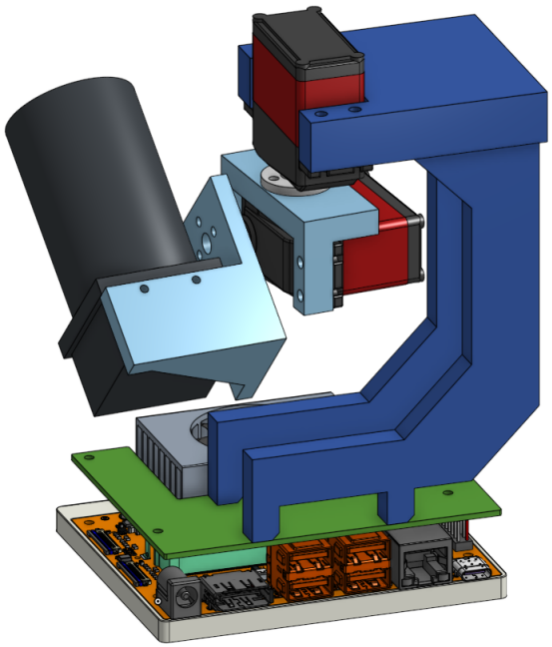
Team Contributions: POC
RoCam

Team #3, SpaceY
Zifan Si
Jianqing Liu
Mike Chen
Xiaotian Lou

November 10, 2025

This document summarizes the contributions of each team member up to the POC Demo. The time period of interest is the time between the beginning of the term and the POC demo.

1 Demo Plans



We will present the proof-of-concept demonstration according to the plan in the Development Plan document. One of the team members will be in charge of the demonstration.

2 Team Meeting Attendance

Student	Meetings
Total	17
Xiaotian Lou	17
Zifan Si	17
Jianqing Liu	17
Shike Chen	17

3 Supervisor/Stakeholder Meeting Attendance

Supervisor's Name: Sirouspour, Shahin

Student	Meetings
Total	3
Xiaotian Lou	3
Zifan Si	3
Jianqing Liu	3
Shike Chen	3

4 Lecture Attendance

Student	Lectures
Total	14
Xiaotian Lou	8
Zifan Si	6
Jianqing Liu	5.5
Shike Chen	5

5 TA Document Discussion Attendance

TA's Name: Tanya Djavaherpour

Student	Lectures
Total	3
Xiaotian Lou	3
Zifan Si	3
Jianqing Liu	3
Shike Chen	3

6 Commits

Student	Commits	Percent
Total	280	100%
Xiaotian Lou	67	23.9%
Zifan Si	88	31.4%
Jianqing Liu	100	35.7%
Shike Chen	25	8.9%

Notes. Commit counts reflect merges to the `main` branch only. During the POC period, Shike Chen and Xiaotian Lou focused on model R&D and large-scale training/ablation (e.g., DINO and YOLO variants). Much of this work involved exploratory code, training scripts, and logs that reside in experimental/backup branches and on the university H100 "Grace" cluster rather than the repository's `main`; as a result, their commit totals under-represent effort. We trained YOLO weights for > 30 hours and accumulated substantial debugging/tuning experience across software, hardware, and environment layers.

We therefore interpret the raw commit distribution with caution. When accounting for model R&D and training conducted outside `main`, the overall contribution across team members is roughly balanced. Reproducible training scripts and experiment summaries will be upstreamed in subsequent PRs.

7 Issue Tracker

Student	Authored (O+C)	Assigned (C only)
Xiaotian Lou	17	10
Zifan Si	13	16
Jianqing Liu	46	20
Shike Chen	2	6

Notes.

- **Counting rule:** "Authored (O+C)" counts issues opened by a member (open or closed). "Assigned (C only)" counts issues where the member was the assignee at closure.
- **Role-based workflow:** During the POC period, Jianqing Liu (project manager, "Pega") centrally opened, assigned, and closed most issues based on team decisions captured in meeting notes; hence his "Authored" count is higher.
- **Credit vs. logistics:** Centralized opening/closing is administrative. Actual implementation credit is reflected in commits, PRs, and code reviews; an issue can have multiple contributors beyond the final assignee.
- **Assignment semantics:** Ownership may change during an issue's lifetime; we attribute "Assigned (C only)" to the final assignee at closure to represent delivery responsibility.

