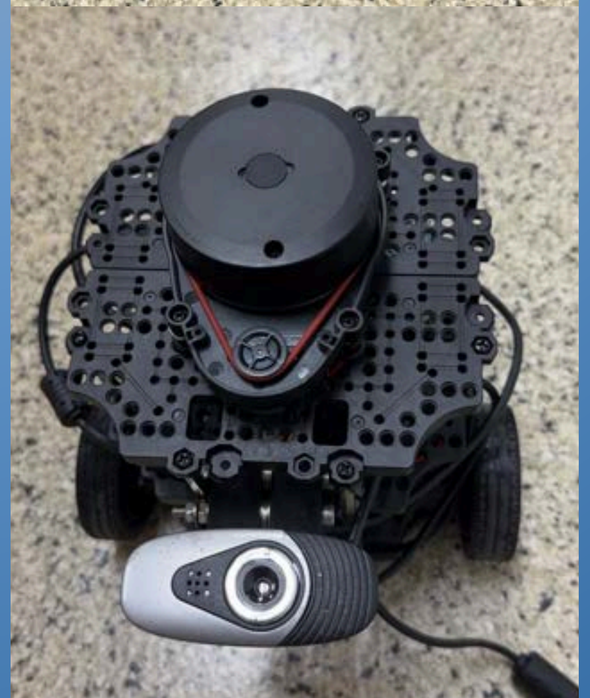
ROBOT NAVIGATION

WITH HAND GESTURE

Presented By: Nara Cheyklin 6538097021
Pannawish Leechasan 6538114021
Pasin Chanaphan 6538121321
Krittin Kitjaruwannakkul 6538007521



MOTIVATION

- Mobile robots increasingly operate in human-shared environments
- Traditional keyboard/joystick control is unintuitive
- Goal: combine autonomous navigation with natural human interaction using hand gestures

HARDWARE ARCHITECTURE

- TURTLEBOT3 BURGER
- OPENCR CONTROL BOARD
- RASPBERRY PI 4 (ON-BOARD COMPUTER)
- LDS-02 LIDAR
- DYNAMIXEL XL430-W250 MOTORS
- USB WEB CAMERA

SYSTEM CONCEPT & ARCHITECTURE

CORE IDEA

- User selects navigation goals visually in RViz
- Hand gestures control when and how the robot executes them

DISTRIBUTED ARCHITECTURE

- **TurtleBot3**: sensing, SLAM, navigation
- **External PC**: vision-heavy gesture recognition
- **ROS 2**: lightweight, real-time communication between components

SOFTWARE STACK OVERVIEW

01 OPERATING ENVIRONMENT

- Ubuntu Linux on both Robot & PC
- ROS 2 Humble (DDS-based middleware)

02 MAIN SOFTWARE MODULES

- SLAM: Cartographer
- Navigation: Nav2 Stack
- Perception: MediaPipe Hands
- Custom ROS 2 Python nodes

03 DESIGN PRINCIPLE

- Modular, scalable, and human-in-the-loop

NAVIGATION SYSTEM (SLAM & NAV2)

SLAM (CARTOGRAPHER)

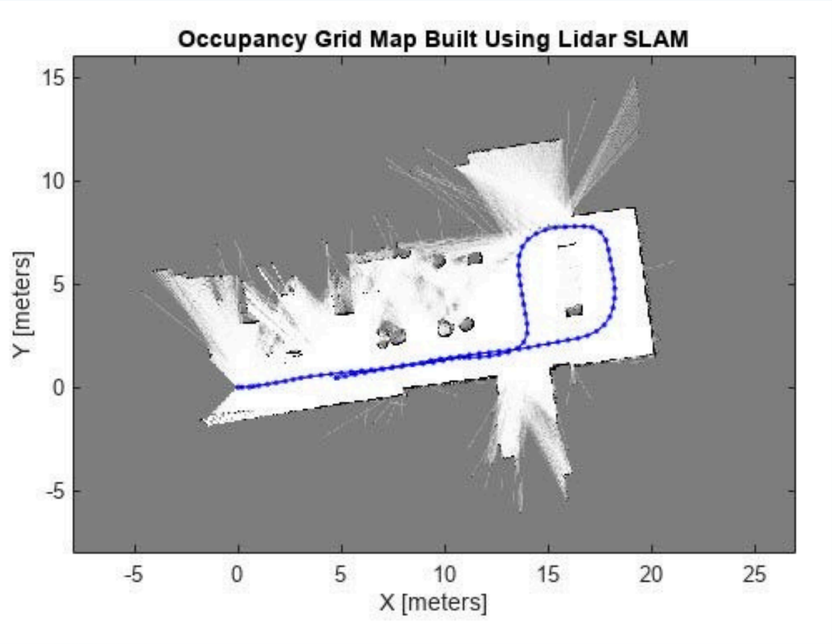
- Fuses LiDAR scans + wheel odometry
- Real-time scan matching and optimization
- Generates 2D occupancy grid map

