10 Architecture Decision Records

2025-10-19

Contents

1 Architecture Decision Records (ADR)

1.1 Vision-Based Pick and Place Robotic System

1.2 ADR Index

ADR #	Title	Status	Date
ADR-001	Use ROS2 Humble as Middleware	Accepted	2025-10-18
ADR-002	Select CycloneDDS over FastDDS	Accepted	2025-10-18
ADR-003	Choose PyTorch over TensorFlow for AI Models	Accepted	2025-10-18
ADR-004	Use MoveIt2 for Motion Planning	Accepted	2025-10-18
ADR-005	Adopt ros2_control for Real-Time Control	Accepted	2025-10-18
ADR-006	Use PostgreSQL as Primary Database	Accepted	2025-10-18
ADR-007	Implement State Machine with BehaviorTree.CPP	Accepted	2025-10-18
ADR-008	Deploy YOLOv8 for Object Detection	Accepted	2025-10-18
ADR-009	Use Docker for Deployment	Accepted	2025-10-18
ADR-010	Adopt ELK Stack for Centralized Logging	Accepted	2025-10-18
ADR-011	Use Grafana + Prometheus for Monitoring	Accepted	2025-10-18
ADR-012	Implement OAuth2 + JWT for Authentication	Accepted	2025-10-18
ADR-013	Choose Intel RealSense D435i Camera	Accepted	2025-10-18
ADR-014	Select UR5e Robot Arm	Accepted	2025-10-18
ADR-015	Use NVIDIA Jetson Xavier for Vision Processing	Accepted	2025-10-18

1.3 ADR Template

```
# ADR-XXX: [Title]

## Status
[Proposed | Accepted | Rejected | Deprecated | Superseded by ADR-YYY]

## Context
[Description of the problem and why a decision is needed]

## Decision
[The decision that was made]

## Consequences
### Positive
- [Benefit 1]
- [Benefit 2]

### Negative
- [Drawback 1]
- [Drawback 2]

### Risks
- [Risk 1 + mitigation]

## Alternatives Considered
1. **[Alternative 1]:** [Brief description] - Rejected because [reason]
2. **[Alternative 2]:** [Brief description] - Rejected because [reason]

## References
- [Link to documentation, discussion, or research]

## Metadata
- **Author:** [Name]
- **Date:** [YYYY-MM-DD]
- **Reviewers:** [Names]
- **Related ADRs:** [ADR-XXX, ADR-YYY]
```

1.4 ADR-001: Use ROS2 Humble as Middleware

1.4.1 Status

Accepted

1.4.2 Context

The system requires a middleware framework to enable communication between distributed modules (vision, planning, control, monitoring). Options include ROS2, ROS1, YARP, OROCOS, and custom middleware.

Requirements: - Real-time support for 1 kHz control loops - Mature ecosystem (motion planning, drivers) - Long-term support (LTS) - Security features (authentication, encryption) - Cross-platform (Linux, Windows)

1.4.3 Decision

Adopt ROS2 Humble Hawksbill (LTS release) as the primary middleware.

1.4.4 Consequences

1.4.4.1 Positive

- **LTS Support:** Maintained until May 2027 (5 years from release)
- **Real-Time:** Supports RT-Preempt Linux, DDS for low-latency communication
- **Ecosystem:** MoveIt2, ros2_control, image_transport, tf2 all available
- **Security:** Built-in DDS security (SROS2), vs ROS1's insecure TCPROS
- **Active Development:** Strong community, frequent updates

1.4.4.2 Negative

- **Learning Curve:** Team must learn ROS2 (vs ROS1 familiarity)
- **Tooling Maturity:** Some ROS1 tools (rqt plugins) still being ported
- **Breaking Changes:** API changes between ROS2 distributions (mitigated by LTS)

1.4.4.3 Risks

- **Vendor Lock-In:** ROS2-specific code not portable to non-ROS systems
  - *Mitigation:* Abstract core logic into ROS-agnostic libraries, use ROS2 as thin wrapper

1.4.5 Alternatives Considered

Alternative	Pros	Cons	Decision
ROS1 (Noetic)	Mature, familiar	EOL 2025, no real-time, insecure	Rejected: No future
YARP	Good real-time	Small community, fewer packages	Rejected: Ecosystem gap
OROCOS	Excellent real-time	Steep learning curve, niche	Rejected: Overkill
Custom	Full control	High development cost, reinvent wheel	Rejected: Not cost-effective

