ECE 445

Spring 2026
Senior Design Document

NuChef AI Culinary Assistant

Team 62
Tony Liu, Griffin Kelley, Jackson Brown

TA: Aniket Chatterjee

1. Introduction

1.1 Problem:

The processed food industry has become increasingly toxic due to chemical flavor additives (12%-32% higher cancer risk), yet cooking often intimidates new chefs when preparing. Therefore, students and working professionals may rely on convenient but unwholesome meals. Most 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 reduces the likelihood that the user will learn. It is also inefficient to design recipes that can be adjusted to the user's available ingredients. 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 is 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:

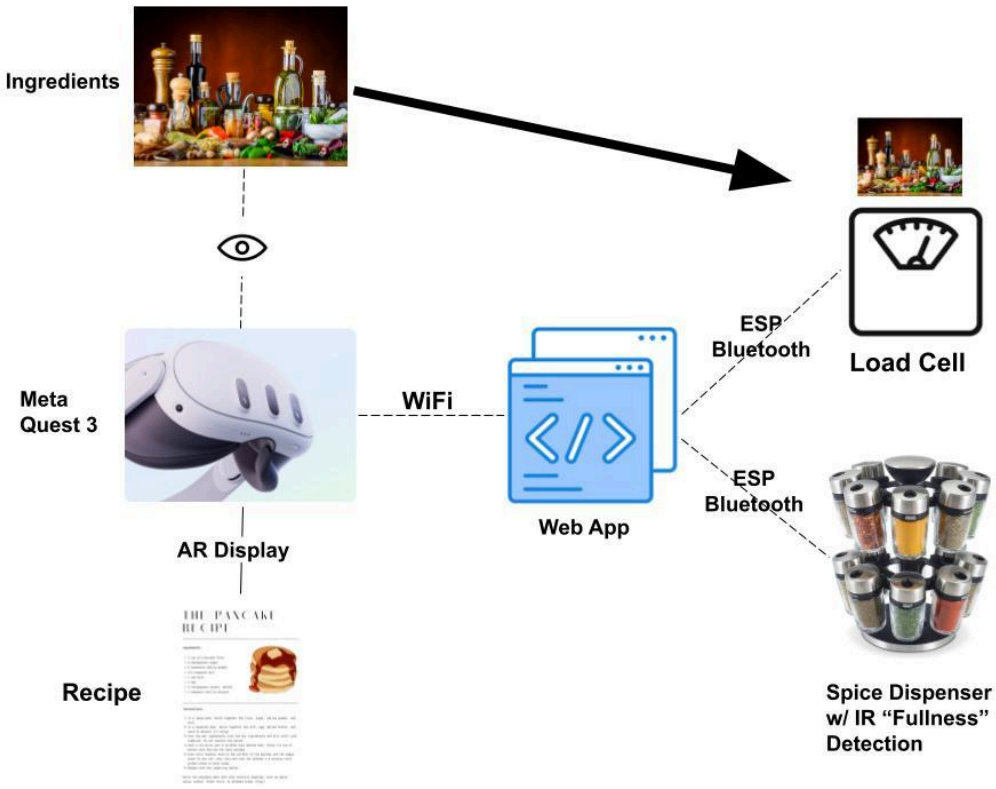


Figure 1: Visual aid demonstrating the recipe creation and spice dispensing process

1.4 High-Level Requirements List:

1. This project will use AI and computer vision to select a recipe based on the ingredients that the user has available. The CV + LLM model will be accurate in recognizing ingredients and will generate sensible recipes. The precise benchmarking for this will need to be adjusted based on feasibility.
2. The spice dispenser must be rechargeable and dispense the correct amount of spice with a maximum error margin of 1 gram.
3. The Meta Quest 3 and the Spice Controller will be able to communicate through a shared web app, exhibiting connection speeds of sub 10 seconds and a range of 100 feet.

2. Design

2.1 Physical Design

Our physical seasoning dispenser will be designed and assembled by the machine shop after providing three iterations of sketches. We provided every essential electric component: 3 stepper motors, 1 load cell sensor, and 3 IR sensors. The figure below shows the basic layout of how the spices will be converged in our physical device.

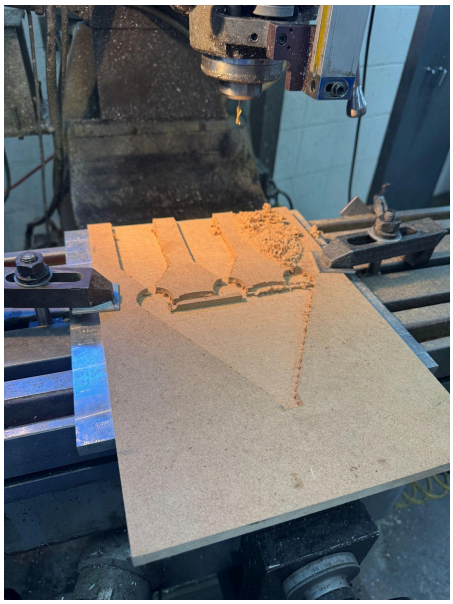


Figure 2: Physical Device in Progress

The three circular holes will be connected to a shaft, controlled by the stepper motor, to regulate the output rate of spice. By analyzing alongside the load cell sensor data from the bottom of the physical device, we are able to precisely dispense the required amount.

