Robust Robot System: A Modular, Decentralized Python Framework

Tim

July 10, 2025

Contents

1 Introduction

This document describes the architecture, components, and usage of a lightweight, modular, and decentralized robotic communication system written entirely in Python. The system is designed to enable fast prototyping and testing of multi-node robot applications without requiring the full ROS 2 stack. It supports real-time image streaming, automatic network discovery, topic-based communication, and monitoring tools.

2 System Overview

The system is based on the following design principles:

- Decentralized publish/subscribe communication using ZeroMQ
- Lightweight modules with clean separation of concerns
- Support for automatic discovery of publishers over a local network
- Real-time image transmission using OpenCV
- Monitoring utilities for inspecting active topics
- Compatibility with virtual environments and packaging via `setup.py`

3 Installation

3.1 Create and Activate a Virtual Environment

```
python3 -m venv venv
source venv/bin/activate
```

3.2 Install the Project in Editable Mode

```
pip install -e .
```

4 Directory Structure

```
robust_robot_system/
|- core/
|  |- init.py
|  |- pubsub.py
|  |- discovery.py
|- nodes/
|  |- camera_node.py
|  |- auto_viewer_node.py
|- tools/
|  |- topic_monitor.py
|- launch.py
|- setup.py
```

5 Core Components

5.1 ZMQPubSub

Handles all TCP-based publish/subscribe logic using ZeroMQ.

```
class ZMQPubSub:
def init(self, role, port="5556", ip="localhost"):
# ... ZeroMQ PUB or SUB setup ...

def publish(self, topic, message):
    self.socket.send_string(f"{topic} {message}")

def listen(self, callback):
    # Starts background thread for receiving messages
```

5.2 Discovery Module

Provides UDP broadcast-based discovery of publisher IPs.

```
def broadcast_identity(port, topics):
# Sends UDP packets to 255.255.255.255 with IP, port, and topics

def discover_publisher():
# Listens on UDP port 9999 for broadcasted discovery packets
```

6 Nodes

6.1 camera_node.py

Streams JPEG-compressed images from the system camera over the network.

```
cap = cv2.VideoCapture(0)
while True:
ret, frame = cap.read()
_, buffer = cv2.imencode('.jpg', frame)
jpg_bytes = base64.b64encode(buffer).decode("utf-8")
pub.publish("camera/image_jpg", jpg_bytes)
```

6.2 auto_viewer_node.py

Automatically discovers a publisher and visualizes incoming images.

```
ip, port = discover_publisher()
sub = ZMQPubSub("sub", ip=ip, port=port)
sub.listen(on_msg)
```

7 Monitoring

7.1 Topic Monitor

Lists all topics recently seen over the network.

```
last_seen = {}
sub.listen(lambda topic, msg: last_seen.update({topic: time.time()}))
```

7.2 Usage

To launch the monitor:

```
python tools/topic_monitor.py
```

8 Multi-Device Usage

To use across multiple machines:

1. Ensure all devices are on the same local network
2. Start the `camera_node.py` on one machine
3. Start `auto_viewer_node.py` on the other machine
4. The system connects automatically via UDP discovery

9 Conclusion

This system demonstrates how to build a fully functional, modular robot communication architecture using modern Python tools. With ZeroMQ, UDP discovery, and OpenCV integration, it is suitable for lightweight robotic prototyping, especially in swarm or multi-agent settings.