Duality AI's LLM Robotic Arm Challenge

1. Overview

Welcome to the Robotic Arm Control Challenge, a hackathon designed to help participants explore large language model (LLM) tools, embodied AI control, and real-time robotic manipulation using a simulated 6-degree-of-freedom robotic arm. This challenge involves development in a ROS2 node, which will interface with Duality's simulation software FalconSim in order to control the arm to complete tasks. Participants will design and refine the toolset an LLM uses to control the robotic arm, optimize the interaction between user prompts and system behavior, and develop advanced task sequences involving manipulation of various objects within a workspace.

Objectives

- Create a robust toolset that balances granularity vs abstraction to allow the LLM to complete all tasks without getting confused.
- Write clear tool descriptions to help the LLM understand capabilities.
- Prompt engineer the best prompt to help guide the LLM for which tools it should use.

- Test the design with complex multi-step tasks.

Importance of LLMs in Robotics

LLM-controlled robots are powerful because they can understand natural language, reason about tasks, and flexibly adapt their actions, but developing these capabilities safely and reliably requires simulation. By combining LLMs with simulated environments, researchers can teach robots to interpret instructions, plan complex behaviors, and generalize to new situations while avoiding the risks, costs, and time constraints of real-world testing. Simulation enables massive scale training, rapid iteration, and exposure to diverse scenarios that LLMs need for robust grounding between language and action. Together, LLM-driven control and simulation make it possible to build robots that are more capable, safer, and more predictable when finally deployed in the real world.

Hackathon Structure

Teams are given a set of challenges of increasing difficulty to use as guidance for developing the toolkit, which can be submitted and scored any number of times. There will be two tasks which will be used for part of the scoring. **The first attempt on these challenges will count towards your final overall score.** They can still be attempted again afterwards.

Development Overview

The core challenge consists of:

- **A ROS2 package at** /home/ubuntu/ros2_ws/src/ur3_controller
  - **ur3_controller_node.py** - Implements all the tools that the LLM has access to
  - **tool_definitions.py** - Contains descriptions of all tools
- **FalconSim** - The simulated environment where the challenge takes place

UR3 SDK

- The UR3 arm has the following primitive commands exposed:
  - Set inverse kinematics goal

    - Set forward kinematics goal
    - Set gripper state (grab/release)
- FalconSim exposes the following information about the environment
    - Dictionary of item names and their pose
    - UR3 end effector pose
    - UR3 joint angles
    - Name of item currently held by UR3
    - Boolean representing if UR3 is in currently moving

It is recommended to look through the ROS2 package to understand what format all the data is expected to be in. The implementation of the UR3 can also be explored. An example implementation of **move_to_position** and **move_to_object** are given.

i. Hackathon Tasks

1. **Tool Design & AI Engineering**
   Participants will:
   - Create new tool definitions, balancing granularity and reasoning efficiency.
   - Develop a prompting strategy for the LLM to achieve greatest clarity and efficiency with the given toolset

2. **Documentation & Presenting Final Team Results**
   The team should document:
    - Tool definitions and reasoning behind tool choices
    - Success and failure cases observed
    - Example prompts and outcomes
    - Developed strategy for prompting the LLM

   See the **Documentation & Presentation** section for details on format.

ii. Key Deliverables

A complete implementation including:

1. **Modified files**
    a. Modified **ur3_controller_node.py** and **defined_tools.py**
    b. Any other modified files, such as gemini_interface.py if the prompt is modified.
    c. Make sure to remove your gemini key for security reasons.
2. **Documentation**
    a. Documentation as described above in the Hackathon Tasks section.
    b. A list of final prompts to use for each task to use for evaluation

iii. Judging Criteria

Judges will run your toolkit and prompts on each task to calculate the final score. The prompts will be run 3 times and the average score will be used. This means your results need to be reproducible. **We will be using the default Gemini 2.5.**

Model Performance

- Each challenge is scored from 0-100 based on the accuracy of the result. The score is the percentage of points for that challenge awarded.
- Judges will use the given prompt with toolkit and any other modified files and run the model to determine the final score. This means that the prompt and toolkit must also be reliable and results must be reproducible.
- The score used will be the best of 3 runs to prevent one-offs.
- This also means that the same toolkit must be able to complete all the tasks.
- No cheesing the challenge such as using "Completes task 1" in the description or prompt, or just using a long and specific tool. **Doing so will result in a 0 for each challenge that uses this.**

iv. Bonus Task

In this bonus task, we will explore how using a better LLM changes results. Write a report comparing performance of Gemini 2.5 and Gemini 3.0.