2.1 Block Diagram

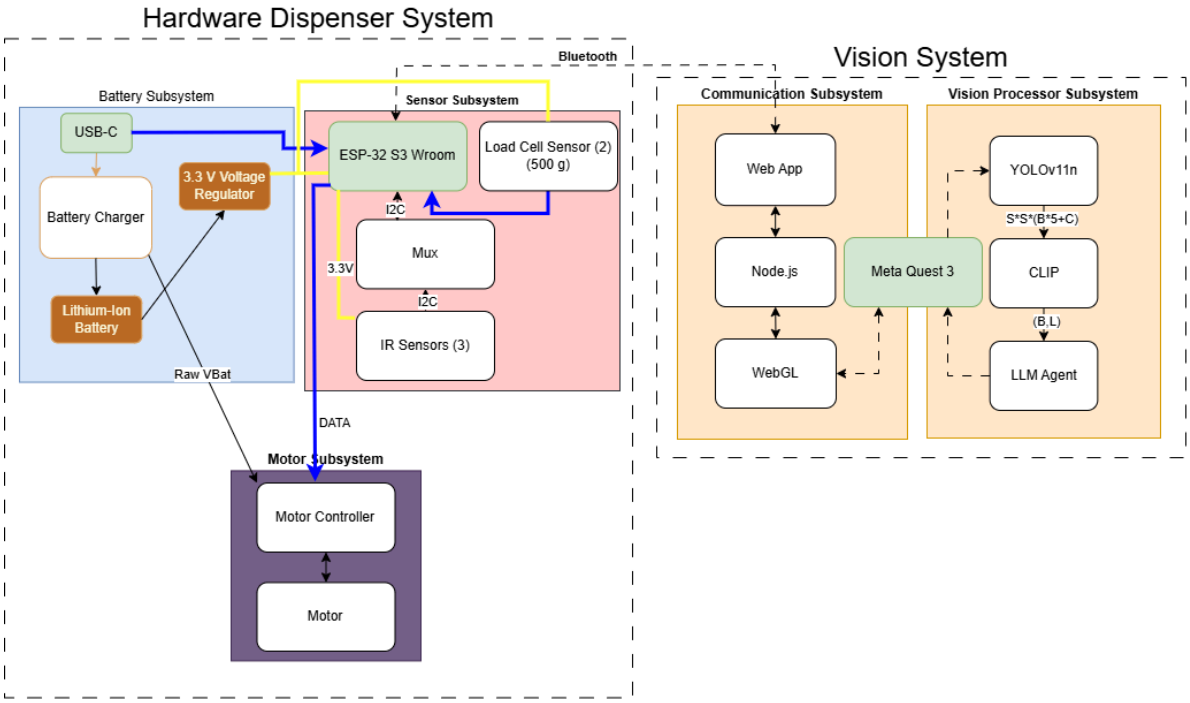


Figure 3: High-Level block diagram of the system

2.2 Hardware Dispenser Subsystems and Requirements

2.2.1 Subsystem 1: Battery System

This subsystem powers the device from a rechargeable battery, supports charging via USB, and generates stable rails for logic and actuation. It provides a regulated 3.3 V for ESP32/sensors and raw lithium-ion battery voltage (3-4.2V) to run the stepper motors while preventing brownouts during motor current spikes.

