Robotics Lab

Update

Pito Salas - May 2024

Robotics Teaching Lab

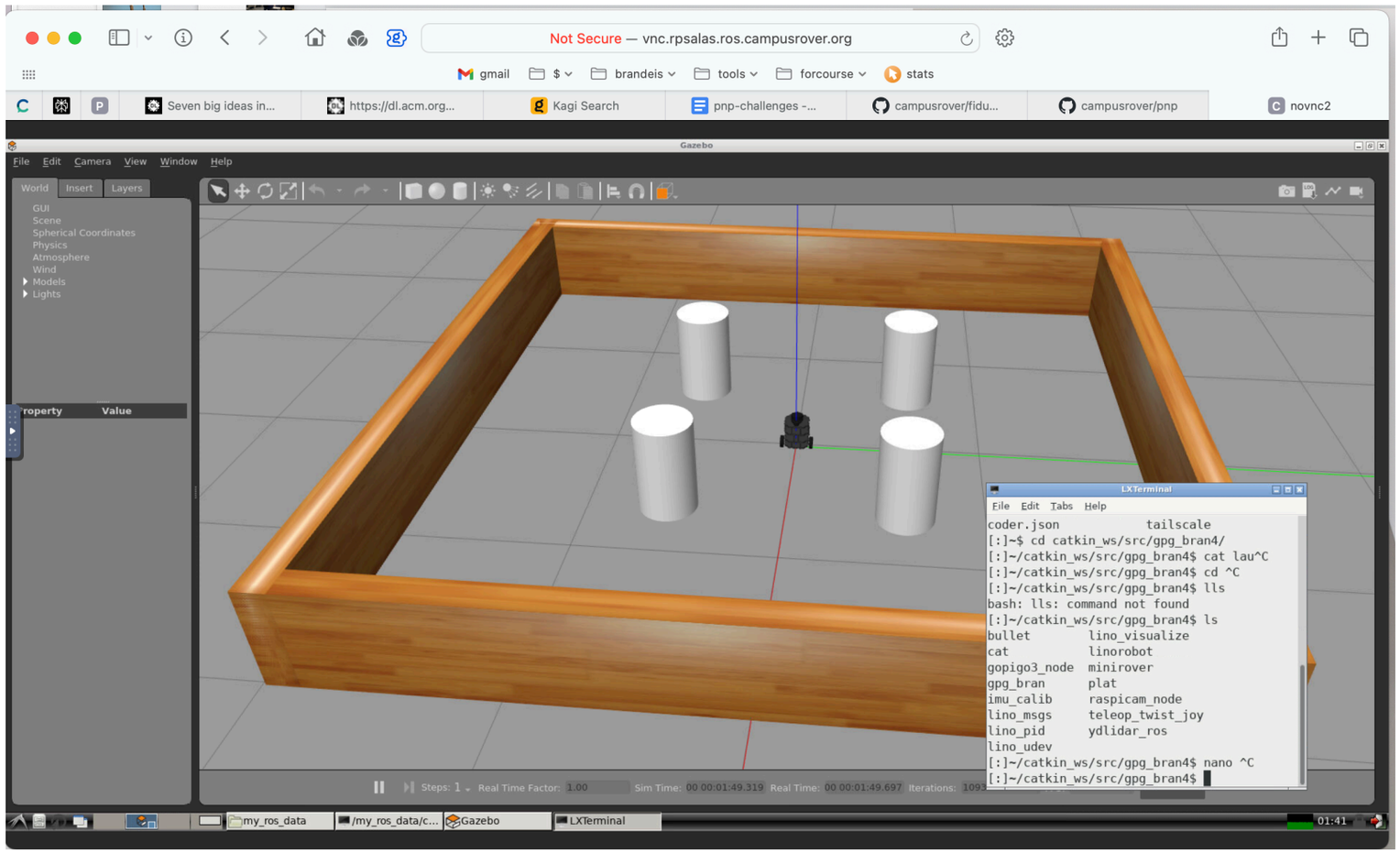
- Located in basement of Abelson-Bass
- Capacity around 20 students (packed)
- Used 24x7 by students during semester
- Structured Lab Times (sections)
- Independent Studies and Projects