- Include differences in actions observed for each challenge.

v. Score Breakdown

Challenge	**Points**
Assessment scenarios	20 pts (10 pts each)
Specified prompts	80 pts (10 pts each)
Gemini 3.0 comparison	10 pts
Total	110

---

2. Task Instructions

i. Toolkit Engineering:

This section will walk you through:

- ➔ Setting up your Falcon account
- ➔ Connecting to the VM
- ➔ Adding to the toolkit
- ➔ Running the scenario

1. **Create a Falcon Account:**
    a. Visit Falcon and sign up for an account if you don't have one
    b. Once registered, log in to access datasets, exercises, and tools.

2. **Create a Google Gemini API key**
    a. Follow Google's instructions on creating an API key https://ai.google.dev/gemini-api/docs/api-key
    b. This key will go in gemini_interface.py in the ROS2 node

3. **Connecting to the VM (VMs will be made active on the day of the Hackathon)**
    a. VM Connection Instructions

4. **Adding to the toolkit**
    a. Tools are defined in /home/ubuntu/ros2_ws/src/ur3_controller/ur3_controller/defined_tools.py
    b. Tools are implemented in /home/ubuntu/ros2_ws/src/ur3_controller/ur3_controller/ur3_controller_node.py

NOTE- It is recommended to look through the package to understand how it works. You may modify other files if you think it will improve performance.

5. **Running the scenario**
    a. To run the scenario with the ROS2 node, simply type **run_ur3** in the terminal and it will launch the scenario along with the ROS2 node. Make sure to kill the node and the scenario before launching again.

---

ii. Documentation & Presentation:

Ensure that your team's work is:

✔ Clear and understandable
✔ Reproducible by others
✔ Professional and impactful for judges

1. **Slide show where each slide corresponds to a scenario**

   a. Indicate scenario number
   b. Show a video (optional)
   c. Include LLM prompt
   d. List sequence of tools used by the arm
   e. Talk about problems faced with solutions such as added or removed tools associated with scenario

2. **Example Slide:**
   a. **Task:** Scenario 1
   b. **LLM Prompt:** Move the red cube to the bowl
   c. **Issue Faced:** Arm moved cube directly into the bowl, pushing it.
   d. **Solution:** Added a tool for moving above an object to drop it.

3. **Written report documenting tools and methodology**
   a. For each tool, write its name, used description, and a brief summary of why you decided to add it.
   b. Talk about any methodology used in prompting the LLM or knowing what tools needed to be added to overcome a scenario.
   c. **Clearly note any modified files** aside from ur3_controller.py and defined_tools.py, including the file location, as well as motivation for modifying them.

---

3. Submission and Final Steps Instructions

i. Necessary Submission Files:

- A single, **<u>Final Packaged Folder</u>** that includes all necessary files:
  - ☐ All modified files (ur3_controller_node.py, defined_tools.py, etc)
    OR

- [ ] Submit a zip of the ur3_controller package at /home/ubuntu/ros2_ws/src/ur3_controller/ur3_controller

- A well-structured PPT and PDF as specified in the Documentation & Presentation section.

ii. Upload Instructions:

1. Compress modified files with the PPT and PDF reports to zip.
2. Upload the zipped folder to a private GitHub repository of your own.
3. Complete the submission form (Here) by:
    a. Providing the GitHub repository link
4. Finally, add the following reviewers as collaborators:
    a. Syed Muhammad Maaz
        i. GitHub Username - Maazsyedm
    b. Rebekah Bogdanoff
        i. GitHub Username - rebekah-bogdanoff
    c. Evan Goldman
        i. GitHub Username - egold010

iii. After Submission:

1. Sharing Results & Feedback
    a. After submission, teams can showcase their models and insights.
    b. Feedback from judges will be provided after evaluation.