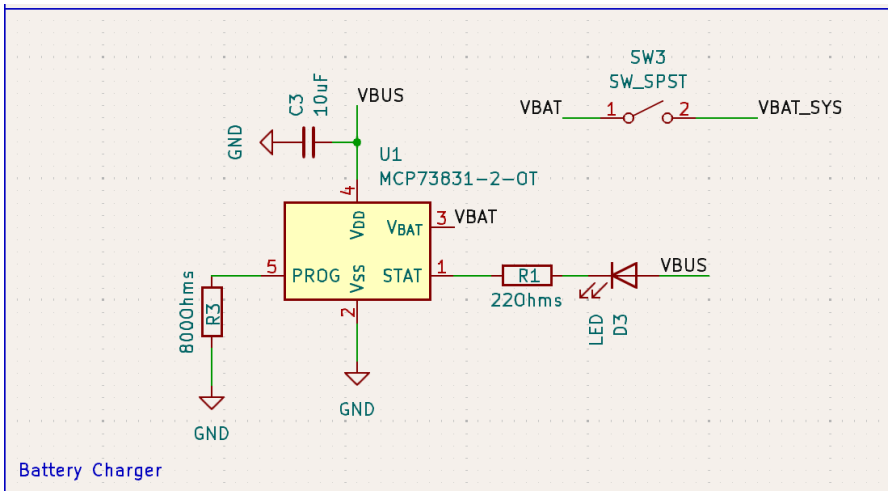
Why Linear Regulator

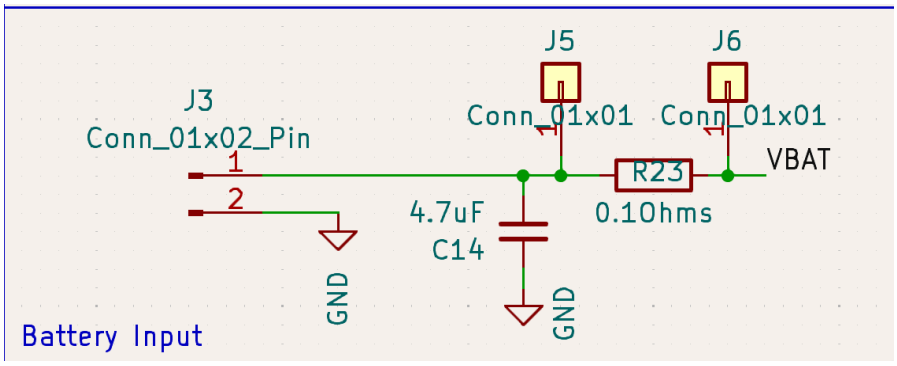
A linear regulator is a fine choice in these circumstances. At full charge, a lithium ion battery is capable of 4.2V, meaning the peak power of a LDO would be (4.2V-3.3V)*I_{max}. From experience with esp32 s3 modules in testing, we have found peak draws for bluetooth low energy transmission and basic sensor usage is not more than around 200 mA peak. Thus that would

make P_{max} = 0.18W. For a buck converter, P_{max} = 3.3V(200mA)/0.9 - 3.3V(200mA) = 0.07W. This difference of 0.11W is nominal relative to the 10WH our battery is capable of.

A second consideration is dropout; the AP2112k has a typical dropout at 300 mA of 0.125V, meaning the functional range of the battery would be 3.425 - 4.2 V. This represents >90% of the lithium ion range, which is acceptable for our design. Using a voltage divider, we can sense battery charge percentage and drop the microcontroller and sensors into sleep mode using software. This will help to protect the battery cells, along with their included protection circuitry in the battery.

The battery charger is currently configured for charging at 1.25A, which is the recommended charging rate for the battery from its datasheet. This is selected from the prog resistor, which governs the charging rate by $I_{reg} = 1000/R_{prog}$. This means a resistor of 800 ohms is chosen.





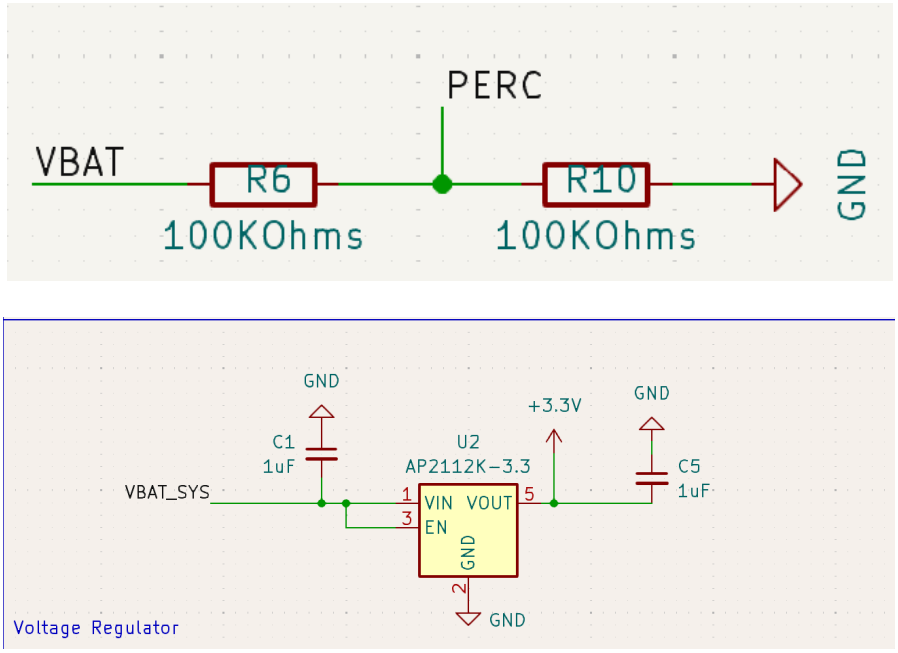


Figure 4: Schematics

Cell Choice

The cell is chosen for its high peak output current (1.5A) and its large battery size. This will allow for long continuous operation without plugging in, which is important for consumer kitchen parts, which are often stored away from power outlets and, as such, should not be too reliant on being plugged in.