Our Robots

- Robots
  - 7 Commercial Robotis Turtlebot3
  - 4 In-house designs
  - Allowing customization and teaching
- Typical architecture
  - Raspberry Pi (Linux)
  - Arduino (for real time)
  - Lidar
  - IMU
  - Motors and Controllers
  - Camera
  - Power

Cluster

- Small cluster (4 computers)
- Web based development environment for students
- Kubernetes + ROS
- For each student
  - Dedicated Linux instance
  - Desktop
  - ROS Stack installed
  - Web based VS Code
  - Easily recoverable
  - All in browser



Cosi119a: Autonomous Robotics

- Structured around "Seven Big Ideas" Paper (roughly)
- "How do robots..."
  - Know where they are in the world?
  - Move through the world?
  - See the world around them?
  - Know where to go?
  - Avoid running into things?
  - Control their bodies?
- Engineering oriented course
  - Core concepts
  - Theory
  - Programming assignments (sim and real)
  - Team based project

Seven Big Ideas in Robotics, and How To Teach Them

David S. Touretzky
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
dst@cs.cmu.edu

ABSTRACT

Robotics is widely recognized as an interdisciplinary mixture of engineering and computer science, but the latter component is not well represented at many undergraduate institutions. The sophisticated technologies that underlie perception, planning, and control mechanisms in modern robots need to be made accessible to more computer science undergraduates.

Following the curriculum design principles of Wiggins and McTighe (*Understanding by Design*, 2^{nd} Ed.), I present seven big ideas in robotics that can fit together in a one semester undergraduate course. Each is introduced with an essential question, such as *"How do robots see the world?"* The answers expose students to deep concepts in computer science in a context where they can be immediately demonstrated. Hands-on labs using the Tekkotsu open source software framework and robots costing under $1,000 facilitate mastery of these important ideas. Courses based on parts of an early version of this curriculum are being offered at Carnegie Mellon and several other universities.

and geometry and linear algebra [13]. But many computer science departments still aren't teaching serious robotics at the undergraduate level. Those that do offer a robotics elective are all too often teaching a high school curriculum where students assemble primitive robots from parts kits and program simple-minded reactive controllers. This paper is a call to action: we *can* make sophisticated robot technologies accessible to undergraduates. Doing so will help them develop into better computer scientists.

In this paper I present seven "big ideas" in robotics that can fit together in a one semester course for junior or senior level computer science majors. Following the *Understanding by Design* methodology of Wiggins and McTighe [17], each big idea is introduced with an essential question, such as *"How do robots see the world?"* The answers to these questions involve deep computer science concepts, some of which can only be fully mastered in graduate classes. But exposing undergraduates to these ideas – at an appropriate level – can broaden their understanding of computer science and encourage them to learn more. All of these ideas are implemented in the Tekkotsu software framework [11, 12] and can

Current Projects

Next generation in-house robot, “BranBot”, Spring 2024

- Collaboration between us and Maker Lab
- Charlie Squires and Tim Hebert

Robot Arm “Pick and Place” project, Spring 2024

- James Lee

BranBot

- 3d generation in-house Robot
- Based on our prototypes: “Platform” (it’s name)
  - We have four that have been thoroughly debugged
  - Each is a little different
- BranBot
  - Outdoor capable (big wheels and strong motor)
  - “Kitified” preparing for student builds
  - Low cost (~500)
  - Standard configuration, customizable
- Built together with Maker Lab
  - Charlie Squires (our hardware designer “in residence”)
  - Tim Hebert (Maker Lab)

Key Challenges

- Each previous robot was made a-la-carte
  - Basically prototypes
  - They are all the same, but a little different
- BranBot's design goals
  - Use the same basic electrical and electronic design
  - Standardize everything
  - Focus on robustness against daily use
  - Keep Cost Down
  - Plan to be able to document it as a kit

Pick and Place

- Robot Arm Control
- Creating new scaffolding for students
- One of the classic robotics and automation challenges
- Prepare for real world robotics
- “A general pick and place implementation using inexpensive robotic arms”

README

What

PNP can simulate any factory floor that can be described within the constraints of its JSON config file (`/config/config.json`).