LOCALIZATION & PLANNING (NAV2)

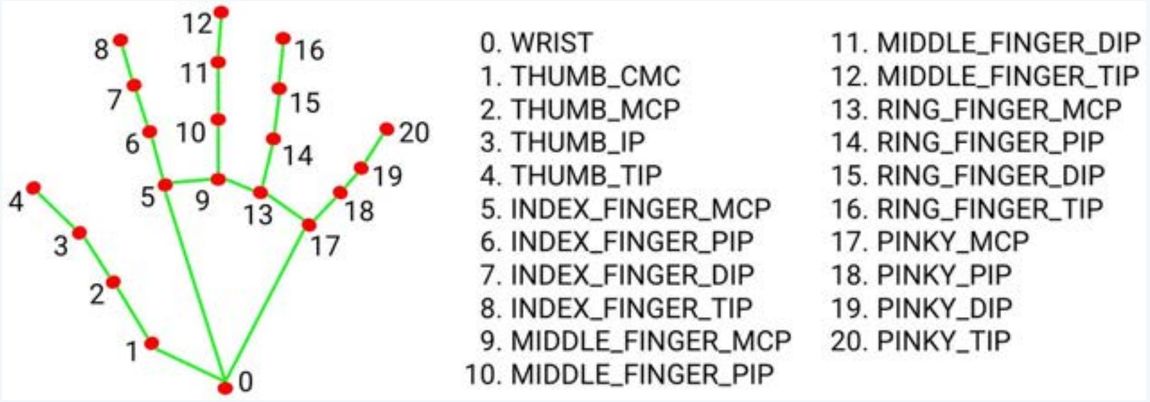
- AMCL estimates robot pose on saved map
- Global planner computes optimal path
- Local planner avoids dynamic obstacles
- Layered costmaps ensure safety

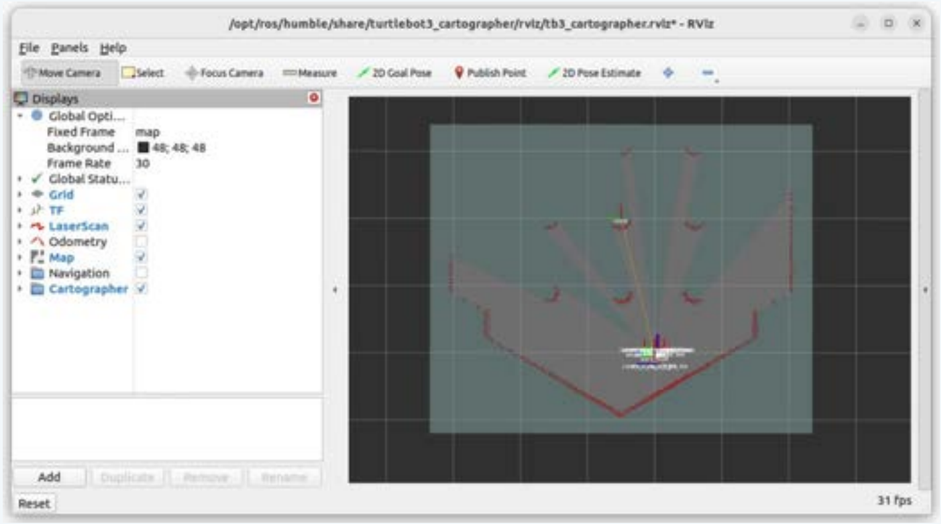
WAYPOINT EXECUTION

- User selects waypoints in RViz
- Custom node sends sequential goals to Nav2 Action Server