1.4.6 References

- ROS2 Humble Documentation
- ROS2 vs ROS1 Comparison

1.4.7 Metadata

- **Author:** System Architect
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead, CTO
- **Related ADRs:** ADR-002 (DDS choice), ADR-004 (MoveIt2), ADR-005 (ros2_control)

1.5 ADR-002: Select CycloneDDS over FastDDS

1.5.1 Status

Accepted

1.5.2 Context

ROS2 supports multiple DDS implementations (rmw): CycloneDDS, FastDDS, RTI Connext, etc. The choice affects network performance, latency, and throughput.

Requirements: - Low latency (<10ms for control messages) - High throughput (camera images @ 30fps) - Small network (1-3 machines), not WAN - Open-source (no licensing fees)

1.5.3 Decision

Use CycloneDDS as the ROS2 middleware (rmw_cyclonedds_cpp).

1.5.4 Consequences

1.5.4.1 Positive

- **Performance:** Benchmarks show 20-30% lower latency than FastDDS on small networks
- **Simplicity:** Easier configuration (fewer tuning parameters)
- **Reliability:** Stable, fewer known bugs
- **License:** Eclipse Public License (EPL) – permissive

1.5.4.2 Negative

- **Features:** Lacks some advanced FastDDS features (discovery server, DDS-Router)
  - *Not critical for single-site deployment*
- **Ecosystem:** Smaller community vs FastDDS (ROS2 default)

1.5.4.3 Risks

- **Compatibility:** Future ROS2 packages might optimize for FastDDS
  - *Mitigation:* Easy to switch rmw via environment variable, test both

1.5.5 Alternatives Considered

Alternative	Pros	Cons	Decision
FastDDS	ROS2 default, most tested	Higher latency, complex config	Rejected: Performance gap
RTI Connext	Enterprise-grade, best performance	Commercial license ($$$)	Rejected: Cost prohibitive
Custom DDS	Tailored to needs	Massive development effort	Rejected: Not feasible

1.5.6 References

- ROS2 DDS Benchmark
- CycloneDDS GitHub

1.5.7 Metadata

- **Author:** Network Engineer
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead
- **Related ADRs:** ADR-001 (ROS2 choice)

1.6 ADR-003: Choose PyTorch over TensorFlow for AI Models

1.6.1 Status

Accepted

1.6.2 Context

The vision pipeline requires deep learning models for object detection and pose estimation. Major frameworks: PyTorch, TensorFlow, JAX.

Requirements: - Research-friendly (iterate on models) - Good ONNX export (for TensorRT deployment) - Strong community in robotics/vision - Python-first API

1.6.3 Decision

Use PyTorch 2.0+ for model development and training.

1.6.4 Consequences

1.6.4.1 Positive

- **Dynamic Graphs:** Easier debugging, more Pythonic
- **Research Adoption:** Most recent papers use PyTorch (YOLOv8, PVNet)
- **ONNX Export:** Excellent support via `torch.onnx`
- **TensorRT:** Smooth PyTorch → ONNX → TensorRT pipeline

- **Community:** Large robotics community (ROS + PyTorch common)

1.6.4.2 Negative

- **Production Deployment:** TensorFlow Lite slightly more mature for embedded
  - *Mitigated by TensorRT path*
- **Google Ecosystem:** Less integration with TensorBoard (but supported)

1.6.4.3 Risks

- **Model Compatibility:** Some TensorFlow models hard to port to PyTorch
  - *Mitigation:* Use ONNX as intermediate format

1.6.5 Alternatives Considered

Alternative	Pros	Cons	Decision
TensorFlow	Production-ready, TF Lite	Static graphs, less research-friendly	Rejected: Harder to iterate
JAX	Functional, fast	Smaller ecosystem, steep curve	Rejected: Too niche
ONNX Only	Framework-agnostic	No training, only inference	Rejected: Need training pipeline

1.6.6 References

- PyTorch Official
- TensorRT PyTorch Workflow

1.6.7 Metadata

- **Author:** AI/ML Lead
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead, Data Scientist
- **Related ADRs:** ADR-008 (YOLOv8 choice), ADR-015 (Jetson Xavier)

1.7 ADR-004: Use MoveIt2 for Motion Planning

1.7.1 Status

Accepted

1.7.2 Context

Motion planning requires computing collision-free trajectories from current to goal configuration. Options: MoveIt2, OMPL standalone, Pilz, custom planner.

