CS498GC Mobile Robotics
Assignment 4 Part 1: Mobile Manipulator

Husky + UR3 + Robotis Gripper Integration

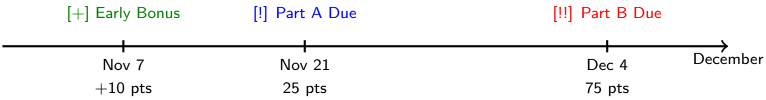
Kulbir Singh Ahluwalia

University of Illinois at Urbana-Champaign
Department of Computer Science

Fall 2025

Today's Demo Outline

Assignment Timeline



Part A (25 points)

- Due: November 21, 2025 @ 11:00 PM
- Early submission: Nov 7 (+10 pts)
- Rosbag + Screen recording

Part B (75 points)

- Due: December 4, 2025 @ 11:00 PM
- Navigation + Manipulation
- MoveIt2 integration

Gradescope

Entry Code: **KDP5G8**

Deliverable 1: Rosbag Recording

Requirements (12.5 points)

- Duration: **30 seconds minimum**
- Format: ROS 2 bag format
- Naming: `assignment4_[studentID].bag`
- Must capture **ALL** active topics

Required Topics

- `/cmd_vel` - Velocity commands
- `/odom` - Odometry (50 Hz)
- `/joint_states` - All joints
- `/imu` - IMU data (100 Hz)
- `/scan` - Laser (10 Hz)
- `/rh_p12_rn_position/command`

Recording Command

```
ros2 bag record -a -o assignment4_netid --max-bag-duration 30
```

Deliverable 2: Screen Recording

Requirements (12.5 points)

- Duration: **30 seconds**
- Resolution: **720p minimum**
- Format: MP4 or MOV
- Naming: `screen_recording_[studentID].mp4`

Timeline Breakdown

Mobile Base (0-10s)	Gripper (10-20s)	Combined (20-30s)

→ 30 sec

Submission Structure

```
assignment4_submission/
  |-- part_a/
      |-- assignment4_[studentID].bag/
      |-- screen_recording_[studentID].mp4
```

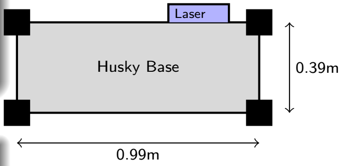
Husky Mobile Base Overview

Specifications

- Manufacturer: **Clearpath Robotics**
- Type: Differential drive
- DOF: 6 in SE(3)
- Mass: 50.0 kg
- Control: `/cmd_vel`

Dimensions

- Size: 0.99m $\times$ 0.67m $\times$ 0.39m
- Wheelbase: 0.57m
- Wheel radius: 0.165m
- 4-wheel configuration



Husky ROS 2 Integration

Control Interface

```xml
<plugin filename="gz-sim-diff-drive-system">
  <left_joint>rear_left_wheel_joint</left_joint>
  <right_joint>rear_right_wheel_joint</right_joint>
  <wheel_separation>0.57</wheel_separation>
  <wheel_radius>0.165</wheel_radius>
  <odom_publish_frequency>50</odom_publish_frequency>
  <topic>cmd_vel</topic>
  <frame_id>odom</frame_id>
  <child_frame_id>base_link</child_frame_id>
</plugin>
```

Topics

- `/cmd_vel` (Twist)
- `/odom` (Odometry @ 50Hz)
- `/tf` (Transforms)

Frames

- Parent: `odom`
- Child: `base_link`
- Wheels: continuous joints

Husky Sensor Suite - IMU

IMU Sensor

- Location: (0, 0, 0.35) from base
- Update rate: **100 Hz**
- Type: Gazebo IMU sensor
- Topic: `/imu`
- Frame: `imu_link`

IMU Data Published

- Linear acceleration (m/s^2)
- Angular velocity (rad/s)
- Orientation (quaternion)