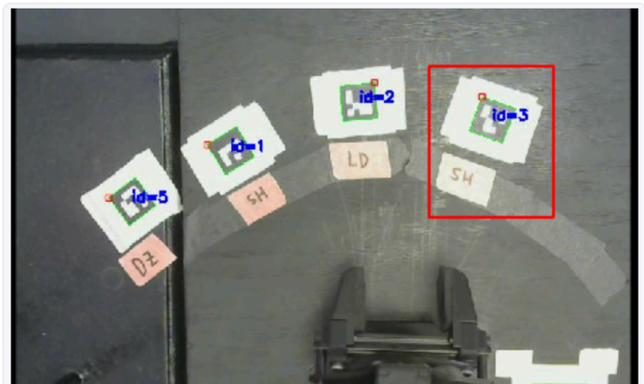
For instance, PNP ships with an example floor that has two tasks: washing that takes 3 seconds, and welding that takes 2 seconds.

```
"tasks": [
    {
        "id": "wash",
        "duration": 3
    },
    {
        "id": "weld",
        "duration": 2
    }
]
```

Tasks are executed at stations. The same task can be executed at more than one station. For example, we see below that washing happens at both `station_0` and `station_2` . This means that if a part needs to be washed, but `station_0` is occupied, it can be handled in `station_2` .

```
"stations": [
    {
        "id": "station_0",
        "fid_id": 1,
        "task_id": "wash",
        "free": true
    },
    {
        "id": "station_1",
        "fid_id": 2,
        "task_id": "weld",
        "free": true
    },
    {
        "id": "station_2",
        "fid_id": 3,
        "task_id": "wash",
        "free": true
    }
]
```

Thus, the mapping from tasks to stations can be one-to-many. Stations are represented by fiducial markers. We see above, for instance, that `station_2` is paired with a fiducial of id 3. The physical fiducial is marked in red below.