Power Connection

To prevent improper battery usage, JST connectors will be used, which only allow the battery to be plugged in the correct direction. The battery will come preassembled and is rechargeable, so the user should not interface with it much.

Interface:

The system interfaces with the larger system by providing all the power used by the sensor and motor subsystem. The regulated 3.3V will power the microcontroller and sensors. The bare battery voltage will drive the motors.

2.2.1.1 R&V

Requirement	Validation
The battery charger must be able to recharge the battery through USB-C	• Using a voltage divider connected to a GPIO, we can obtain the real-time voltage of the battery. • This can be graphed over time to show that the battery is charging despite the draw of the circuit
The voltage regulator must be able to supply 3.3V +/-5% respectively to all other components consistently	Using a multimeter and including test pads for ground plane and 3.3V rail, we can graph voltage under load and across a variety of situations.
The battery must be able to supply enough current to drive the entire system. We anticipate the total system requiring a maximum of 1.5A discharge, mostly to drive the stepper motors.	Using the 0.1 ohm resistor placed across the battery input, we can measure current using ohms law. From this we can find peak supplied current by the battery.

Parts:

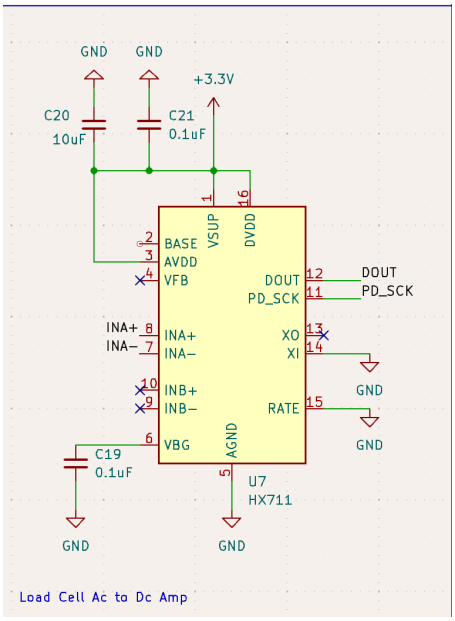
- Battery charger: MCP73831
- 3.3V LDO Voltage Regulator: AP2112K-3.3
- USB-C Port: Generic 16-pin part
- 2500 mAh lithium-ion battery from Adafruit

2.2.2 Subsystem 2: Sensor Subsystem

This subsystem measures ingredient mass for more accurate recipes and measurements. The subsystem also measures the mass of dispensed spices for calibration and accuracy purposes. Finally, IR sensors inside the spice containers detect the level of spice within the container. The ESP32 reads the sensors, filters/calibrates the data (tare + scale factor), and forwards measurements to the main motor controller and web application.

The HX711 is selected for its ample libraries available and ease of use. It is designed for load cells in particular.

The ESP32 S3 Wroom 1 was selected for its large number of GPIOs, its built-in antenna and crystal, and its ability to handle Bluetooth Low Energy mode. The C3 mini was initially considered, but was found not have enough GPIOs to handle the number of sensors being used.



The VL53L4CD was selected for its turnkey readiness as well, with libraries available for its intended usage of level sensing.

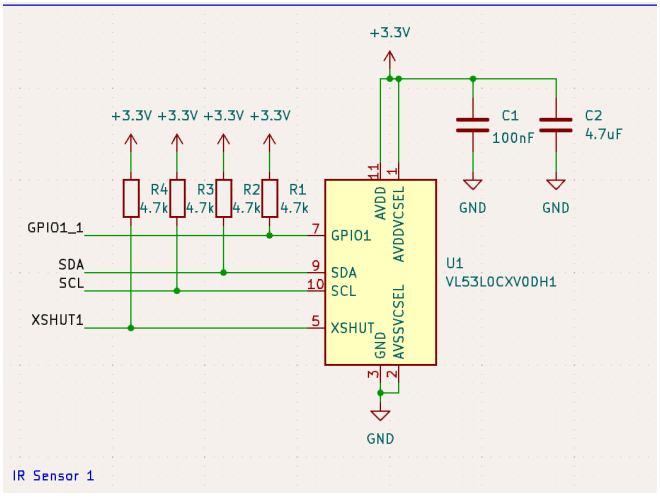


Figure 5: Schematics

2.2.2.1 R&V

Requirement	Validation
The load sensors must be able to weigh the ingredients and spices +/- 1 gram for the spices and +/- 2-3 grams for the ingredients	A measurement using our load cell can be taken. Then, it can be validated using a scale with a known tolerance.
The IR sensors must accurately measure and signal the distance to the spice level, accurate to 1% of the spice container size	IR measurement can be taken. For the container and spice, a full measurement and the measured measurement can be taken, and the percent full can be calculated by dividing the values.
The ESP32 must communicate over GPIO with the sensors and over Bluetooth with the web app to allow for proper readings and control	Data can be flashed to the GPIO, and can then be sent to the web app through Bluetooth. This will validate if the information is being transmitted correctly.