Requirements: - Collision checking with environment - Inverse kinematics (IK) solvers - Integration with ROS2 - Support for UR5e robot

1.7.3 Decision

Adopt MoveIt2 as the motion planning framework.

1.7.4 Consequences

1.7.4.1 Positive

- **Comprehensive:** IK, planning, collision checking, scene management in one package
- **OMPL Integration:** Access to RRT*, PRM, etc.
- **URDF Support:** Load robot model easily
- **ROS2 Native:** Full integration with ros2_control, TF2
- **Active Development:** Continuous improvements, bug fixes

1.7.4.2 Negative

- **Complexity:** Large codebase, steep learning curve
- **Performance:** Can be slow for complex scenes (>500 obstacles)
  - *Mitigated by scene simplification, octomaps*
- **Overhead:** Heavy for simple point-to-point moves
  - *Mitigated by direct IK for simple cases*

1.7.4.3 Risks

- **Breaking Changes:** MoveIt2 API still evolving (Humble → Jazzy)
  - *Mitigation:* Stick with Humble LTS, test upgrades carefully

1.7.5 Alternatives Considered

Alternative	Pros	Cons	Decision
OMPL Standalone	Lightweight, fast	No IK, no ROS integration	Rejected: Missing features
Pilz Industrial	Deterministic, real-time	Limited to simple motions (PTP, LIN)	Rejected: Not flexible enough
Custom Planner	Tailored to task	High development cost	Rejected: Reinventing wheel
Descartes	Cartesian planning	Trajectory-only, no collision	Rejected: Narrow use case

1.7.6 References

- MoveIt2 Documentation
- OMPL Documentation

1.7.7 Metadata

- **Author:** Robotics Engineer
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead
- **Related ADRs:** ADR-001 (ROS2), ADR-005 (ros2_control)

1.8 ADR-005: Adopt ros2_control for Real-Time Control

1.8.1 Status

Accepted

1.8.2 Context

Real-time motor control requires deterministic communication with drives at 1 kHz. Options: ros2_control, custom control loop, vendor-specific SDK.

Requirements: - 1 kHz control loop - Hardware abstraction (portable across robots) - Support for PID, trajectory, admittance controllers - Integration with MoveIt2

1.8.3 Decision

Use ros2_control framework with custom hardware interface for UR5e (or EtherCAT drives).

1.8.4 Consequences

1.8.4.1 Positive

- **Standardization:** Controller plugins (PID, trajectory) reusable across robots
- **Hardware Abstraction:** Swap robot without changing high-level code
- **MoveIt2 Integration:** Seamless via FollowJointTrajectory action
- **Real-Time:** Designed for RT-Preempt Linux
- **Community Controllers:** Pre-built controllers (joint, Cartesian, admittance)

1.8.4.2 Negative

- **Complexity:** Requires implementing `HardwareInterface` for custom robots
- **Debugging:** Harder to debug than simple control loop
- **Overhead:** Slight latency vs direct motor driver access (~1-2ms)

1.8.4.3 Risks

- **RT Performance:** Non-RT code in hardware interface can break real-time
  - *Mitigation:* Careful profiling, isolate RT-critical code

1.8.5 Alternatives Considered

Alternative	Pros	Cons	Decision
Custom Loop	Full control, minimal overhead	No standardization, hard to maintain	Rejected: Not scalable
Vendor SDK (UR API)	Optimized for UR5e	Vendor lock-in, no ROS integration	Rejected: Not portable
Orocos RTT	Best real-time performance	Steep learning curve, small community	Rejected: Overkill

1.8.6 References

- ros2_control Documentation
- Hardware Interface Tutorial

1.8.7 Metadata

- **Author:** Controls Engineer
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead, Robotics Engineer
- **Related ADRs:** ADR-001 (ROS2), ADR-004 (MoveIt2)

1.9 ADR-006: Use PostgreSQL as Primary Database

1.9.1 Status

Accepted

1.9.2 Context

Need a relational database for persistent storage (tasks, users, configs). Options: PostgreSQL, MySQL, SQLite.