Arm

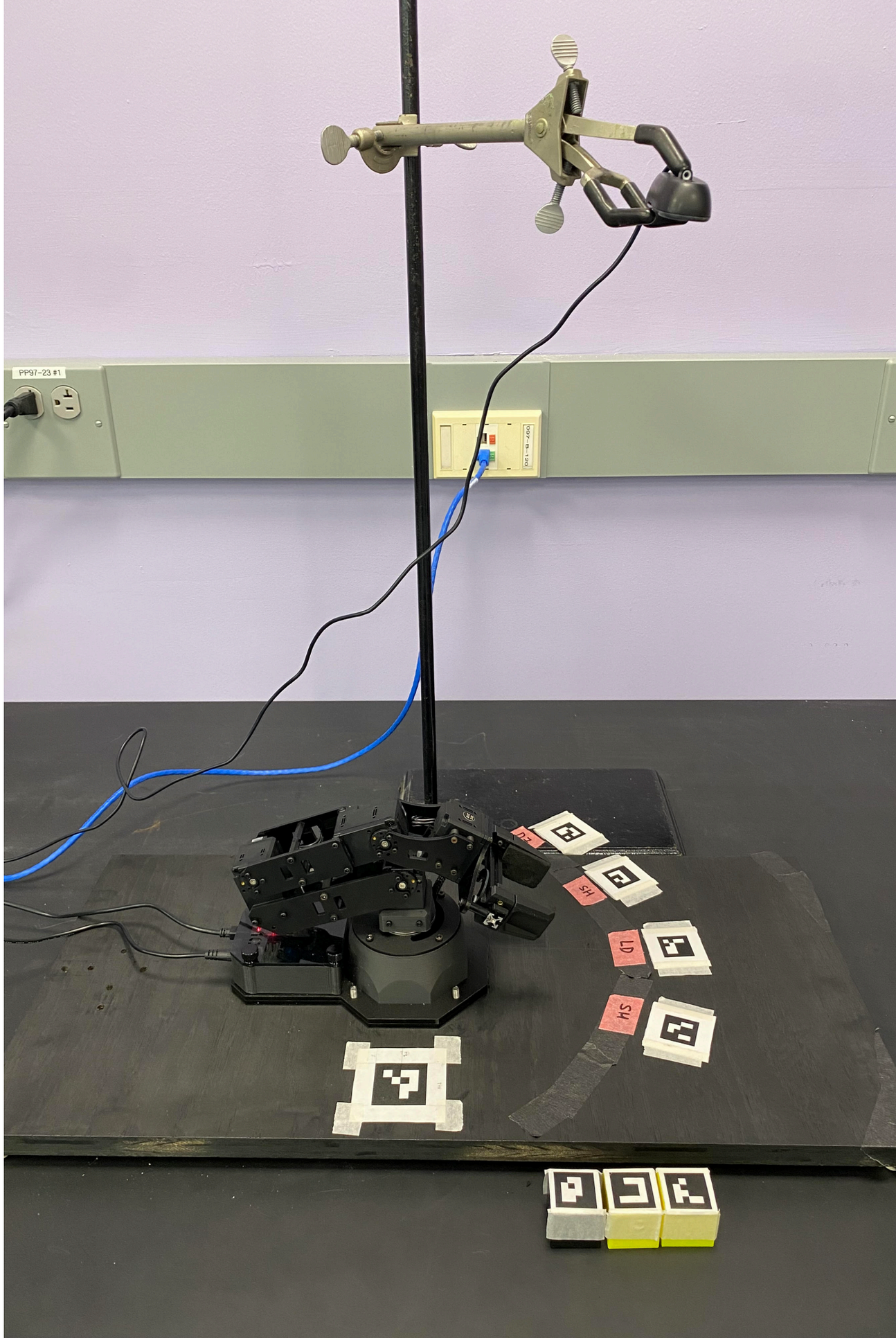
- The least expensive non-toy arm we could find
- Raspberry Pi for control
- Just four degrees of freedom (+ gripper)
- Note that gripper always points away from center of base
- That will be important later

Pick and Place

- Concept
  - Controlling a factory manufacturing process
  - Simplified model to pursue certain challenges
- Cargos, Stations and Tasks
  - Cargos: Work pieces in factory
  - CargoTypes: Types of cargos, requiring a sequence of tasks
  - Tasks: processes or operations to be done to cargos
- Quite flexible
  - But still simplistic
  - But exposes key challenges