MEDIAPIPE GESTURE REGOGNITION





MediaPipe Hands Framework

- Detects 21 hand landmarks (wrist, MCP, PIP, DIP, fingertip)
- Provides 2D normalized coordinates for each joint
- Robust to hand orientation and scale

Gesture Interpretation

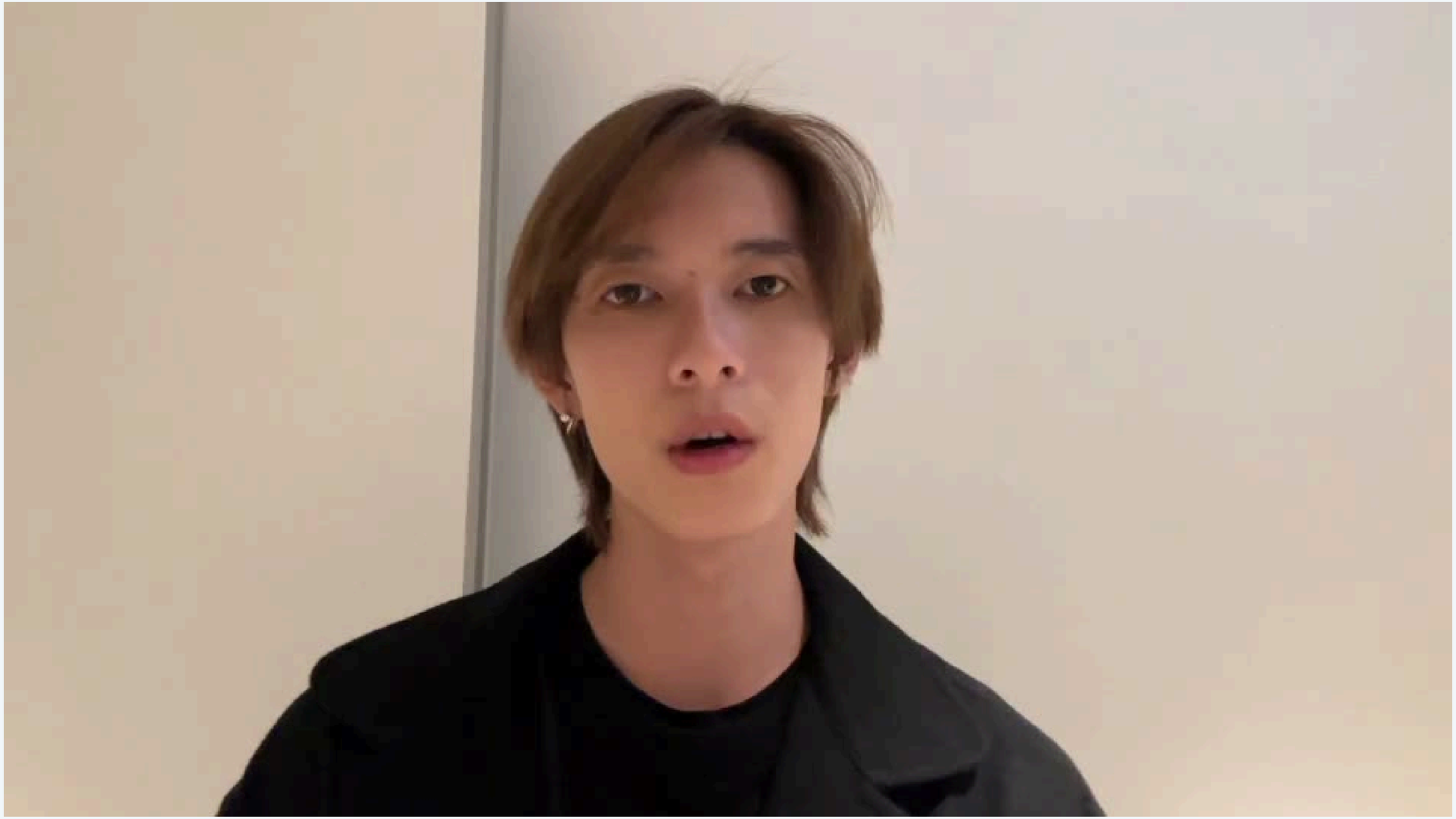
- Custom finger-counting logic based on landmark geometry
- Commands mapped to navigation control:
  - 1 finger → Start navigation
  - 2 fingers → Pause current goal
  - 4 fingers → Continue navigation
  - 5 fingers → Stop & cancel goals