Requirements: - ACID compliance - JSON support (for flexible schemas) - Open-source, mature - Good performance (<100ms query)

1.9.3 Decision

Use PostgreSQL 15 as the primary relational database.

1.9.4 Consequences

1.9.4.1 Positive

- **JSONB Support:** Native JSON column type (fast indexing)
- **Extensibility:** PostGIS (geospatial), pg_trgm (full-text search)
- **Performance:** Better than MySQL for complex queries

- **Community:** Large, active community
- **License:** PostgreSQL License (permissive)

1.9.4.2 Negative

- **Complexity:** More config options vs SQLite
- **Resource Usage:** Heavier than SQLite (~50MB RAM baseline)

1.9.4.3 Risks

- **Scaling:** Single-master write bottleneck (not a concern for this project)

1.9.5 Alternatives Considered

Alternative	Pros	Cons	Decision
MySQL	Ubiquitous, familiar	Weaker JSON support	Rejected: JSONB advantage
SQLite	Lightweight, serverless	No concurrent writes, no network	Rejected: Multi-user needs
MongoDB	Native JSON, schemaless	No ACID, harder to join	Rejected: Need relational integrity

1.9.6 References

- PostgreSQL Documentation
- PostgreSQL vs MySQL

1.9.7 Metadata

- **Author:** Backend Engineer
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead
- **Related ADRs:** None

1.10 ADR-007: Implement State Machine with BehaviorTree.CPP

1.10.1 Status

Accepted

1.10.2 Context

Task orchestration requires high-level state management (IDLE → SCAN → DETECT → PICK → PLACE). Options: FSM (SMACH), Behavior Trees, custom code.

Requirements: - Modular (reusable behaviors) - Reactive (respond to events) - Human-readable (visualize logic) - Easy error handling

1.10.3 Decision

Use BehaviorTree.CPP for task orchestration.

1.10.4 Consequences

1.10.4.1 Positive

- **Modularity:** Behaviors are reusable nodes (action, condition, decorator)
- **Composability:** Build complex trees from simple nodes
- **Visualization:** Groot tool for real-time tree visualization
- **Error Handling:** Built-in fallback, retry nodes
- **Performance:** C++ implementation, fast
- **Industry Adoption:** Used in industrial robotics (Nvidia Isaac, ROS2)

1.10.4.2 Negative

- **Learning Curve:** BT paradigm different from FSM
- **Debugging:** Tree execution harder to trace vs linear FSM

1.10.4.3 Risks

- **Complexity:** Overly complex trees hard to understand
  - *Mitigation:* Keep trees shallow (<5 levels), document well

1.10.5 Alternatives Considered

Alternative	Pros	Cons	Decision
SMACH (ROS1)	Familiar, simple FSM	Not ported to ROS2, linear flow only	Rejected: No ROS2 support
FlexBE	GUI editor	Heavy, overkill for simple tasks	Rejected: Too complex
Custom FSM	Simple, clear	Not reusable, hard to extend	Rejected: Reinventing wheel

1.10.6 References

- BehaviorTree.CPP
- Groot Visualization

1.10.7 Metadata

- **Author:** Software Architect
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead, Robotics Engineer
- **Related ADRs:** ADR-001 (ROS2)

1.11 ADR-008: Deploy YOLOv8 for Object Detection

1.11.1 Status

Accepted

1.11.2 Context

Vision pipeline requires real-time object detection. Options: YOLOv5/v8, Faster R-CNN, SSD, EfficientDet.

Requirements: - Real-time inference (<50ms on Jetson Xavier) - High accuracy (mAP >90%) - Easy to train on custom datasets - ONNX export support

1.11.3 Decision

Use YOLOv8 (Ultralytics) for object detection.