2. Future Opportunities & Improvements
    a. Continue exploring advanced topics such as:
        i. Using an MCP with an LLM

3. Stay connected with us via Discord for:
    a. Future challenges

b. Internship and apprenticeship opportunities
c. Community events and AI workshops

NOTE: If you face any issues along the process join our discord channel linked above where we will be providing support to you for the hackathon.

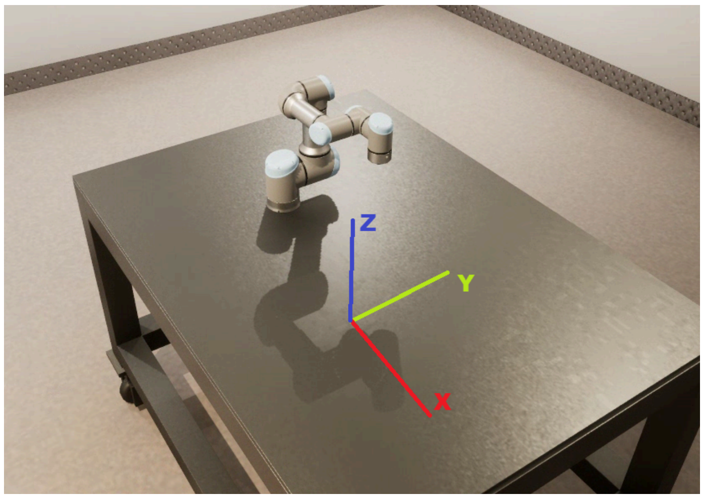
---

4. Useful Information

Starting Arm Transform:

0	0	1	30.62
-1	0	0	13.07
0	-1	0	35.81
0	0	0	1

Coordinate System:



Objects

Object	**Dimensions (cm)**	**Name**
Cube	10×10×10	‘[Color] cube’ e.g. ‘red cube’, ‘blue cube’ OR just ‘cube’ if no color specified

Sandwich Materials	20×20×2.4	‘bread’, ‘lettuce’, ‘tomato’, ‘cheese’, ‘turkey’, ‘plate’
Button	Cylinder 20×3	‘button’
Stove	30×30×8	‘stove’

Default LLM Prompt (in case you modify it):

```
prompt = f"""
    You are a planning module that controls a UR3 robotic arm by calling
tools.

    You receive:
    1. A natural language TASK.
    2. A set of available TOOLS with their names, descriptions, and
parameter schemas.

    Your job:
    - Break the TASK into a sequence of tool calls.
    - Each tool call must use only the tools defined in TOOLS.
    - Every tool call must include all required parameters, with values of
the correct type.

    TOOLS (JSON schema):
    {tools_json}

    TASK:
    \"\"\"{task_prompt}\"\"\"


    OUTPUT FORMAT (VERY IMPORTANT):
    Return ONLY a JSON array (no extra text, no explanations), where each
element is:

    {{
    "tool": "<tool name as in TOOLS.name>",
    "parameters": {{
        "<parameter_name_1>": <value>,
        "<parameter_name_2>": <value>,
        ...
    }}
    }}

    Rules:
    - Do NOT invent new tools. Use only tools listed in TOOLS.
    - For list parameters, return a JSON list with the correct number and
type of elements.
    - For float parameters, return JSON numbers (not strings).
    - Every required parameter for a chosen tool must be present.
    - Do NOT include comments or trailing commas.
    - Do NOT include any additional keys besides "tool" and "parameters".
    - The top-level value MUST be a JSON array.
    - If the task requires multiple steps, include multiple tool calls in
the order they should be executed.
    - If a single tool call is enough, return an array with just one
element.

    Now produce the JSON plan of tool calls for the TASK.
"""
```

4. Common Issues and Troubleshooting

FAQ's

1. **It's asking for a confirmation code when logging into VM**
    - If you select the third option, it will allow you to enter a recovery email. Enter joel.mondal@duality.ai and it will log in.

Support and Communication

1. **Join the official Duality Community Discord Server for:**
    a. Real-time help
    b. Announcements
    c. Live Q&A with organizers
    d. Invite Link Here

We appreciate your hard work and look forward to reviewing your project! Feel free to share your work on LinkedIn and show the world your real-world problem solving skills in action!

Tag DualityAI to:

- Celebrate your team's efforts and creativity.
- Get noticed by industry experts and recruiters

— The Duality AI Team