Husky Sensor Suite - Laser Scanner

Laser Scanner

- Location: (0.4, 0, 0.15) from base
- Type: GPU LiDAR
- Range: 0.1m to 30.0m
- Samples: 720
- FOV: $\pm 90^\circ$ (π radians)
- Update rate: **10 Hz**
- Topic: `/scan`

Key Points

Both sensors publish at high frequency for real-time navigation and localization

Laser Scanner URDF Configuration

```xml
<sensor name="laser" type="gpu_lidar">
  <update_rate>10</update_rate>
  <topic>scan</topic>
  <ray>
    <scan><horizontal>
      <samples>720</samples>
      <min_angle>-1.570796</min_angle>  <!-- -90 degrees -->
      <max_angle>1.570796</max_angle>   <!-- +90 degrees -->
    </horizontal></scan>
    <range><min>0.1</min><max>30.0</max></range>
  </ray>
</sensor>
```

Note

GPU LiDAR provides efficient simulation of laser scanning with 720 samples per scan

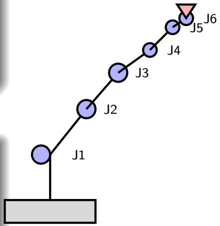
UR3 Manipulator Overview

Specifications

- Manufacturer: **Universal Robots**
- Model: UR3 collaborative robot
- DOF: **6 revolute joints**
- Total reach: 0.54m
- Mounting: (0.2, 0, 0.3) on Husky

Joint Configuration

- Joint 1: Shoulder Pan (Z-axis)
- Joint 2: Shoulder Lift (Y-axis)
- Joint 3: Elbow (Y-axis)
- Joint 4: Wrist 1 (Y-axis)
- Joint 5: Wrist 2 (Z-axis)
- Joint 6: Wrist 3 (Y-axis)



UR3 Joint Specifications - Joints 1-3

Joint	Range	Velocity	Effort	Link
Joint 1	$\pm\pi$ rad	3.15 rad/s	56 Nm	0.10m
Joint 2	$\pm\pi$ rad	3.15 rad/s	56 Nm	0.24m
Joint 3	$\pm\pi$ rad	3.15 rad/s	28 Nm	0.21m

Link Masses (Joints 1-3)

- Links 1-2: 2.0 kg (base, shoulder)
- Link 3: 1.5 kg (elbow)

UR3 Joint Specifications - Joints 4-6

Joint	Range	Velocity	Effort	Link
Joint 4	$\pm\pi$ rad	3.20 rad/s	12 Nm	0.08m
Joint 5	$\pm\pi$ rad	3.20 rad/s	12 Nm	0.08m
Joint 6	$\pm\pi$ rad	3.20 rad/s	12 Nm	0.05m

Link Masses (Joints 4-6)

- Links 4-6: 0.5-0.2 kg (wrist assembly)

ROS 2 Control

- Plugin: `gz-sim-joint-position-controller-system`
- Control mode: Joint position control
- Topic: `/joint_states`

UR3 URDF Configuration

```xml
<!-- UR3 Mount to Husky Base -->
<joint name="ur3_mount_joint" type="fixed">
  <parent link="base_link"/>
  <child link="ur3_mount"/>
  <origin xyz="0.2 0 0.3" rpy="0 0 0"/>
</joint>

<!-- Example: UR3 Joint 1 -->
<joint name="ur3_joint1" type="revolute">
  <parent link="ur3_mount"/>
  <child link="ur3_link1"/>
  <axis xyz="0 0 1"/>
  <limit lower="-3.14159" upper="3.14159"
         velocity="3.15" effort="56"/>
</joint>
```

GitHub Resources

- Official: `github.com/UniversalRobots/Universal_Robots_ROS2_Driver`
- MoveIt2 configs available for Humble/Jazzy

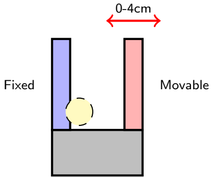
Robotis Gripper Configuration

Specifications

- Type: **Robotis RH-P12-RN Gripper**
- DOF: 1 (revolute joint)
- Base: 0.05m $\times$ 0.08m $\times$ 0.04m
- Fingers: Dual finger design
- Attachment: UR3 link 6