1.11.4 Consequences

1.11.4.1 Positive

- **Speed:** 30+ FPS on Jetson Xavier (TensorRT)
- **Accuracy:** State-of-the-art mAP (95%+ on COCO)
- **Ease of Use:** Simple Python API (`ultralytics` package)
- **Custom Training:** One command to fine-tune on custom data
- **Export:** Native ONNX/TensorRT export
- **Active Development:** Frequent updates from Ultralytics

1.11.4.2 Negative

- **Model Size:** YOLOv8-large is 80MB (vs 20MB for YOLOv5-small)
  - *Mitigated by YOLOv8-nano variant (6MB)*
- **Dependency:** Relies on Ultralytics library (could change API)

1.11.4.3 Risks

- **License:** AGPL-3.0 (must open-source modifications)
  - *Mitigation:* Use as black box, no modifications (or buy enterprise license)

1.11.5 Alternatives Considered

Alternative	Pros	Cons	Decision
Faster R-CNN	High accuracy	Slow (200ms), complex	Rejected: Too slow
SSD	Balanced speed/accuracy	Lower mAP than YOLO	Rejected: Accuracy gap
EfficientDet	Good efficiency	Harder to deploy	Rejected: YOLO easier
YOLOv5	Stable, proven	Slightly slower than v8	Rejected: v8 improvements

1.11.6 References

- YOLOv8 Documentation
- YOLOv8 Benchmarks

1.11.7 Metadata

- **Author:** Computer Vision Engineer
- **Date:** 2025-10-18
- **Reviewers:** AI/ML Lead, Tech Lead
- **Related ADRs:** ADR-003 (PyTorch), ADR-015 (Jetson Xavier)

1.12 ADR-009: Use Docker for Deployment

1.12.1 Status

Accepted

1.12.2 Context

Deployment requires reproducible environments across dev, test, production machines. Options: Docker, conda, bare metal.

Requirements: - Reproducibility (same environment everywhere) - Isolation (dependencies don't conflict) - Easy rollback (version control) - CI/CD integration

1.12.3 Decision

Use Docker for containerized deployment with Docker Compose for multi-container orchestration.

1.12.4 Consequences

1.12.4.1 Positive

- **Reproducibility:** Dockerfile specifies exact environment
- **Isolation:** Each container has own dependencies
- **Portability:** Runs on any Docker-compatible host
- **CI/CD:** Easy to build, test, deploy in pipelines
- **Rollback:** Tag images, revert to previous version
- **Multi-Container:** Docker Compose for ROS2 + DB + monitoring

1.12.4.2 Negative

- **Overhead:** ~100MB per container, slight performance cost
- **Complexity:** Requires understanding Docker networking, volumes
- **GPU Access:** Requires nvidia-docker for GPU containers

1.12.4.3 Risks

- **Debugging:** Harder to debug inside containers
  - *Mitigation:* Remote debugging, VS Code devcontainers

1.12.5 Alternatives Considered

Alternative	Pros	Cons	Decision
Conda	Good for Python	No isolation for system libs	Rejected: Not comprehensive
Snap/Flatpak	OS-level packages	Complex for multi-service apps	Rejected: Overkill
Bare Metal	No overhead	Hard to reproduce	Rejected: Ops nightmare

1.12.6 References

- Docker Documentation
- NVIDIA Container Toolkit

1.12.7 Metadata

- **Author:** DevOps Engineer
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead
- **Related ADRs:** None

1.13 ADR-010: Adopt ELK Stack for Centralized Logging

1.13.1 Status

Accepted

1.13.2 Context

System generates logs from multiple nodes (vision, planning, control). Need centralized logging for search, analysis, debugging.

Requirements: - Full-text search (find logs by keyword) - Real-time ingestion (logs appear <1 sec) - Visualization (dashboards, charts) - Scalable (handle 10k logs/min)

1.13.3 Decision

Use ELK Stack (Elasticsearch, Logstash, Kibana) + Filebeat for log shipping.

1.13.4 Consequences

1.13.4.1 Positive

- **Search:** Elasticsearch provides powerful full-text search
- **Visualization:** Kibana dashboards for log analysis
- **Scalability:** Elasticsearch scales horizontally
- **Flexibility:** Logstash filters/transforms logs

- **Community:** Large ecosystem, many integrations

1.13.4.2 Negative

- **Resource Usage:** Elasticsearch requires 2GB+ RAM
- **Complexity:** 4-component stack (ES, Logstash, Kibana, Filebeat)
- **Cost:** If using Elastic Cloud (mitigated by self-hosting)

1.13.4.3 Risks

- **Performance:** Slow queries on large log volumes
  - *Mitigation:* Index retention policies (delete logs >30 days)

