RoboLang v1 Developer Guide — Building the Runtime Interpreter

Edition 1.0 | November 2025

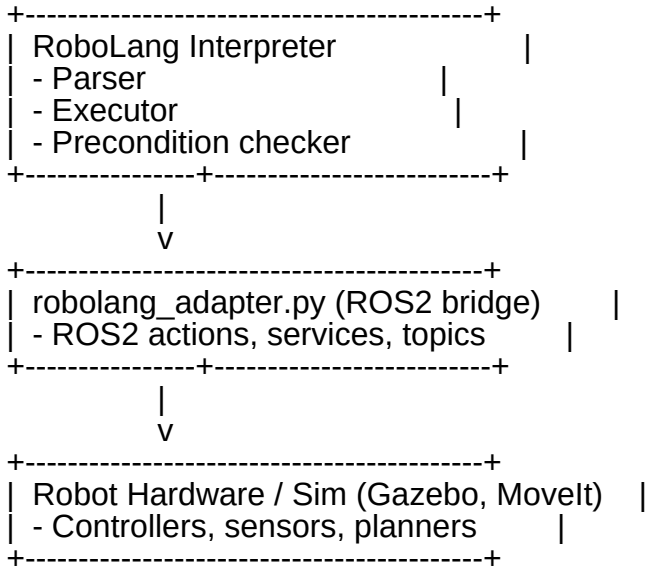
Overview

This guide describes how to implement the RoboLang Runtime Interpreter, responsible for executing RoboLang tasks using ROS2 actions and services through the Python adapter layer.

Contents

---

1. Architecture



---

2. Minimal Parser (v1)

Example Python code:

```
class RoboLangRuntime:
    def __init__(self, adapter):
        self.adapter = adapter
```

```
    def execute(self, rob_code: str):
        for line in rob_code.splitlines():
            line = line.strip()
            if not line or line.startswith("//"):
                continue
            print(f"[EXEC] {line}")
            self._dispatch(line)

    def _dispatch(self, line: str):
        if line.startswith("move"):
            self.adapter.move_to_pose(["joint_1"], [0.5])
        elif line.startswith("grasp"):
            self.adapter.set_gripper(1.0)
        elif line.startswith("place"):
            self.adapter.set_gripper(0.0)
        elif line.startswith("inspect"):
            self.adapter.inspect("object", "camera_top")
        elif line.startswith("wait for"):
            seconds = float(re.findall(r"\d+", line)[0])
            self.adapter.wait_for(seconds)
        elif line.startswith("communicate"):
            parts = re.findall(r'\"(.*?)\"', line)
            if len(parts) == 2:
                channel, msg = parts
                self.adapter.communicate(channel, msg)
```

---
3. Binding Runtime Variables

A context dictionary maps RoboLang symbols to real-world entities.

```
context = {
    "robot_arm_1": "robot_arm_1",
    "blue_box_1": "object_42",
    "shelf_A": [0.5, -0.8, 1.2, 0.1, 1.0, -0.2],
    "cell_A": "workcell_1"
}
```

---
4. Precondition Checking

```
def check_preconditions(adapter, pre_block: str):
    lines = [ln.strip() for ln in pre_block.splitlines() if ln.strip()]
    for ln in lines:
        if ln.startswith("region_clear"):
            region = ln.split()[-1]
            if not adapter.check_region_clear(region):
                raise RuntimeError(f"Region {region} not clear")
        if ln.startswith("sensor_ok"):
            sensor = re.findall(r'\"(.*?)\"', ln)[0]
            if not adapter.check_sensor_ok(sensor):
                raise RuntimeError(f"Sensor {sensor} not OK")
```

---
5. Executing RoboLang Tasks

Example task file:

```
task store_box(robot r, object box, location src, location dst, region cell) {
    pre {
        robot_ready r;
        region_clear cell;
    }
    plan {
        move r to src;
        grasp r box;
        move r to dst;
        place r box at dst;
        communicate r to "fleet" with "TASK_COMPLETE";
    }
}
```

Python runtime:

```
adapter = RobotAdapter("robot_arm_1")
runtime = RoboLangRuntime(adapter)
adapter.wait_for_servers()
runtime.execute(code)
```

---
6. Connecting to the LLM Planner

LLM -> RoboLang -> Runtime -> ROS2 -> Robot

Example FastAPI endpoint:

```
@app.post("/execute_robocode")
async def execute_task(payload: dict):
    code = payload.get("robocode", "")
    runtime.execute(code)
    return {"status": "ok"}
```

---
7. Deployment Options

Local ROS2 Workspace:
```
  colcon build
  ros2 run robolang runtime
```

Docker Container Example:

```
FROM ros:humble
RUN apt-get update && apt-get install -y python3-pip
COPY . /workspace
WORKDIR /workspace
RUN pip install -r requirements.txt
CMD ["python3", "main.py"]
```

---
8. Advanced Topics

- Task chaining

- Dynamic variable resolution
- Safety mode simulation
- Multi-robot scheduling

---
9. Checklist

■ Install adapter and verify ROS2 connections
■ Implement parser
■ Load and execute .rob tasks
■ Add precondition checks
■ Integrate with FastAPI
■ Containerize and deploy

---
10. Future Roadmap
---
v2: Full grammar parser (ANTLR)
v3: Concurrent task execution
v4: LLM-based optimization

---
11. Conclusion

RoboLang v1 ecosystem:
- DSL for readable, AI-generatable robotic tasks
- ROS2 Adapter bridge
- Runtime interpreter for execution

Together, they enable a unified, safe, and intelligent multi-robot system.