8 CICD

We use GitHub Actions with four workflows that cover documentation, firmware, and the operator UI.

build-tex (push to `main`, `docs/**`, or manual) Compiles LaTeX with TeX Live 2025 on Ubuntu and commits the built PDFs back to the repository. The job runs `make -B` under `docs/` and then auto-commits any updated `.pdf` files (`contents: write`).

check-tex (pull requests touching `docs/**`) Validates that the LaTeX compiles on PRs, uploads any changed PDFs as an artifact, then formats all `.tex` sources with `latexindent` (80-column wrapping) and commits the formatting changes. This keeps the documentation reproducible and consistently formatted.

firmware-ci (push/PR under `src/pcb-firmware/**`) Builds the embedded firmware with Cargo on Ubuntu and uploads the resulting binary as an artifact. The job also enforces Rust source formatting via `cargo fmt --check`.

react-ci (push/PR for the React app) Builds and smoke-tests the operator UI on Node 20. Dependencies are installed with `npm ci`; a lightweight `ci:smoke` runs on every change. Unit tests and production build currently run in *best-effort* mode (non-blocking), and any build artifacts are uploaded when present.

By POC close, we will deliver substantially deeper, React-focused UI tests (components, hooks, interactions) if we finish full stack intergration.

- **Component tests** with React Testing Library + Jest/Vitest: role-based queries (no test-ids), user interactions via `@testing-library/user-event`, keyboard/focus flows, and accessibility checks with `jest-axe`.
- **State & hooks** coverage: test reducers, Context providers, and custom hooks (e.g., `useGimbal`, `useVideo`) using `@testing-library/react` and `@testing-library/react-hooks`; assert stable re-renders and memoization (`React.memo`, `useCallback`, `useMemo`).
- **Coverage gates**: enforce $\geq$80% lines/branches on UI packages; coverage upload and `--coverageReporters` in CI; mark unit+integration jobs as *required*.
- **Linting/consistency**: `eslint` with `eslint-plugin-react` and `eslint-plugin-jsx-a11y`; `prettier --check` as a required gate.

Near-term roadmap (to be enforced via branch protection)

- **Formatting gates:** add `black --check` for Python sources and `prettier --check` for the React codebase. If formatting is not clean, the PR will fail and be blocked from merging.
- **Cross-platform test matrix:** add a Python test job that runs the full unit test suite (15 tests) on {`ubuntu-latest`, `macos-latest`, `windows-latest`} $\times$ Python {3.9, 3.10, 3.11, 3.12, 3.13}. The job will install dependencies and execute the tests; failures will block merges.

9 Team Charter Trigger Items

Quantified Triggers (per Team Charter)

- **Meeting cadence & attendance:** weekly team meetings; prior notice required for any absence; target attendance $\geq$85%.
- **Preparation & quality:** come to meetings with concise updates, blockers, and options; maintain decision logs and lightweight RACI notes.
- **PR/issue hygiene:** every task has a GitHub issue with an owner and estimate; all code changes go through PRs that reference an issue; no direct pushes to `main`.
- **Review SLA & CI gates:** at least one reviewer approval; address review comments within 7 days (unless a different SLA is agreed); CI must be green (docs build, firmware build, UI smoke).
- **Documentation standards:** LaTeX compiles reproducibly on CI; PDFs are published on `main`; `latexindent` enforces consistent formatting on PRs.

- **Runtime KPIs (demo readiness):** track end-to-end latency (p50/p95), FPS, target re-acquisition time, crash-free session rate, and UI task success; for the POC demo we target $\sim$100–500 ms latency, $\geq$15 FPS, and $\leq$2 s re-acquisition.
- **Underperformance trigger:** if a member underdelivers for >3 weeks, initiate pairing/guardrails and, if needed, escalate to TA/instructor.

Violations Observed During the POC Period

None observed. All members met the charter triggers: weekly meetings were held with advance notice for any conflicts; PRs referenced issues and passed CI; review comments were addressed within the 7-day SLA; documentation built reproducibly; runtime KPIs were recorded for the demo.

Plan to Sustain Compliance

- Keep branch protection (no direct pushes to `main`; $\geq$1 review; CI required).
- Promote formatting to required checks: `black --check` (Python) and `prettier --check` (React).
- Add a cross-platform Python test matrix (Ubuntu/macOS/Windows; Python 3.9–3.13) as a required status check.
- Use a meeting-issue template (owner, ETA, blockers) and maintain the decision log to preserve cadence and traceability.
- Clarify KPI thresholds for upcoming milestones and monitor p50/p95 latency, FPS, and re-acquisition time on each demo run.

10 Additional Productivity Metrics

At this time, we have no additional productivity metrics to report.