the 150 ms requirement, so the communication delay was not noticeable during normal user interaction.

3.5 Local Inference Subsystem

Table 5 detailed the important requirements and experiments that we carried out to ensure our AI is robust for the users' commercial use. As usual, to be concise yet informative here, we emphasized:

a. High accuracy

b. Low latency

c. Intuitive Usage

The local inference subsystem was evaluated on a held-out validation set. The YOLO-based model produced bounded ingredient detections with class labels, bounding boxes, and confidence scores. Model performance was summarized using mean average precision, top-1 classification accuracy, and a confusion matrix in Figure 13. The most common errors occurred between visually similar food categories, especially under low-light or partially occluded conditions. However, the inference results were sufficient for the project goal of generating ingredient-aware cooking guidance.

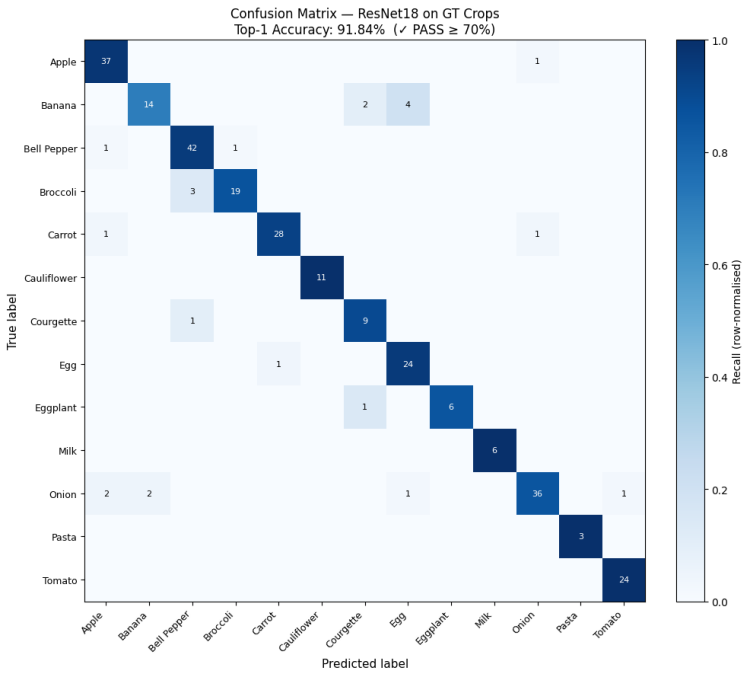


Figure 13: Confusion Matrix

4 Cost

The Meta Quest 3 used in the prototype was borrowed from the library, so it is excluded from the prototype parts total. However, a marketable version of NuChef would require the headset to be borrowed, rented, or purchased separately. As shown in Table 9 in Appendix **??**, the total prototype parts cost, excluding the Meta Quest 3, is $127.20. Applying estimated sales tax of 10% and estimated shipping of 5% adds $12.72 and $6.36, respectively, bringing the adjusted parts cost to $146.28.

Labor cost was estimated using the ECE 445 final report cost model, which defines each partner's labor cost as the ideal hourly salary multiplied by the actual hours spent and by a factor of 2.5 [11]. With an ideal salary of $\$40/\mathrm{h}$ and 60 hours per team member, the labor cost is $\$40/\mathrm{h} \times 60\,\mathrm{h} \times 2.5 = \$6{,}000$ per team member. For three team members, the total labor cost is $18,000. The machine shop cost is estimated as $\$70/\mathrm{h} \times 15\,\mathrm{h} = \$1{,}050$. Therefore, the total estimated project cost is $\$146.28 + \$18{,}000 + \$1{,}050 = \$19{,}196.28$.