Parts:

- Microcontroller: ESP-32-S3-Wroom-1
- 2x Load Cell Sensors: TAL220B
- IR Sensors: VL53L4CD
- AC to DC amplifier for load cell: HX711

2.2.3 Subsystem 3: Motor Subsystem

This subsystem uses 3 stepper motors to dispense 3 different spices into a central funnel and cup. The motors are used to turn a wide cylinder blocking the opening of the spice containers. The cylinder has a notch in it, allowing a measurable amount of spice to be dispensed upon each rotation. The motors are controlled by the ESP32, using the load cell sensors to dispense proper amounts of spice.

2.2.3.1 R&V

-The motors must be controlled by the ESP32 using GPIO and allow for precise rotations +/- 1°

Requirement	Validation

The motors must be controlled by the ESP32 using GPIO and allow for precise rotations +/- 2º	Can control the motors to step 200 times, which represents a full 360 degree rotation of the motor. Can then measure the difference in starting and final position to ensure negligible drift.
The amount of spice dispensed each rotation should not vary by more than 5% from the mean.	50 samples can be taken, and the mean, standard deviation, and maximum distance from the mean can be found.
The amount of spice lost before reaching the cup should be negligible (within 1% of dispensing without a funnel)	After taking the initial samples without the funnel, 20 more samples can be taken, and the difference in the mean of these samples should be within 1% of the mean of the dispensed spices without the funnel.

Parts:

- Motor Controller 3x: DRV8833
- Stepper Motors: Generic NEMA 17 model

2.3 Vision Subsystems

2.3.1 Communication Subsystem

2.3.1.1 Subsystem Overview

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 / Recipe Planner. It combines user interaction events (e.g., spice/ingredient selection, tare, confirmations, step navigation) with continuous weight telemetry from the ESP32 into a single timestamped event stream and a continuously updated cooking state. This consolidated state is forwarded to the Vision Processor / Recipe Planner so the system can enforce step logic (e.g., “stop at target grams,” “advance to next step,” “add 2g more”) while keeping the AR/VR overlay synchronized with physical actions.
This subsystem is intentionally designed to be minimal and fast: it focuses on low-latency forwarding, basic validation, and robust reconnection rather than heavy computation.

2.3.1.2 System Architecture

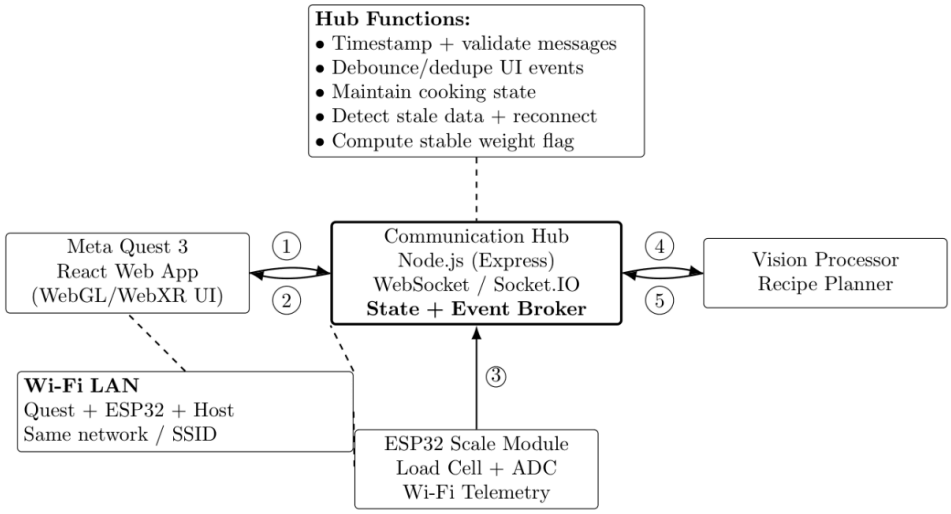


Figure 6. Communication subsystem flow: Quest UI events and ESP32 telemetry are unified by a local Node.js hub into a single cooking state/event stream for the Vision Processor / Recipe Planner, with feedback relayed back to the headset.

① UI Events: select spice, go to next/previous step, confirm, tare
② UI Feedback: target reached, add more/less, warnings
③ Telemetry: weight (grams), sensor status; raw readings converted to filtered readings with a stability flag
④ Forwarded Stream + Snapshots: timestamped events plus the current cooking state
Planner Commands: step updates, target grams, and messages to display on the UI
⑤ Planner Commands: step updates, target grams, and messages to display on the UI
**Note that the Communication Hub will be subject to change that uses easier web services such as FastAPI.*

2.3.1.3 Reliability and Reconnection Strategy

To support uninterrupted cooking sessions, the subsystem should include:

- Automatic reconnection for Quest UI and ESP32 stream
- State resynchronization after reconnect by pushing a full cooking state snapshot
  Message validation and deduplication to prevent repeated UI clicks or retransmitted packets from corrupting state

2.3.1.4 Cooking State Management

The backend maintains a continuously updated cooking state object that is used to synchronize all components. At minimum, it includes:

- Selection State: active ingredient/spice label, and optional target grams
- Scale State: latest weight in grams, tare-applied flag, and timestamps
- Stability State: stable/unstable flag and stable-weight estimate
- Workflow State: current recipe step index, step confirmation status
  Connection Health: Quest connected, ESP32 connected, Planner connected

This state is updated on every incoming UI event or ESP32 message and is forwarded as needed to ensure the planner and UI stay consistent even after reconnects.