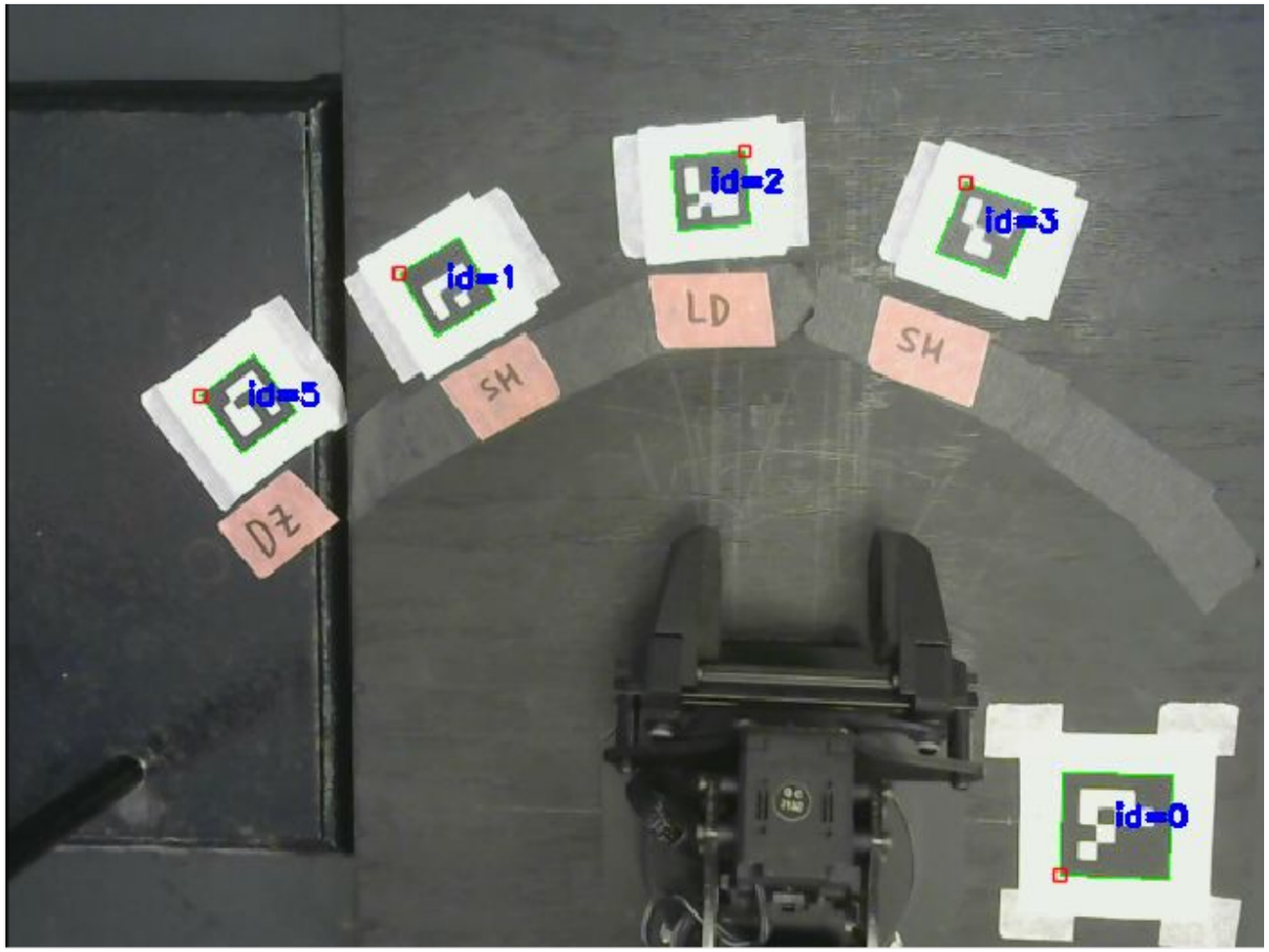
Experimental Setup

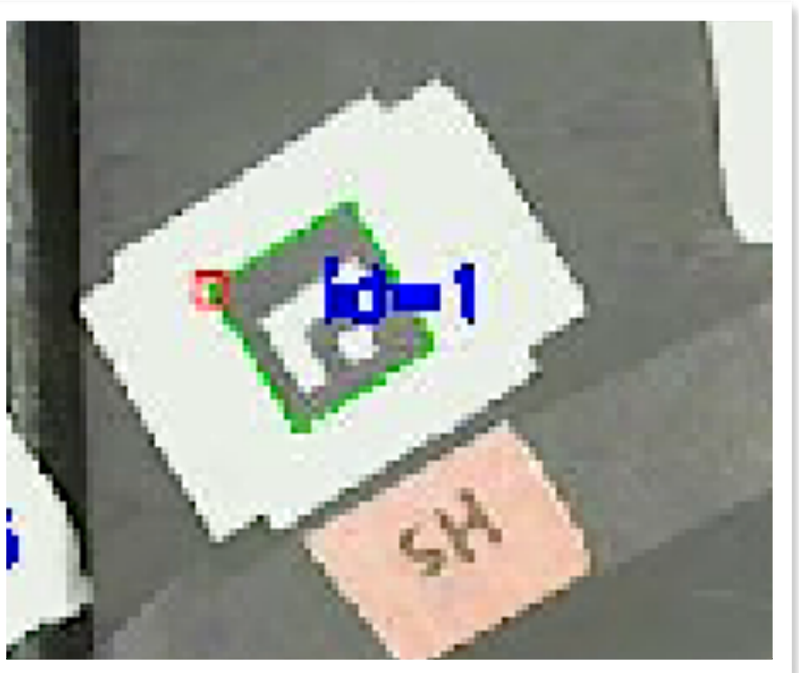
- Blocks at the bottom are the “cargos”
  - Representing work pieces
- Semicircle of fiducials
  - Representing 4 stations
- Single fiducial
  - Calibration of relative position of camera ↔ arm
- 3 blocks at bottom
  - Representing cargos
- Control computer out of view