5 Conclusion

There are many ethical considerations over the pipeline of our project. Working with a 3.7V Lithium battery with power scaling modules in the design, we must consider electrical safety as specified in the Safe Practice for Lead Acid and Lithium Batteries document [12]. We must store the battery when not in use in a safe place, ensuring that the pins are isolated and can not cause a short and possibly a fire. Our design needs to have ventilation to avoid any gas buildup. Additionally, we need to ensure that any errors in our design will not cause a short with our battery. We will have failsafes in place so that a short is not possible in our design. Because we are using a specifically Lithium battery, we will ensure that the voltage does not decay below 3.0V/cell or exceed 4.2 V/cell. Checking for swollen batteries is also an easy way to see if something is wrong. If the battery begins to swell or make noises, we will disconnect the battery and isolate it immediately.

As we are working with food, one major consideration is whether the materials and processes are food-safe. This is especially important in a commercial context, as we can not sell something made for food that will cause harm to users. Because of this, we need to consider FDA compliance to make our product safe [13]. The first consideration is the material. The containers must be in a material that has the NSF 51 Food Equipment Materials certification, for example, silicone. The material must also be able to withstand frequent exposure to cleaning compounds and sanitizing agents. Additionally, the material must withstand a range of temperatures without releasing anything dangerous into the food. These temperatures can vary, but for our purposes we want a rating above 100 °C to allow safe dishwashing. All of these considerations impact FDA compliance, which is an important part of getting our product safe and marketable.

When considering the ACM code of ethics, 1.2 is the most vague but also very important: "Avoid Harm [14]". This also ensures to follow section I.1 of the IEEE Code of Ethics, which says "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices...[15]." Since cooking involves a lot of sharp and hot objects, we must avoid hazards and keep the user safe. One important way is to make sure the recipe does not block the vision of the person cooking. Having impaired vision can lead to accidents and injuries in the kitchen. We also suggest that the person remove the headset while they are cooking on an open flame. This ensures the chef's vision and hand movements are not impaired while working with dangerous materials. These are considerations that we have to make to make sure users can avoid harm from our product.

Our project intends to have a beneficial impact on society as it helps reduce food waste and educate people on new recipes and ways to cook. By generating a recipe specific to the ingredients one has, they may also save money on ingredients while being resourceful with what they have. The generation of recipes can also expose people to new foods and cultures, expanding the project beyond country boundaries and joining people through food.

References

[1] Meta Platforms, Inc. "Meta Quest Developer Documentation," Meta Horizon OS Developers, Accessed: Feb. 13, 2026. [Online]. Available: https://developers.meta.com/horizon/.

[2] Meta Platforms, Inc. "Unity Passthrough Camera API Samples," GitHub, Accessed: Feb. 13, 2026. [Online]. Available: https://github.com/oculus-samples/Unity-PassthroughCameraApiSamples.

[3] Unity Technologies. "Unity Sentis: AI Inference Engine," Accessed: Feb. 13, 2026. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.sentis.

[4] AI Recipes Recommendations. "AI Recipes Recommendation Dataset," Roboflow Universe, Accessed: Feb. 13, 2026. [Online]. Available: https://universe.roboflow.com/ai-recipes-recommendations/ai-recipes-recommendation.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv preprint arXiv:1506.02640*, 2015. [Online]. Available: https://arxiv.org/abs/1506.02640.

[6] Avia Semiconductor, *HX711: 24-Bit ADC for Weigh Scales Datasheet*. Accessed: Feb. 13, 2026. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf.

[7] OpenAI. "GPT Models API Documentation," Accessed: Feb. 13, 2026. [Online]. Available: https://platform.openai.com/docs.

[8] StepperOnline, *17HE15-1504S Full Datasheet*. Accessed: Mar. 11, 2026. [Online]. Available: %7Bhttps://www.artme-3d.de/wp-content/uploads/2024/03/17HE15-1504S.pdf%7D.

[9] STMicroelectronics, *VL53L4CD: Time-of-Flight high accuracy low power proximity sensor Datasheet*. Accessed: Mar. 11, 2026. [Online]. Available: %7Bhttps://www.st.com/resource/en/datasheet/vl53l4cd.pdf%7D.