Joint Limits

- Joint: `rh_p12_rn_joint` (revolute)
- Range: 0.0 rad to 1.05 rad
- 0.0 rad = Fully open
- 1.05 rad = Fully closed (60°)
- Velocity: 0.5 rad/s
- Effort: 15 Nm



Gripper Control Commands

ROS 2 Control Interface

Topic: `/rh_p12_rn_position/command`
Message Type: `std_msgs/msg/Float64`

Open Gripper

```
ros2 topic pub -1 \
  /rh_p12_rn_position/command \
  std_msgs/msg/Float64 "{data: 0.0}"
```

Close Gripper

```
ros2 topic pub -1 \
  /rh_p12_rn_position/command \
  std_msgs/msg/Float64 "{data: 1.05}"
```

Gripper Automated Controller

C++ Implementation

Located in: `src/gripper_controller.cpp`

```cpp
// Create publisher for gripper control
publisher_ = this->create_publisher<std_msgs::msg::Float64>(
    "/rh_p12_rn_position/command", 10);

// Timer callback: Toggle every 5 seconds
auto message = std_msgs::msg::Float64();
message.data = is_open_ ? 1.05 : 0.0;  // Toggle state
publisher_->publish(message);
is_open_ = !is_open_;
```

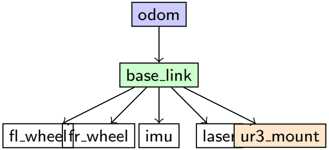
Note

This code runs in a timer callback that fires every 5 seconds, automatically opening and closing the gripper for demonstration purposes.

Base Transforms

odom (world frame)
base_link (Husky)

- front_left_wheel
- front_right_wheel
- rear_left_wheel
- rear_right_wheel
- imu_link (0, 0, 0.35)
- laser_link (0.4, 0, 0.15)
- ur3_mount (0.2, 0, 0.3)



Transform Hierarchy - Part 2: Manipulator

UR3 Arm Transforms

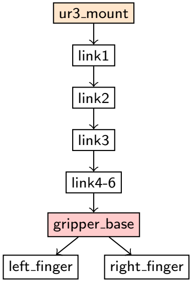
ur3_mount → UR3 Chain:

- ur3_link1 (shoulder pan)
- ur3_link2 (shoulder lift)
- ur3_link3 (elbow)
- ur3_link4 (wrist 1)
- ur3_link5 (wrist 2)
- ur3_link6 (wrist 3)

Gripper Transforms

ur3_link6 → Gripper:

- gripper_base (0, 0.085, 0.05)
- gripper_left_finger (fixed)
- gripper_right_finger (revolute)



Critical Transform Definitions

```xml
<!-- UR3 Mount to Base -->
<joint name="ur3_mount_joint" type="fixed">
  <parent link="base_link"/>
  <child link="ur3_mount"/>
  <origin xyz="0.2 0 0.3" rpy="0 0 0"/>
</joint>

<!-- Gripper to UR3 Wrist -->
<joint name="gripper_base_joint" type="fixed">
  <parent link="ur3_link6"/>
  <child link="gripper_base"/>
  <origin xyz="0 0.085 0.05" rpy="0 0 0"/>
</joint>

<!-- Example Wheel Joint -->
<joint name="rear_left_wheel_joint" type="continuous">
  <parent link="base_link"/>
  <child link="rear_left_wheel"/>
  <origin xyz="-0.256 0.285 0" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
</joint>
```

Gazebo Plugin Configuration

Differential Drive Plugin

```xml
<plugin filename="gz-sim-diff-drive-system">
  <left_joint>rear_left_wheel_joint</left_joint>
  <right_joint>rear_right_wheel_joint</right_joint>
  <wheel_separation>0.57</wheel_separation>
  <wheel_radius>0.165</wheel_radius>
  <odom_publish_frequency>50</odom_publish_frequency>
</plugin>
```