Challenge #1

- Being robust around placement
  - Of stations
  - Payloads
  - Camera
- Approach:
  - Use of fiducials
    - Fixed to arm base
    - Indicating each station
    - Marking each payload
- Overhead view
  - CV identification
  - Green boxes, red square and id

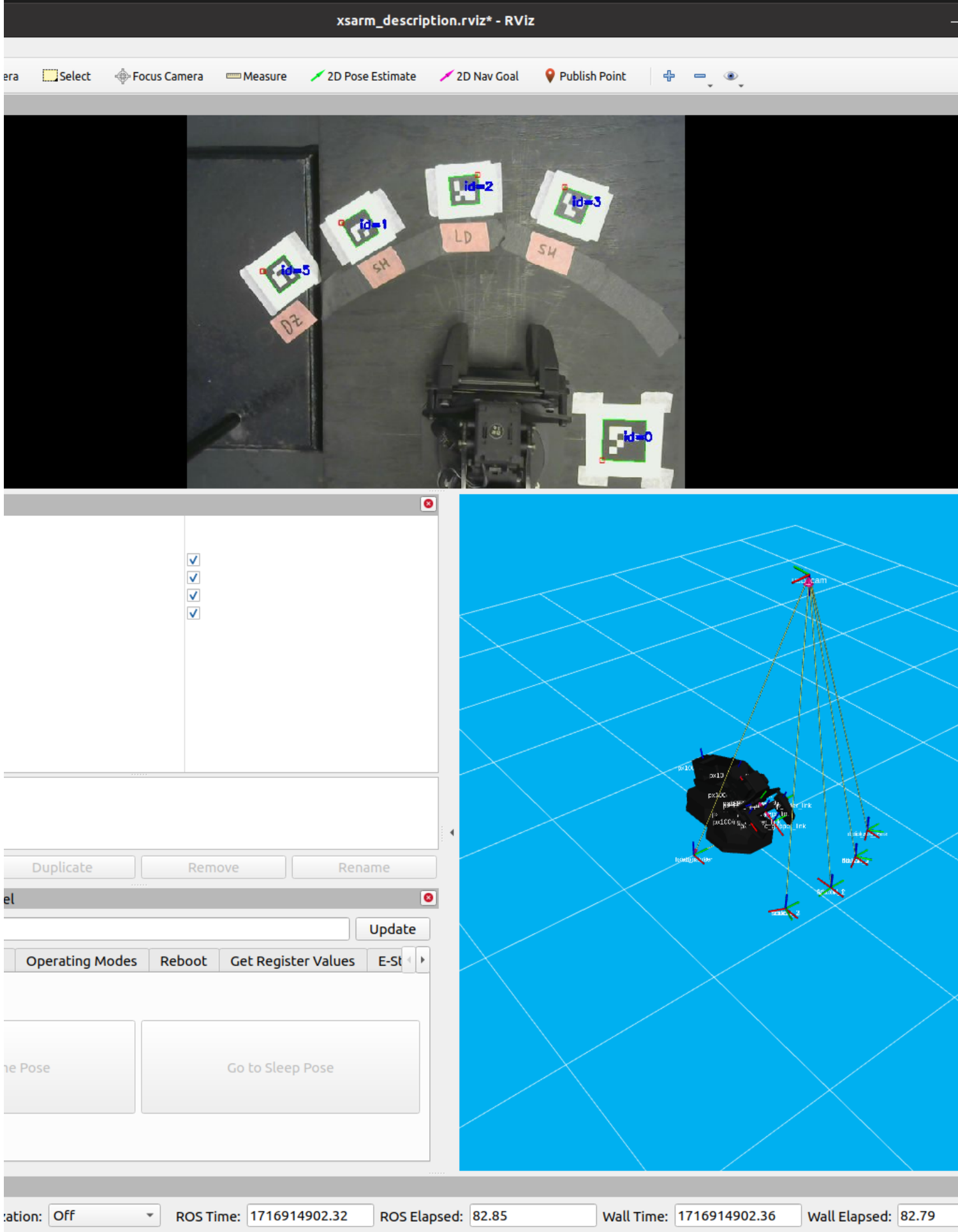




Challenge #2

- Controlling 4DOF Arm
- Move the gripper to the correct pose
  - Correct 3D location
  - Correct 3D orientation
- In general would require 6 DOF

- Solution
  - Stations and Cargos arranged in semi circle
  - Change trig calculations



Challenge #3

- Generalize scheduling plan
- Flexible and concurrent algorithm
- Driven by pnp-ml specification (still evolving)
  - *s* stations
  - t tasks
  - c cargo types
  - r cargos
- General
  - But constrained by arm


```
{
  "stations": [
    {
      "id": "station_0",
      "fid_id": 1,
      "task_id": "wash",
      "free": true
    },
    {
      "id": "station_1",
      "fid_id": 2,
      "task_id": "weld",
      "free": true
    },
    {
      "id": "station_2",
      "fid_id": 3,
      "task_id": "wash",
      "free": true
    }
  ],
  "tasks": [
    {
      "id": "wash",
      "duration": 3
    },
    {
      "id": "weld",
      "duration": 2
    }
  ],
  "cargoTypes": [
    {
      "id": "raw_parts",
      "tasks": ["wash", "weld", "wash"]
    },
    {
      "id": "washed_parts",
      "tasks": ["weld", "wash"]
    }
  ],
  "cargoes": [
    {
      "id": "cargo_0",
      "fid_id": 10,
      "type": "raw_parts",