[10] Espressif Systems, *ESP32 Technical Reference Manual*. Accessed: Feb. 13, 2026. [Online]. Available: https://www.espressif.com/.

[11] ECE 445 Staff and ECE Editorial Services, *Preparing Your Final Report for ECE 445, Senior Design*, Final report preparation guidelines, Department of Electrical and Computer Engineering, University of Illinois Urbana-Champaign, Sep. 2019.

[12] Spring 2016 Course Staff ECE 445: Senior Design Project Laboratory. "Safe Practice for Lead Acid and Lithium Batteries," Accessed: Mar. 11, 2026. [Online]. Available: %7Bhttps://courses.grainger.illinois.edu/ece445/documents/GeneralBatterySafety.pdf%7D.

[13] U.S. Food and Drug Administration. "Guidance for Industry: Preparation of Premarket Submissions for Food Contact Substances (Chemistry Recommendations)," Accessed: Feb. 13, 2026. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/guidance-industry-preparation-premarket-submissions-food-contact-substances-chemistry.

[14] Association for Computing Machinery. "ACM Code of Ethics and Professional Conduct," Accessed: Feb. 13, 2026. [Online]. Available: https://www.acm.org/code-of-ethics.

[15] IEEE. “IEEE Code of Ethics,” Accessed: Feb. 13, 2026. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

Appendix A Requirements & Verifications

A.1 Requirements and Verification Tables

This appendix lists the subsystem-level requirements and validation procedures used to verify the final design. The verification results are discussed in the main Verification section, with these tables providing the detailed requirement-by-requirement reference.

Table 1: Battery subsystem requirements and validation.

Requirement	**Validation**
The battery charger must be able to recharge the battery through USB-C.	Using a voltage divider connected to a GPIO pin, we can obtain the real-time voltage of the battery. This voltage can be graphed over time to show that the battery is charging despite the power draw of the circuit.
The voltage regulator must be able to supply $3.3\,\mathrm{V} \pm 5\%$ to all other components consistently.	Using a multimeter and test pads for the ground plane and $3.3\,\mathrm{V}$ rail, we can measure and graph the voltage under load across a variety of operating conditions.
The battery must be able to supply enough current to drive the entire system. We anticipate the total system requiring a maximum discharge current of $1.5\,\mathrm{A}$, mostly to drive the stepper motors.	Using the $0.1\,\Omega$ resistor placed across the battery input, we can measure current using Ohm's law. From this measurement, we can determine the peak current supplied by the battery.

Table 2: Sensor subsystem requirements and validation.

Requirement	**Validation**
The load sensors must be able to weigh spices within $\pm 1\,\mathrm{g}$ and ingredients within ± 2–$3\,\mathrm{g}$.	A measurement using the load cell can be taken and then validated against a scale with a known tolerance.
The IR sensors must accurately measure and signal the distance to the spice level, with an accuracy of 1% of the spice container size.	An IR measurement can be taken for both a full and partially filled container. The measured fullness percentage can then be calculated by comparing the measured distance against the calibrated full and empty container distances.
The ESP32 must communicate over GPIO with the sensors and over Bluetooth with the web application to allow for proper readings and control.	Data can be read from the GPIO-connected sensors and then sent to the web application through Bluetooth. This validates whether the sensor information is transmitted correctly.

Table 3: Motor subsystem requirements and validation.

Requirement	**Validation**
The motors must be controlled by the ESP32 using GPIO and allow for precise rotations within $\pm 2^\circ$.	The motors can be commanded to step 200 times, representing a full 360° rotation. The difference between the starting and final positions can then be measured to ensure negligible drift.
The amount of spice dispensed during each rotation should not vary by more than 5% from the mean.	Fifty samples can be taken, and the mean, standard deviation, and maximum distance from the mean can be calculated.

Table 4: Vision communication subsystem requirements and validation.