Joint Position Controller

```xml
<plugin filename="gz-sim-joint-position-controller-system">
  <joint_name>ur3_joint1</joint_name>
  <joint_name>ur3_joint2</joint_name>
  <!-- ... joints 3-6 ... -->
  <joint_name>gripper_joint</joint_name>
</plugin>
```

Verifying Transforms

Useful Commands

```
# Generate TF tree visualization
ros2 run tf2_tools view_frames
evince frames.pdf

# Check specific transform
ros2 run tf2_ros tf2_echo base_link ur3_link6

# Monitor all transforms
ros2 topic echo /tf

# Visualize in RViz
rviz2 -d $(ros2 pkg prefix --share husky_ur3_simulation)/rviz/view_robot.rviz
```

Common Issues

- Missing transforms: Check joint definitions
- Wrong frame IDs: Verify parent/child links
- Static vs dynamic transforms: Use appropriate publisher

Main Launch File - Part 1: Setup

```python
def generate_launch_description():
    # 1. Declare launch arguments
    use_sim_time = LaunchConfiguration('use_sim_time')
    world = LaunchConfiguration('world')

    # 2. Process URDF with xacro
    robot_description_content = Command([
        FindExecutable(name='xacro'), ' ', urdf_file
    ])

    # 3. CRITICAL FIX: Wrap in ParameterValue
    robot_description = {
        'robot_description': ParameterValue(
            robot_description_content, value_type=str
        )
    }
```

Main Launch File - Part 2: Nodes & Launch

```python
    # 4. Define nodes
    robot_state_publisher = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        parameters=[robot_description,
                    {'use_sim_time': use_sim_time}]
    )

    gazebo = ExecuteProcess(
        cmd=['gz', 'sim', '-v', '4', '-r', world],
        output='screen'
    )

    # 5. Return LaunchDescription
    return LaunchDescription([
        declare_use_sim_time, declare_world,
        robot_state_publisher, gazebo, spawn_robot, bridge
    ])
```

Key Launch Components

Robot State Publisher

```
robot_state_publisher = Node(
    package='robot_state_publisher',
    executable='robot_state_publisher',
    parameters=[robot_description,
                {'use_sim_time': use_sim_time}]
)
```

Gazebo Launch

```
gazebo = ExecuteProcess(
    cmd=['gz', 'sim', '-v', '4', '-r', world],
    output='screen'
)
```

Robot Spawner

```
spawn_robot = Node(
    package='ros_gz_sim',
    executable='create',
```

Bridge Node Setup

```
bridge = Node(
    package='ros_gz_bridge',
    executable='parameter_bridge',
    name='ros_gz_bridge',
    arguments=[
        # Format: topic@ros_type@gz_type
        '/cmd_vel@geometry_msgs/msg/Twist@gz.msgs.Twist',
        '/odom@nav_msgs/msg/Odometry@gz.msgs.Odometry',
        '/joint_states@sensor_msgs/msg/JointState@gz.msgs.Model',
        '/imu@sensor_msgs/msg/Imu@gz.msgs.IMU',
        '/scan@sensor_msgs/msg/LaserScan@gz.msgs.LaserScan',
        '/clock@rosgraph_msgs/msg/Clock@gz.msgs.Clock'
    ]
)
```

ROS-Gazebo Bridge - Topic Mapping

Bridge Format

`topic@ros_type@gz_type`

Control Topics

- `/cmd_vel` - Velocity commands
- `/joint_states` - Joint positions

System Topics

- `/clock` - Simulation time
- `/tf` - Transforms

Sensor Topics

- `/odom` - Odometry (50 Hz)
- `/imu` - IMU data (100 Hz)
- `/scan` - Laser scan (10 Hz)