1.13.5 Alternatives Considered

Alternative	Pros	Cons	Decision
Splunk	Enterprise-grade	Expensive ($$$)	Rejected: Cost prohibitive
Graylog	Simpler than ELK	Smaller community	Rejected: Prefer ELK ecosystem
Loki (Grafana)	Lightweight, integrates with Grafana	Less mature	Considered, but ELK more proven
CloudWatch	Managed service	Vendor lock-in, cost	Rejected: On-prem requirement

1.13.6 References

- Elastic Stack
- Filebeat

1.13.7 Metadata

- **Author:** Site Reliability Engineer
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead, DevOps
- **Related ADRs:** ADR-011 (Grafana monitoring)

1.14 ADR-011: Use Grafana + Prometheus for Monitoring

1.14.1 Status

Accepted

1.14.2 Context

Need real-time monitoring of system metrics (CPU, RAM, cycle time, error rate). Options: Grafana, Datadog, New Relic.

Requirements: - Real-time dashboards (<1 sec latency) - Time-series data storage - Alerting (threshold-based) - Open-source

1.14.3 Decision

Use Grafana for visualization + **Prometheus** for metrics collection.

1.14.4 Consequences

1.14.4.1 Positive

- **Open-Source:** No licensing costs
- **Flexibility:** Highly customizable dashboards
- **Alerting:** Built-in alert manager
- **Ecosystem:** 100+ data sources (Prometheus, InfluxDB, PostgreSQL)
- **Community:** Large user base, many plugins

1.14.4.2 Negative

- **Setup:** Requires configuring exporters (node_exporter, custom)
- **Retention:** Prometheus default 15 days (need long-term storage for InfluxDB)

1.14.4.3 Risks

- **Scalability:** Prometheus single-node (not distributed)
  - *Mitigation:* Federation or Thanos for multi-cluster

1.14.5 Alternatives Considered

Alternative	Pros	Cons	Decision
Datadog	All-in-one, easy setup	$15/host/month	Rejected: Cost for small deployment
New Relic	APM + monitoring	Expensive, vendor lock-in	Rejected: Overkill
CloudWatch	AWS-native	Only works with AWS	Rejected: On-prem deployment

1.14.6 References

- Grafana Documentation
- Prometheus Documentation

1.14.7 Metadata

- **Author:** Site Reliability Engineer
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead
- **Related ADRs:** ADR-010 (ELK logging)

1.15 ADR-012: Implement OAuth2 + JWT for Authentication

1.15.1 Status

Accepted

1.15.2 Context

Web dashboard and API require user authentication. Options: Basic Auth, OAuth2, SAML, API keys.

Requirements: - Secure (encrypted tokens) - Stateless (no server-side sessions) - Role-based access control (RBAC) - Token expiry/refresh

1.15.3 Decision

Use OAuth2 with **JWT (JSON Web Tokens)** for authentication and authorization.

1.15.4 Consequences

1.15.4.1 Positive

- **Stateless:** JWT contains all user info (no DB lookup per request)
- **Scalable:** No session storage required
- **Standard:** OAuth2 widely adopted, many libraries
- **RBAC:** JWT claims include user roles
- **Security:** Tokens expire, refresh tokens for re-auth

1.15.4.2 Negative

- **Token Size:** JWT larger than session ID (~200 bytes)
- **Revocation:** Hard to revoke JWT before expiry
  - *Mitigated by short TTL (15 min) + refresh tokens*

1.15.4.3 Risks

- **Secret Leakage:** If JWT secret exposed, all tokens compromised
  - *Mitigation:* Rotate secrets regularly, store in secret manager

1.15.5 Alternatives Considered

Alternative	Pros	Cons	Decision
Basic Auth	Simple	Send password every request (insecure)	Rejected: Not secure
Session Cookies	Familiar	Requires server-side storage	Rejected: Not stateless
API Keys	Simple for bots	No user identity, hard to rotate	Rejected: Need user auth

Alternative	Pros	Cons	Decision
SAML	Enterprise SSO	Complex, overkill for small system	Rejected: Too complex

1.15.6 References

- OAuth2 Specification
- JWT.io

1.15.7 Metadata

- **Author:** Security Engineer
- **Date:** 2025-10-18
- **Reviewers:** Tech Lead, Backend Engineer
- **Related ADRs:** None

1.16 ADR-013: Choose Intel RealSense D435i Camera

1.16.1 Status

Accepted

1.16.2 Context

Vision system requires RGB-D camera for object detection and pose estimation. Options: RealSense D435i, Azure Kinect, Orbbec Astra.