2.3.1.5 R&V

Requirements:	*Verification:*
Low-latency event forwarding: Must receive Quest spice-selection/step events over Wi-Fi and forward them to the Vision Processor with end-to-end latency ≤ 150 ms and capacity ≥ 10 events/sec.	Instrument timestamps at Quest-send and planner-receive. Generate a scripted burst of 10 events/sec for 60 s. Verify 95th percentile latency ≤ 150 ms and no dropped events (check unique event IDs).
Connection reliability: Must maintain simultaneous connections to Quest + ESP32 for ≥ 30 minutes and auto-recover from disconnects with reconnect ≤ 5 s.	Run a 30-minute soak test with ESP32 streaming weight at 10–20 Hz and Quest sending periodic events. Force disconnects (disable ESP32 Wi-Fi; close Quest browser). Verify reconnection ≤ 5 s and state snapshot resynchronizes UI/planner correctly.

Parts:

- Meta Quest 3
- Wi-Fi

2.3.2 Vision Processor Subsystem

2.3.2.1 Overview

The Vision Processor Subsystem performs NuChef's core perception by converting Meta Quest 3 RGB frames into a structured, real-time ingredient representation used by the downstream

recipe-planning logic. Earlier multi-stage designs (detect → crop → classify) introduce extra latency and compounded errors, so our final design uses a single YOLOv11 model to jointly perform detection and classification in one forward pass by referring to the original proposed YOLO architecture's usage of GoogleNet[4].. The model is trained to recognize our top-level ingredient categories, protein and vegetable, and outputs a bounded list of detections (class + confidence + location) at real-time frame rates. This output is formatted as a structured ingredient list and published to the planner so the user can receive immediate, step-by-step guidance in the headset overlay.

Output format wise: For each processed frame, the subsystem outputs a set of detections

$$\{(c, p, b)\}$$

where $c \in \{protein, vegetable\}$, $p \in [0, 1]$ is the confidence score, and b=(x,y,w,h) is the bounding box. If a segmentation variant is used later, b can be replaced by a mask m, but the current design assumes detection boxes for simplicity and speed. To reach a single-step detection and classification, YOLO is, in our knowledge, the best candidate.

2.3.2.2 Vision Processor Workflow Diagram

Here is a much detailed plan for both online and offline implementation of our vision processor with the zoomed-in illustration of YOLOv11 in our usage.

Online inference	Offline Training
Frame ingestion over Wi-Fi: Quest frames arrive and enter a buffer.	Collect dataset.
Frame buffer + rate control: Frames are processed at a fixed target rate (e.g., 10 FPS) to prevent backlog and keep latency stable.	Annotate detections and map labels into the two target classes.
Preprocessing: Resize/normalize (optional crop for the cooking workspace).	Split/augment for blur, low light, occlusion, and scale variation.
YOLOv11 inference: Single pass producing candidate detections in two classes (protein, vegetable).	Fine-tune YOLOv11 on our GTX PCs
Postprocessing: Confidence thresholding and NMS (and a cap on maximum detections, e.g., ≤ 20) to bound compute time and reduce clutter.	Evaluate on held-out data using mAP and per-class metrics.
Structured output: Publish {(c,p,b)}to the planner/UI.	Export/deploy the best weights.