Requirement	Validation
The system must receive Quest spice-selection and step events over Wi-Fi and forward them to the Vision Processor with end-to-end latency $\leq 150\,\mathrm{ms}$ and capacity ≥ 10 events/s.	Timestamps can be instrumented at Quest-send and planner-receive. A scripted burst of 10 events/s can be generated for 60 s. The system passes if the 95th percentile latency is $\leq 150\,\mathrm{ms}$ and no dropped events are detected by checking unique event IDs.
The system must maintain simultaneous connections to the Quest and ESP32 for at least 30 minutes and automatically recover from disconnects with a reconnect time $\leq 5\,\mathrm{s}$.	A 30-minute soak test can be run with the ESP32 streaming weight at 10–20 Hz and the Quest sending periodic events. Disconnects can be forced by disabling ESP32 Wi-Fi and closing the Quest browser. The system passes if reconnection occurs within 5 s and the state snapshot correctly resynchronizes the user interface and planner.

Table 5: Local inference subsystem requirements and validation.

Requirement	Validation
For each processed frame, the system must output ≤ 20 ingredient regions with (x, y, w, h) boxes or pixel masks and include confidence scores. The model must achieve ≥ 0.425 mAP.	Inference can be run on a representative test set to confirm that postprocessing enforces a maximum of 20 detections. The trained YOLOv11 model can then be evaluated on the validation dataset, and mAP at 0.5–0.95 can be reported.
For every region, the system must output a top-1 ingredient label and confidence, with top-1 accuracy $\geq 70\%$ on a held-out validation set. The system must then publish the structured ingredient list to the LLM/planner.	On a held-out validation set with at least 10 supported ingredient classes and no train/validation leakage, the system must achieve $\geq 70\%$ top-1 region classification accuracy, macro-F1 ≥ 0.80, and produce a confusion matrix with per-class precision and recall.

A.2 Verification Results Tables

The completed system was verified using block-level tests for communication latency, load-cell accuracy, and IR distance sensing. Table 6 shows that the communication subsystem met the latency and reliability requirements. Table 7 shows that the load cell stayed within the required ingredient and spice measurement tolerances. Table 8 shows that the IR distance sensor produced spice-fill estimates with an average percent differ-

ence of 1.59%.

Table 6: Communication latency validation test results.

Metric	Measured	Threshold	Result
Events sent	300	–	–
Events received OK	300	300, drop = 0	PASS
Dropped/error events	0	0	PASS
p_{50} latency	2.5 ms	–	–
p_{95} latency	3.4 ms	≤ 150 ms	PASS
Maximum latency	4.1 ms	–	–
Overall	–	–	**PASS**

Table 7: Load cell accuracy validation test results.

Item	Load Cell (g)	Actual (g)	Difference (g)	% Difference
Carrot	177.24	177	-0.24	-0.14%
Phone	213.32	213	-0.32	-0.15%
Zucchini	225.96	225	-0.96	-0.43%
Measurer	157.91	158	$+0.09$	$+0.06\%$
Solder spool	446.87	448	$+1.13$	$+0.25\%$
Ingredient average	–	–	**0.548 g**	**0.20%**
MSG	18.78	19	$+0.22$	$+1.16\%$
MSG	34.58	34	-0.58	-1.71%
MSG	58.15	58	-0.15	-0.26%
MSG	7.18	7	-0.18	-2.57%
MSG	3.54	4	$+0.46$	$+11.50\%$
Spice average	–	–	**0.318 g**	**3.44%**

Table 8: IR distance sensor versus weight validation test results.

Weight (g)	**IR Distance**	**IR %**	**Weight %**	**% Difference**
0	153	0%	0.00%	0.00%
44	132.5	19%	19.01%	$+0.37\%$
89.48	113	38%	38.66%	-0.85%
124.48	91.9	58%	53.78%	$+3.97\%$
164.48	73.2	75%	71.06%	$+4.37\%$
231.48	47.2	100%	100.00%	0.00%
Average	–	–	–	**1.59%**

Appendix B Design

An appendix can go here! Make sure you use the `\label{appendix:a}` above so that you can reference this section in your document.