      "cur_task_idx": -1,
      "last_station": "None",
      "last_placed": "Never"
    },
    {
      "id": "cargo_1",
      "fid_id": 11,
      "type": "raw_parts",
      "cur_task_idx": -1,
      "last_station": "None",
      "last_placed": "Never"
    },
    {
      "id": "cargo_2",
      "fid_id": 12,
      "type": "washed_parts",
      "cur_task_idx": -1,
      "last_station": "None",
      "last_placed": "Never"
    }
  ],
  "drop_off_zone_fiducial": "fiducial_5"
}
```

Challenge #4

- Turn it into student scaffolding
- Documented API
- Sample challenges
- Detailed instructions
- Github link

pnp Public

Edit Pins | Watch

main | 1 Branch | 0 Tags | Go to file t | Code

lj8128 End testing of github repo transfer. 6a9abdb · 5 days ago 9 Commits

config	Finalize source code of project.	last week
images	Add a README file, an images directory, and an i...	5 days ago
launch	Finalize source code of project.	last week
src	Finalize source code of project.	last week
CMakeLists.txt	Make first commit.	2 weeks ago
README.md	Finalize README	5 days ago
package.xml	Make first commit.	2 weeks ago

README

PNP

This project is called 'PNP' for 'pick and place'. It can simulate a factory floor, where parts are *picked up* and *placed down* during various stages of a manufacturing process.

Table of Contents

Video

Watch the project in action!

What

PNP can simulate any factory floor that can be described within the constraints of its JSON config file (`/config/config.json`).

Demo