Important

Bridge must be running for ROS 2 to communicate with Gazebo

Teleoperation Launch

```python
def generate_launch_description():
    return LaunchDescription([
        # Teleop keyboard for mobile base
        Node(
            package='teleop_twist_keyboard',
            executable='teleop_twist_keyboard',
            name='teleop_twist_keyboard',
            prefix='xterm -e',  # Opens in new terminal
            output='screen',
            remappings=[('/cmd_vel', '/cmd_vel')]
        ),

        # Gripper controller
        Node(
            package='husky_ur3_simulation',
            executable='gripper_controller',
            name='gripper_controller',
            output='screen'
        )
    ])
```

Launch Commands

```bash
# Full simulation
ros2 launch husky_ur3_simulation gazebo_sim.launch.py

# Teleoperation (separate terminal)
ros2 launch husky_ur3_simulation teleop.launch.py
```

Master Launch Script - Environment Setup

```bash
#!/bin/bash
# master_launch.sh - Single-command simulation startup

# Check and deactivate conda if active
if [ ! -z "$CONDA_PREFIX" ]; then
    echo "Deactivating conda..."
    conda deactivate 2>/dev/null || true
fi

# GPU configuration for NVIDIA
export __NV_PRIME_RENDER_OFFLOAD=1
export __GLX_VENDOR_LIBRARY_NAME=nvidia

# Source ROS 2 environment
source /opt/ros/jazzy/setup.bash
source ~/ros2_ws/install/setup.bash
```

Master Launch Script - Launch Process

```bash
# Launch simulation in background
./launch_gazebo_qt_fixed.sh headless &
GAZEBO_PID=$!

echo "Waiting for simulation to initialize..."

# Wait for initialization
for i in {1..10}; do
    if ros2 topic list 2>/dev/null | grep -q "/clock"; then
        echo "    Simulation started successfully!"
        break
    fi
    echo "  Checking... ($i/10)"
    sleep 1
done
```

Master Launch Script - Verification

```bash
# Verify all critical topics are available
echo "Checking for required topics..."

TOPICS=("/cmd_vel" "/odom" "/joint_states" "/imu" "/scan")
for topic in "${TOPICS[@]}"; do
    if ros2 topic list | grep -q "$topic"; then
        echo "    $topic"
    else
        echo "    $topic (missing)"
    fi
done

echo "Launch complete!"
```

GPU Fix Environment Variables

Essential GPU Environment Variables

```
# NVIDIA GPU Offloading
export __NV_PRIME_RENDER_OFFLOAD=1
export __GLX_VENDOR_LIBRARY_NAME=nvidia

# Qt5 OpenGL Fix
export QT_XCB_GL_INTEGRATION=xcb_glx

# Software Rendering Fallback
export LIBGL_ALWAYS_SOFTWARE=true  # Only if GPU fails
```

Launch Modes

```
# Headless (most stable)
./launch_gazebo_qt_fixed.sh headless

# With GUI
gz sim -v 4 -r world.sdf
```

Tip

Use headless mode for recording to avoid Qt issues

Terminal Workflow (4 Terminals)

Terminal 1: Simulation

```
./master_launch.sh
```

Terminal 2: Teleoperation

```
ros2 run teleop_twist_keyboard
```

Terminal 3: Recording

```
./record_assignment.sh
```

Terminal 4: Monitoring

```
ros2 topic hz /odom
```

Key Commands

- **Open Gripper**: `ros2 topic pub -1 /gripper/cmd [data: 0.0]`
- **Close Gripper**: `ros2 topic pub -1 /gripper/cmd [data: 1.05]`
- **Check Topics**: `ros2 topic list | grep odom`

Debugging Commands - Part 1: Topics & Nodes

Check Topics

```
# List all topics
ros2 topic list

# Check topic rate
ros2 topic hz /odom
ros2 topic hz /imu

# Echo topic data
ros2 topic echo /joint_states
```

Check Nodes

```
# List active nodes
ros2 node list

# Node info
ros2 node info \
  /robot_state_publisher

# Check parameters
ros2 param list
```