B.1 Hardware

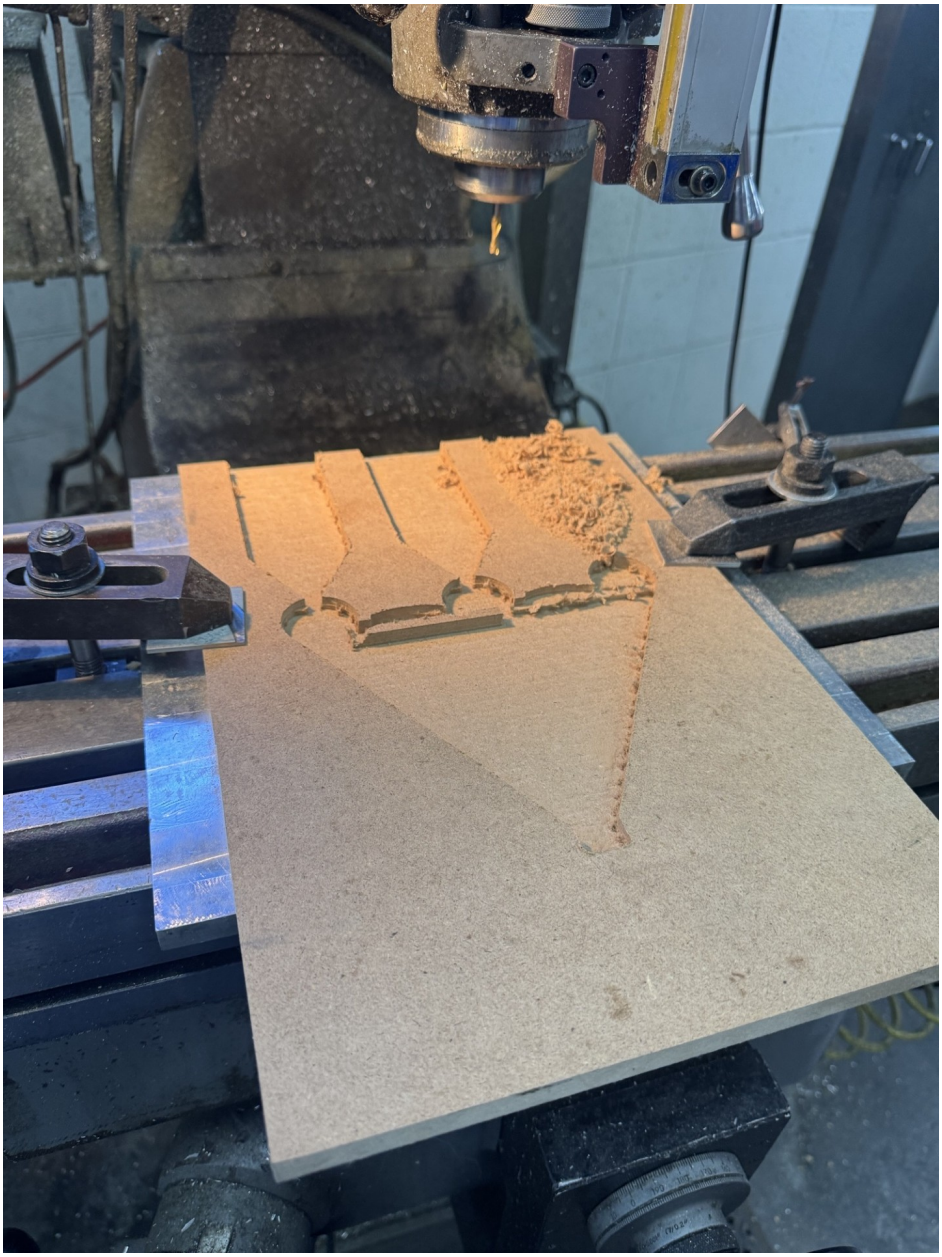


Figure 14: Main Wooden Design

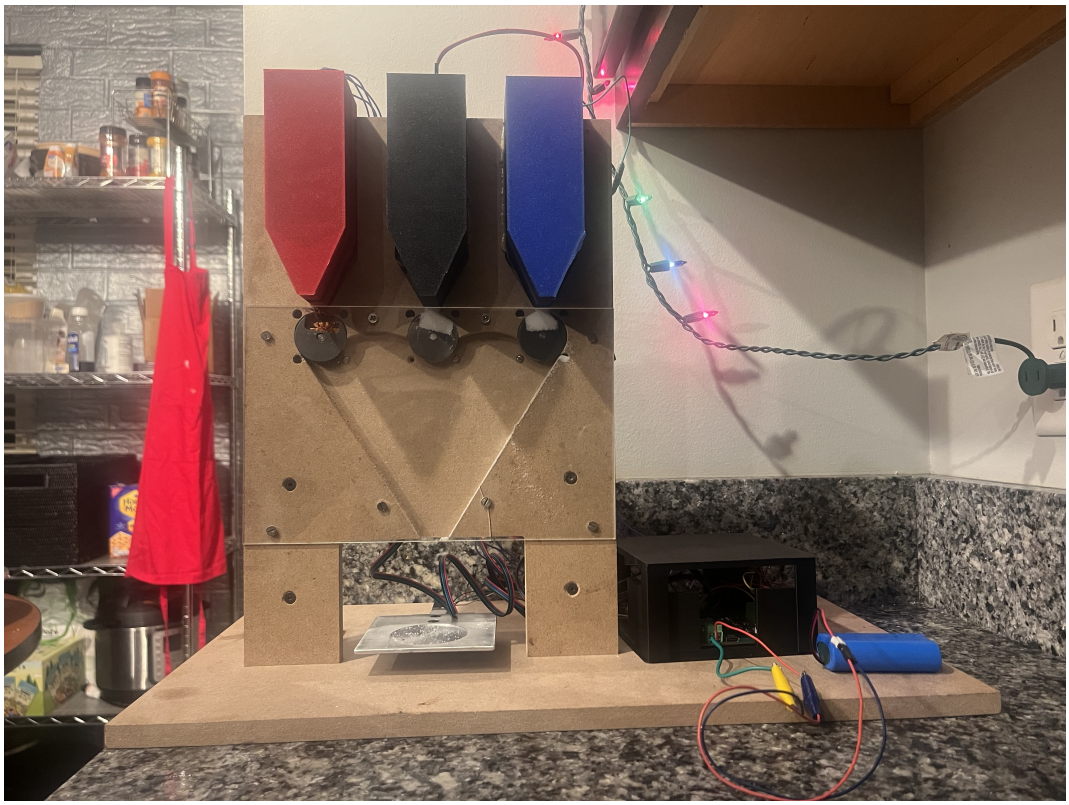


Figure 15: Entire Design

B.1.1 Electric PCB Hardware

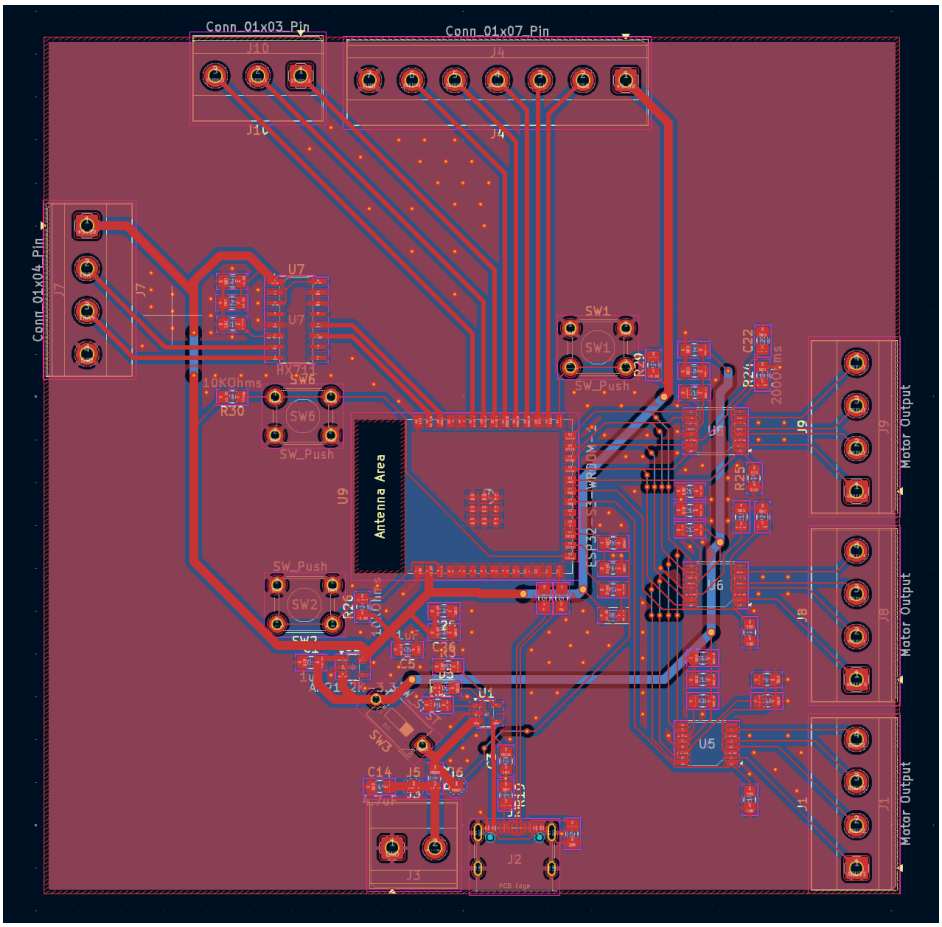


Figure 16: General PCB Layout

B.1.2 PCB Schematics Hardware