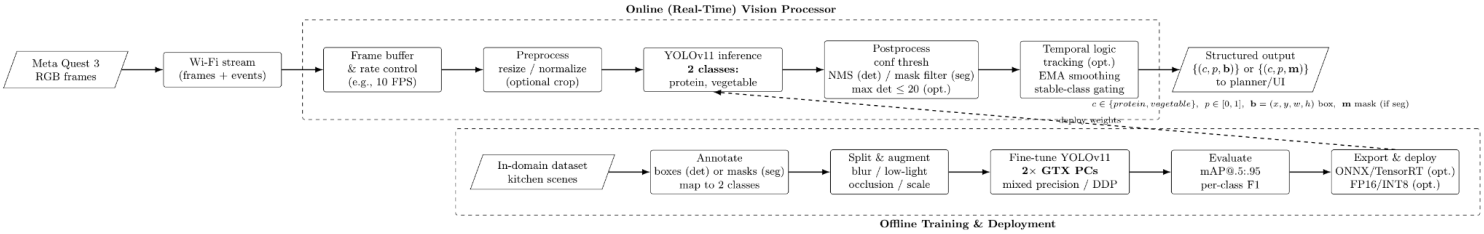


Figure 7: Overall Vision Processor Workflow

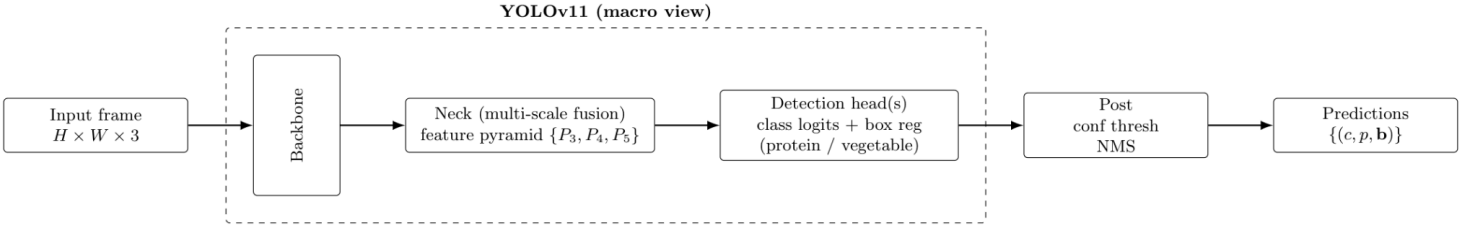


Figure 8: YOLOv11 CNN

2.3.2.3 LLM Agent

The LLM-based Recipe Planner is responsible for converting the Vision Processor's perception output into an actionable cooking procedure. While the Vision Processor focuses on real-time detection (protein/vegetable with confidence and location), the LLM agent performs higher-level reasoning: selecting an appropriate recipe, generating step-by-step instructions, and adapting those instructions as the detected ingredient set changes. The resulting instructions and alerts are then sent back to the headset interface for presentation as an AR/VR overlay.

2.3.2.4 R&V

Requirements:	***Verification:***
Detection/segmentation output quality: For each processed frame, must output ≤ 20 ingredient regions with (x, y, w, h) boxes or pixel masks, and include confidence scores; must achieve ≥ 0.425 mAP.	Run inference on a representative test set and confirm that postprocessing enforces max detections ≤ 20. Evaluate the trained YOLOv11 model on the validation dataset and report mAP at 0.5~0.95
Recognition + interface to planner: For every region, must output a top-1 ingredient label	Testing on the validation dataset and pass if overall accuracy is 70%. Then compute a

and confidence, with top-1 accuracy≥ 70% on a held-out validation set; then must publish the structured ingredient list to the LLM/planner.	confusion matrix.

Parts:

- NVIDIA RTX
- Candidate region model: YOLO11 (det/seg) (bounding boxes or segmentation)
- Vision-language classifier: CLIP / OpenCLIP / MobileCLIP (region embedding + text embedding matching)
- LLM agent (recipe selection + instruction generation using detected ingredients)

2.4 Tolerance Analysis:

The load cell we are likely to use is the TAL220B. For the 1 KG model, there are a few specs that could be concerning. The repeatability is 0.05%, meaning the spread in measuring an identical unit many times could be as great as 0.5 grams in each direction. Likewise, the creep would be up to 1 gram per 3 minutes. There is also non-linearity, which at the gram range measurements we will be doing can be an issue, which is 0.05%, or around 0.5 grams of drift from the linear calibration at the midpoint. To alleviate this, we will calibrate the scale in the 1-10 gram range, which will remove the non-linearity effect in the range, but introduce a higher error in the 100-1000 gram range. This is acceptable, as we do not expect to measure spices in this quantity. Creep will be alleviated by quick dispensing, and also the mechanical fail-safe of calibrating the dispenser to dispense the correct amount, allowing the load cell to be a backup, and to allow the user to add new spices of unknown weight. This means we anticipate a combined error in dispensing of no more than 1 gram, accounting for load cell error, mechanical error, and deviance across samples.

3. Cost and Schedule

3.1 Cost

We are renting the Meta Quest 3 from the Library, though a marketable version of our project would require one to be rented/owned. Without the Meta Quest, the overall price of the parts in the project as seen below is $146.01. 10% tax adds $14.60 and 5% shipping adds $7.30, bringing the overall price to $167.91. We would consider a weekly salary of $40/hr * 2.5 * 60 (hours to complete) = $6000 per team

member. $6000*3 = $18000 in overall labor cost. The machine shop charges $70/hr * 15 (hours to complete) = $1050. Summing this all, we get $167.91 + $18000 + $1050 = $19,217.91.