ROS Integration

- Camera → /image_raw/compressed
- Gesture node publishes /gesture_command
- Navigation node reacts in real time

HAND_GESTURE_NODE

MULTIPOINT_NAV_NODE



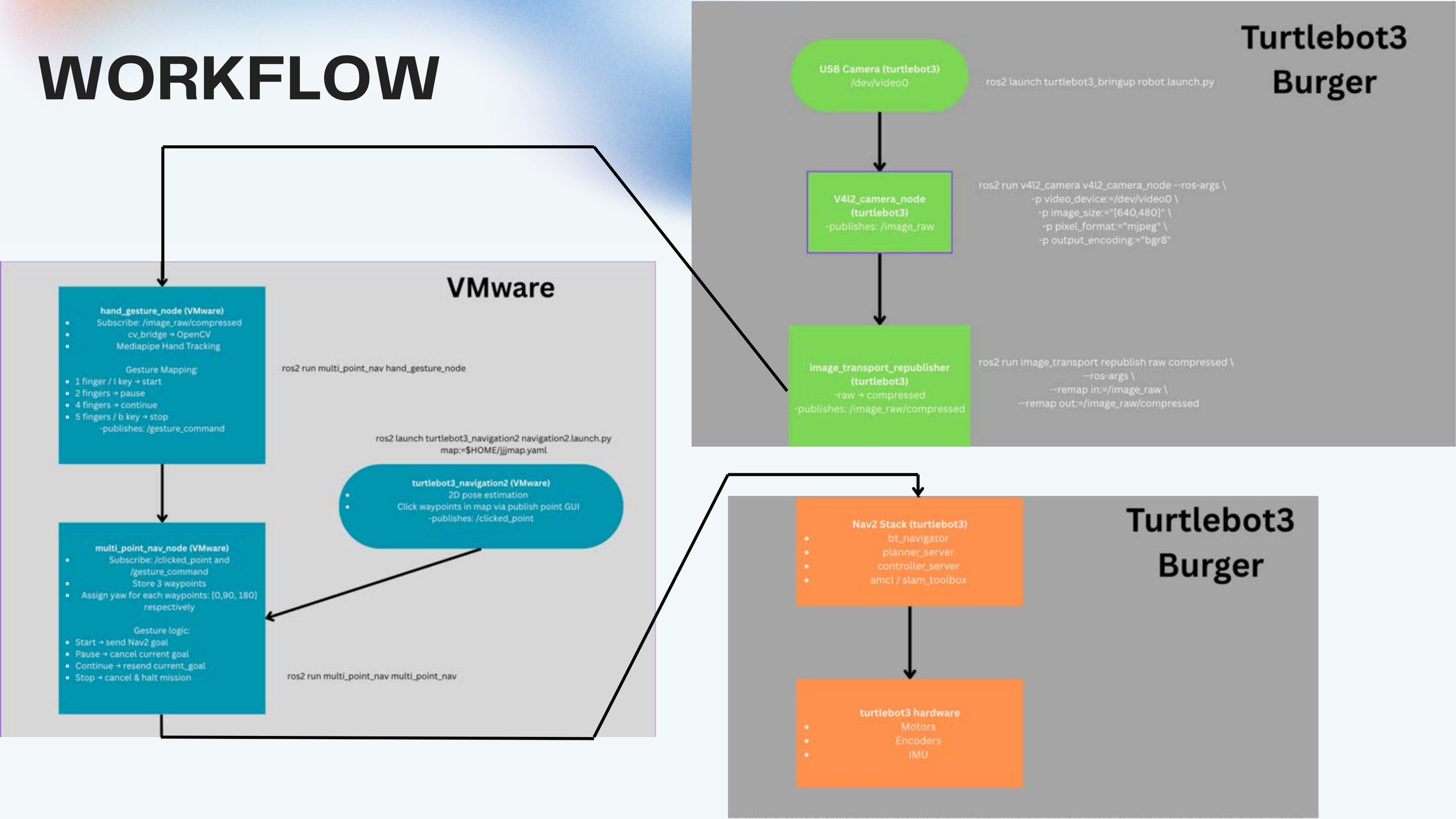
LAUNCH FILE FOR VMWARE

navigation2.launch.py +
multi_point_nav_node
+hand_gesture_node
= l**aunch file**