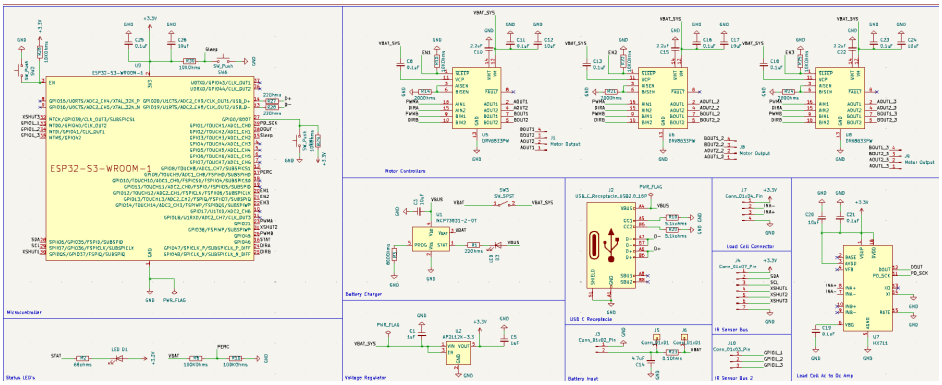


Figure 17: Entire Schmatic Design

B.2 Software

B.2.1 YOLO Training Framework

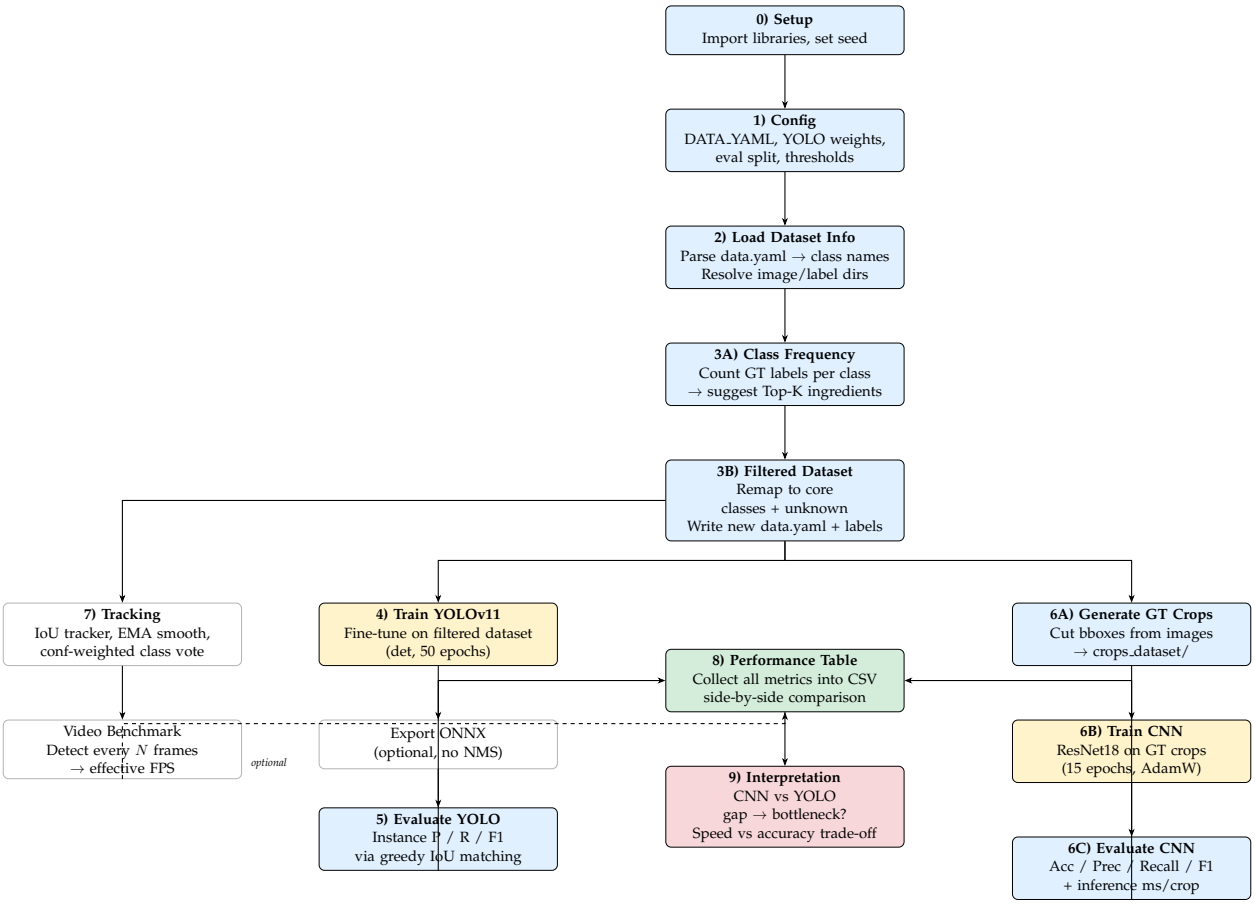


Figure 18: Pipeline for Vision Processor

Appendix C Cost

Table 9: Project Parts Cost Analysis

Description	Manufacturer	Quantity	Extended Price
Battery Manager: MCP73871	Microchip	1	$1.73
3.3V LDO Voltage Regulator: AP2112K-3.3	Diodes Incorporated	1	$0.22
USB-C Port: Generic 16-pin part	Hirose Connector	1	$1.45
2500 mAh lithium-ion battery from Adafruit	Adafruit	1	$14.95
Load Cell Amplifier: HX711	SparkFun	1	$11.50
ESP32 S3 WROOM 1	Espressif Systems	1	$5.49
VL53L4CD Time of Flight Distance Sensor (1–1300 mm)	Adafruit	3	$44.85
Motor Controller: DRV8833	Pololu	3	$32.85
Stepper Motors: 17HE15-1504	StepperOnline	3	$32.97
Total	–	–	**$145.01**