Quick Tip

Use `ros2 topic list -v` to see topic types alongside names

Debugging Commands - Part 2: Transforms & Gazebo

TF Debugging

```
# Check specific transform
ros2 run tf2_ros tf2_echo \
  base_link ur3_link6

# Visualize TF tree
ros2 run tf2_tools view_frames
evince frames.pdf

# Monitor all TFs
ros2 run tf2_ros tf2_monitor
```

Gazebo Debugging

```
# List Gazebo topics
gz topic -l

# Monitor model state
gz topic -e -t \
  /model/husky_ur3

# Check simulation time
gz topic -e -t /clock
```

Common Issue

If transforms missing: Check robot_state_publisher is running

Error Handling - Qt5/OpenGL Issues

Qt5/OpenGL Errors

```
# Error: "Cannot create platform OpenGL context"

# Solution:
export QT_XCB_GL_INTEGRATION=xcb_glx
export LIBGL_ALWAYS_SOFTWARE=false
export QT_QPA_PLATFORM=xcb
```

Quick Fix for Most Issues

1. Deactivate conda environment
2. Source ROS 2 workspace
3. Rebuild if necessary

Error Handling - Missing Topics

Missing ROS Topics

```
# Check if bridge is running
ros2 node list | grep bridge

# Restart bridge with required topics
ros2 run ros_gz_bridge parameter_bridge \
  /cmd_vel@geometry_msgs/msg/Twist@gz.msgs.Twist \
  /odom@nav_msgs/msg/Odometry@gz.msgs.Odometry
```

Verify Topics

```
# List all available topics
ros2 topic list

# Check specific topic frequency
ros2 topic hz /odom
```

Error Handling - Transform Errors

Transform Chain Issues

```
# Error: "Transform from X to Y does not exist"

# Check TF tree
ros2 run tf2_tools view_frames

# Monitor all transforms
ros2 run tf2_ros tf2_monitor

# Verify robot_state_publisher
ros2 node info /robot_state_publisher
```

Common Causes

- Robot state publisher not running
- URDF not loaded correctly
- Missing joint states

Error Handling - GPU Configuration

NVIDIA GPU Setup

```
# Verify GPU is detected
nvidia-smi

# Enable GPU offloading
export __NV_PRIME_RENDER_OFFLOAD=1
export __GLX_VENDOR_LIBRARY_NAME=nvidia
```

Software Rendering Fallback

```
# If GPU issues persist, use software rendering
export LIBGL_ALWAYS_SOFTWARE=true
export MESA_GL_VERSION_OVERRIDE=4.5
```

Automated Recording Script - Part 1: Setup

```bash
#!/bin/bash
# record_assignment.sh

# 1. Check simulation is running
if ! ros2 topic list | grep -q "/clock"; then
    echo "Error: Simulation not running!"
    exit 1
fi

# 2. Verify required topics
REQUIRED_TOPICS=("/imu" "/odom" "/joint_states" "/scan" "/cmd_vel")
for topic in "${REQUIRED_TOPICS[@]}"; do
    if ros2 topic list | grep -q "$topic"; then
        echo "    $topic"
    else
        echo "    $topic (missing)"
    fi
done
```

Automated Recording Script - Part 2: Recording

```
# 3. Countdown timer
for i in {10..1}; do
    echo -ne "\rStarting in: $i seconds... "
    sleep 1
done

# 4. Start recording with 30-second timeout
ros2 bag record -a \
    -o assignment4_netid \
    --max-bag-duration 30 &
RECORD_PID=$!

# 5. Show progress indicator
for i in {1..30}; do
    echo -ne "\rRecording: $i/30 seconds... "
    sleep 1
done
```

Automated Recording Script - Part 3: Verification

```bash
# 6. Wait for recording process to complete
wait $RECORD_PID

# 7. Verify recording was successful
if [ -d "assignment4_netid" ]; then
    echo "Recording completed successfully!"

    # Display bag info
    ros2 bag info assignment4_netid

    # Check file size
    du -h assignment4_netid/
else
    echo "Error: Recording failed!"
    exit 1
fi

echo "    Ready for submission"
```