Requirements: - RGB: 1920×1080 @ 30fps - Depth: Range 0.3-3m, accuracy ±2% - USB 3.0 interface - ROS2 driver available - <$500 cost

1.16.3 Decision

Use Intel RealSense D435i RGB-D camera.

1.16.4 Consequences

1.16.4.1 Positive

- **Performance:** 1920×1080 RGB @ 30fps, 1280×720 depth @ 30fps
- **Accuracy:** Depth error <2% at 1m
- **SDK:** librealsense2 library, ROS2 wrapper (realsense2_camera)
- **IMU:** Built-in IMU (accelerometer, gyroscope) for odometry
- **Cost:** $350 (vs $400 for Kinect)
- **Compact:** 90mm × 25mm × 25mm

1.16.4.2 Negative

- **Indoor Only:** Struggles with sunlight (active IR stereo)
- **Range:** 3m max (vs 5m for Kinect)

- **Support:** Intel discontinued RealSense division (but SDK still maintained)

1.16.4.3 Risks

- **Availability:** Potential supply chain issues (mitigated by stocking spares)

1.16.5 Alternatives Considered

Alternative	Pros	Cons	Decision
Azure Kinect	Better depth accuracy, 5m range	$400, larger (103mm), no ROS2 driver	Rejected: Cost, no ROS2 support
Orbbec Astra	Cheap ($150), similar specs	Lower quality, less support	Rejected: Quality concerns
Stereo Camera	Passive (works outdoors)	Requires calibration, slower	Rejected: Active IR easier

1.16.6 References

- RealSense D435i Datasheet
- realsense2_camera ROS2 Wrapper

1.16.7 Metadata

- **Author:** Hardware Engineer
- **Date:** 2025-10-18
- **Reviewers:** Computer Vision Engineer, Tech Lead
- **Related ADRs:** ADR-015 (Jetson Xavier for processing)

1.17 ADR-014: Select UR5e Robot Arm

1.17.1 Status

Accepted

1.17.2 Context

Need 6-DOF robot arm for pick-and-place. Options: UR5e, ABB IRB 1200, KUKA iiwa, Fanuc CRX.

Requirements: - Payload: 5kg - Reach: 850mm - Repeatability: ±0.1mm - Collaborative (safe near humans) - ROS2 driver available - <$40k cost

1.17.3 Decision

Use Universal Robots UR5e (collaborative robot).

1.17.4 Consequences

1.17.4.1 Positive

- **Specifications:** 5kg payload, 850mm reach, ±0.03mm repeatability
- **Collaborative:** ISO/TS 15066 compliant (force-limited)
- **ROS2 Support:** Excellent ROS2 driver (ur_robot_driver)
- **Ease of Use:** PolyScope pendant for manual teach
- **Ecosystem:** Large community, many grippers/accessories compatible
- **Reliability:** Proven in industry (10,000+ deployed)

1.17.4.2 Negative

- **Cost:** $35,000 (vs $25k for ABB IRB 1200)
  - *Justified by collaborative safety, better support*
- **Speed:** Max 1 m/s (vs 1.5 m/s for ABB)
  - *Acceptable for 30 picks/min target*

1.17.4.3 Risks

- **Vendor Dependence:** Locked into UR ecosystem for accessories
  - *Mitigation:* Use standard ISO flange, avoid proprietary interfaces

1.17.5 Alternatives Considered

Alternative	Pros	Cons	Decision
ABB IRB 1200	Cheaper ($25k), faster	Not collaborative, complex ROS2 setup	Rejected: Safety gap
KUKA iiwa	Best force sensing	$80k, overkill for task	Rejected: Cost
Fanuc CRX	Good specs, reliable	Weaker ROS2 support	Rejected: ROS2 ecosystem
Dobot Magician	Cheap ($1k), beginner-friendly	Low payload (0.5kg), poor accuracy	Rejected: Not industrial-grade

1.17.6 References

- UR5e Specifications
- ur_robot_driver (ROS2)

1.17.7 Metadata

- **Author:** Robotics Engineer
- **Date:** 2025-10-18
- **Reviewers:** Mechanical Engineer, Tech Lead
- **Related ADRs:** None

1.18 ADR-015: Use NVIDIA Jetson Xavier for Vision Processing

1.18.1 Status

Accepted

1.18.2 Context

Vision pipeline (YOLO, pose estimation) requires GPU for real-time inference. Options: Jetson Xavier, Jetson Orin, Cloud GPU, x86 + NVIDIA GPU.