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: Generic NEMA 17 model	StepperOnline	3	$32.97

3.2 Schedule

Date	Software	Hardware
Mar. 2nd	App: Basic backend for Meta ESP communication. Start ESP programming (or similar board before ESP arrival) for motor control and sensor data transmission to the app. Validate that YOLOv11 inference script runs	PCB schematic and routing fully completed. PCB must be ordered to arrive on time. Begin working on Breadboard demo.

	end-to-end for sample images.	
Mar. 9th (Breadboard Demo)	Demonstrate motor stepping + HX711 load cell reading + IR sensor readout. App: Web dashboard to display live weight + send motor commands (dispense/stop/tare).	Full Breadboard ready for the breadboard demo. This will include the motors, IR sensors, and load cells. Dev Boards may be used in this stage of the project.
-Spring Break-	Continuation on vision development and clean up dataset. Train the basic YOLOv11 and record base metrics. Define LLM output schema.	Mechanical work that plans out drilling holes for mounting devices.
Mar. 23rd	Integration: End-to-end pipeline test: UI event → ESP dispense command → weight telemetry → UI feedback. Add safety limits (timeouts, max steps). Vision Integration: Run YOLOv11 live on stream or recorded loop and output {(c,p,b)} JSON. Add simple temporal smoothing to reduce flicker.	PCB: Assemble and test rails (battery/charger/3.3V), then sensors, then motor drivers. Mech: Machine shop iteration #2: updated design based on PCB and motor constraints.
Mar. 30th	Demo-Ready Software: Quest UI connected to backend; commands work reliably. Vision: YOLOv11 runs at target FPS with capped detections (≤20). Planner: LLM produces structured step list and updates UI when ingredient set changes (stable detections).	PCB Debugging and Testing Continuation.
Apr. 6th (Progress Demo)	Hardening: Improve error handling (sensor disconnect, noisy readings, motor stall). Add calibration workflow (tare + scale factor + per-spice motor calibration). Vision: Expand dataset, retrain YOLOv11 v2, improve	Progress Demo Hardware: Integrated dispenser can dispense to target grams (show a few trials). Sensors stable and streamed continuously for demo duration (≥10 min).

	accuracy and stability. Planner: Add guardrails to prevent recipe thrashing; include safety notes for risky steps.	
Apr. 13th	Mock Demo Script: Full run-through with logging. Verify reconnect ≤5s and state restores cleanly. Finalize thresholds (conf/NMS, stable-weight rules). Prepare plots/tables for verification evidence.	Refinement: Secure wiring, improve funnel alignment, reduce spillage. Battery runtime + charging verification. More repeatability trials (≥50 samples for one spice).
Apr. 20th (Mock Demo)	Freeze Features: Only bug fixes + performance tuning. Collect final verification results: dispensing error stats, latency logs, vision metrics (mAP/accuracy). Package final demo build + backups (recorded stream fallback).	Mock Demo Hardware: Reliable end-to-end demonstration (dispense-to-target + live UI + vision output + planner steps). Final cable management and safety checks.
Apr. 27th (FINAL DEMO)	Demo	DEMO

4. Discussion of Societal Impact, Engineering Standards, Ethics, and Safety Considerations

Our project is beneficial to society as it helps to 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.

Working with a 3.7V Lithium battery with power scaling modules in the design, we must consider electrical safety. 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. We will also partake in special safety training on the use of Lithium batteries, as they are generally more flammable and can

cause chemical fires. 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. 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 purpose,s we would want a rating above 100 °C to allow 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 to avoid harm. Since cooking involves a lot of sharp and hot objects, we must avoid hazards for the user. 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 may also have to suggest that the person remove the headset while they are cooking on an open flame. While it does make it more convenient to have the headset remain on, it could affect the chef's safety to have the recipe and any other things in their vision while cooking. These are considerations that we have to make to make sure users can avoid harm from our product.

Overall, our product is safe on both the manufacturer and user side. Our battery is low-voltage, and overall electrical injuries are preventable as the user does not need to directly interact with the PCB or wires in any way. Below are documents on safe battery handling and Battery Safety: https://courses.grainger.illinois.edu/ece445/documents/GeneralBatterySafety.pdf
Safe Current Limits:
https://courses.grainger.illinois.edu/ece445/documents/Safe_Current_Limits.pdf

5. References

[1] “ACM Code of Ethics and Professional Conduct” (2018), ACM, https://www.acm.org/code-of-ethics (accessed Feb 13th, 2026)

[2] “What are food contact substances and what’s their relationship to FDA compliance?” (2020), ISM, https://www.industrialspec.com/about-us/blog/detail/fda-compliant-food-grade-food-safe-meanings (accessed Feb 13th, 2026)

[3] “Guidance for Industry: Preparation of Premarket Submissions for Food Contact Substances (Chemistry Recommendations)” (2007), U.S. Food & Drug Administration, https://www.fda.gov/regulatory-information/search-fda-guidance-documents/guidance-industry-preparation-premarket-submissions-food-contact-substances-chemistry (accessed Feb 13th, 2026)

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *arXiv preprint* arXiv:1506.02640, Jun. 2015, doi: 10.48550/arXiv.1506.02640.