Demo Recordings & Documentation I

Video Demonstrations

- **Main Demo Recording:**
  `https://tinyurl.com/cs498gc-demo1`
  (Full demonstration with audio commentary)
- **Navigation Demo:**
  `https://tinyurl.com/cs498gc-demo2`
  (Husky + UR3 with local/global navigation)

Documentation

- **Slide Deck PDF:**
  `https://tinyurl.com/cs498gc-slides`
- **Course Website:**
  `https://kulbir-singh-ahluwalia.com/cs498gc/fa25/`

Note

Original links available in Campuswire Post #122

Automated Launch Script - Core Components

One-Command Setup

Automates launching of 8 terminal windows

Part A Components (Required)

1. Gazebo simulation (Husky + UR3 + Gripper)
2. Robot state publisher & controllers
3. Teleoperation (keyboard control)
4. Gripper controller
5. ROS 2 bag recording (30 seconds)

Part B Components (Advanced)

6. MoveIt2 setup (motion planning)
7. Navigation stack (Nav2)
8. Topic monitoring & verification

Automated Launch Script - Usage

Quick Start

```
cd ~/ros2_ws/assignment4_scripts/
chmod +x launch_assignment4.sh
./launch_assignment4.sh
```

Key Features

- Auto-deactivates Conda conflicts
- Sources ROS 2 environment automatically
- Color-coded terminal output for easy tracking
- Built-in error checking and recovery
- Configurable for Humble or Jazzy

Note

Script opens multiple terminals - ensure you have gnome-terminal installed

GitHub Resources I

Main Repository

https://github.com/kulbir-ahluwalia/husky_ur3_simulator

Required Packages

- **Husky**: github.com/husky/husky
- **UR ROS2**: github.com/UniversalRobots/UR_ROS2_Driver
- **Robotis**: github.com/ROBOTIS-GIT/DynamixelSDK
- **Gazebo**: github.com/ros-controls/gz_ros2_control

Installation Guide

See assign4-part1-helpful-links.md for full URLs

Quick Reference - Commands & Topics

Key Commands

- `./master_launch.sh`
- `ros2 bag record -a`
- `ros2 topic hz /odom`
- `ros2 run tf2_tools view_frames`
- `ros2 topic list -v`

Important Topics

- `/cmd_vel` - Robot control
- `/odom` - Odometry @ 50 Hz
- `/imu` - IMU data @ 100 Hz
- `/scan` - Laser @ 10 Hz
- `/joint_states` - All joints
- `/clock` - Sim time

Pro Tip

Use `ros2 bag record -a --max-bag-duration 30` for automatic 30-second recording

Quick Reference - Troubleshooting & Submission

Common Issues

- Qt5 errors $\rightarrow$ Set GPU env vars
- Missing topics $\rightarrow$ Check bridge
- Transform errors $\rightarrow$ View TF tree
- Conda conflicts $\rightarrow$ Deactivate
- Slow simulation $\rightarrow$ Use headless mode

Assignment Deliverables

- **Part A:** Due Nov 21
- 30-second rosbag file
- 30-second screen recording
- Submit via Gradescope
- Entry Code: **KDP5G8**

Early Submission Bonus

Submit by November 7 for +10 extra credit points!

Resources and Support

Documentation

- **ROS 2**: docs.ros.org/jazzy
- **Gazebo**: gazebosim.org
- **Course**: kulbir-singh-ahluwalia.com/cs498gc

Office Hours

- Wed 1:30-2:30 PM, SC 4407
- kulbir@illinois.edu

Key Files

- `husky_ur3_gz.urdf.xacro`
- `gazebo_sim.launch.py`
- `master_launch.sh`
- `helpful-links.md`

Questions?

Good luck with Assignment 4!

Remember: Early submission by Nov 7 $= +10$ points

[GitHub] GitHub Resources Available
[Docs] See helpful-links.md for package URLs
[?] Post on Campuswire for help