Requirements: - GPU inference (<50ms for YOLOv8) - TensorRT support - <$1000 cost - Low power (<30W) - Compact form factor

1.18.3 Decision

Use NVIDIA Jetson Xavier NX (16GB variant).

1.18.4 Consequences

1.18.4.1 Positive

- **GPU:** 512 CUDA cores, 21 TOPS AI performance
- **TensorRT:** Native support, 3× speedup vs PyTorch
- **Memory:** 16GB shared RAM/GPU (enough for models + buffers)
- **Power:** 15W typical (vs 300W for desktop GPU)
- **Size:** 70mm × 45mm (credit card size)
- **Cost:** $500 (vs $1500 for Orin, $2000 for RTX 4060)
- **SDK:** JetPack (Ubuntu 20.04 + CUDA + TensorRT + ROS2)

1.18.4.2 Negative

- **Performance:** Slower than desktop RTX (but sufficient for 30fps)
- **Cooling:** Passive cooling only, can throttle under load
  - *Mitigation:* Add active fan ($10)*
- **Ubuntu 20.04:** Older than 22.04 (but ROS2 Humble can run on 20.04)

1.18.4.3 Risks

- **Supply:** Jetson supply constrained (COVID, chip shortage)
  - *Mitigation:* Order early, stock spares

1.18.5 Alternatives Considered

Alternative	Pros	Cons	Decision
Jetson Orin	2× faster (40 TOPS)	$1500, overkill	Rejected: Cost, not needed
x86 + RTX 4060	Powerful, upgradable	$2000, 300W, bulky	Rejected: Power/size

Alternative	Pros	Cons	Decision
Cloud GPU (AWS)	Scalable	Latency (50-100ms), recurring cost	Rejected: Latency
Jetson Nano	Cheap ($100)	Too slow (128 CUDA cores)	Rejected: Performance

1.18.6 References

- Jetson Xavier NX
- TensorRT Benchmarks

1.18.7 Metadata

- **Author:** Embedded Systems Engineer
- **Date:** 2025-10-18
- **Reviewers:** Computer Vision Engineer, Tech Lead
- **Related ADRs:** ADR-003 (PyTorch), ADR-008 (YOLOv8)

1.19 Summary Table: All ADRs

ADR	Category	Decision	Key Benefit	Key Risk
ADR-001	Middleware	ROS2 Humble	Real-time, LTS, security	Learning curve
ADR-002	Middleware	CycloneDDS	Low latency	Smaller community
ADR-003	AI/ML	PyTorch	Research-friendly, ONNX export	TF Lite more mature
ADR-004	Planning	MoveIt2	Comprehensive, OMPL	Complexity
ADR-005	Control	ros2_control	Standardization, portability	Debugging
ADR-006	Database	PostgreSQL	JSONB support, extensibility	Heavier than SQLite
ADR-007	Orchestration	BehaviorTree.CPP	Modularity, reactivity	Learning curve
ADR-008	Vision	YOLOv8	Speed, accuracy	AGPL license
ADR-009	Deployment	Docker	Reproducibility, isolation	Debugging
ADR-010	Logging	ELK Stack	Search, visualization	Resource usage
ADR-011	Monitoring	Grafana + Prometheus	Open-source, flexible	Setup complexity
ADR-012	Security	OAuth2 + JWT	Stateless, scalable	Token revocation
ADR-013	Hardware	RealSense D435i	Cost, ROS2 support	Indoor only
ADR-014	Hardware	UR5e Robot	Collaborative, ROS2 driver	Cost

ADR	Category	Decision	Key Benefit	Key Risk
ADR-015	Hardware	Jetson Xavier NX	GPU, TensorRT, compact	Supply chain

---

Document Status: Complete **Last Updated:** 2025-10-18 **Review Cycle:** Quarterly (update as decisions change) **Version Control:** Git (track changes via commits)