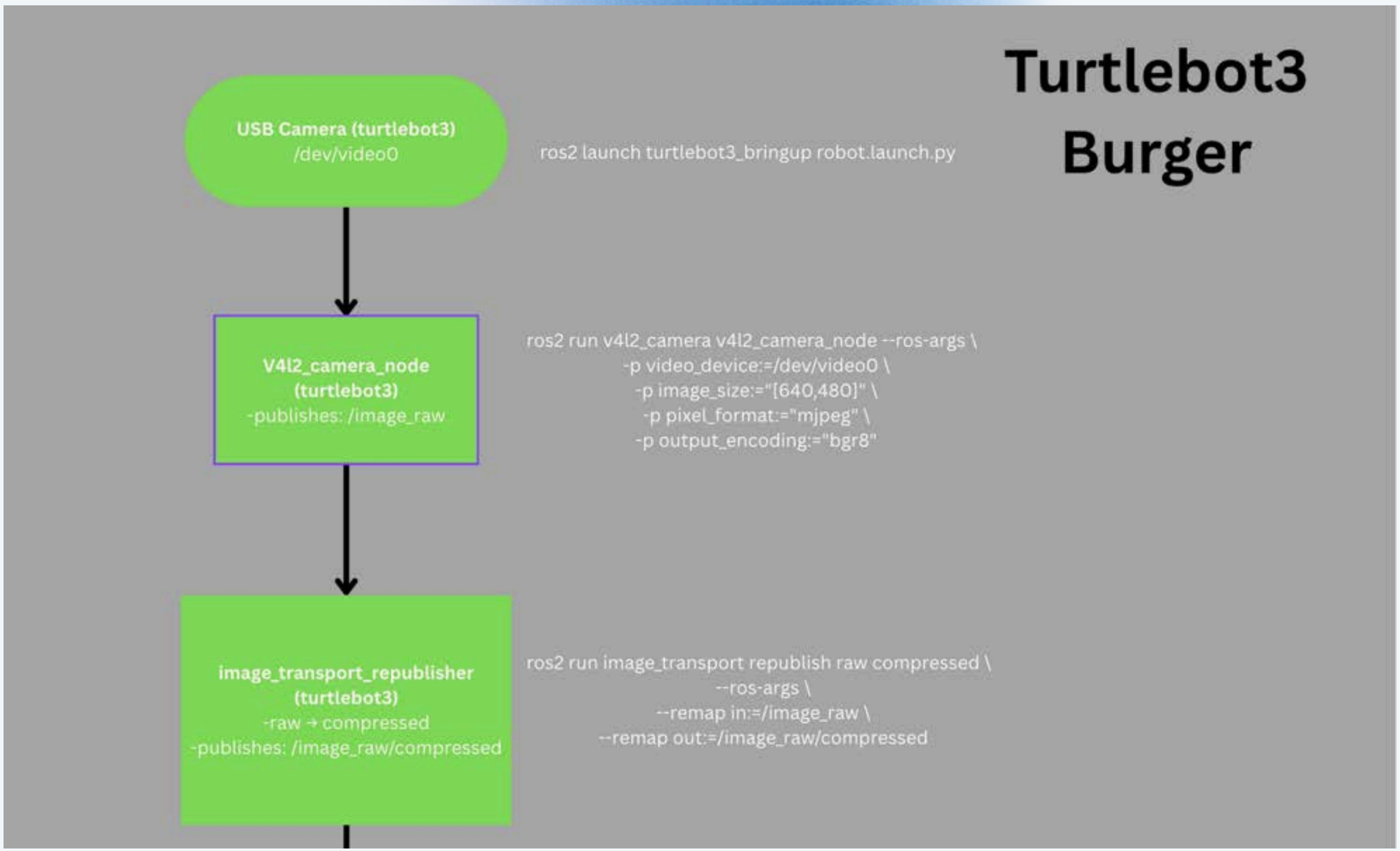
ROS2 LAUNCH

- ros2 launch turtlebot3_navigation2 navigation2.launch.py map:=$HOME/jjjmap.yaml
- ros2 run multi_point_nav multi_point_nav
- ros2 run multi_point_nav hand_gesture_node

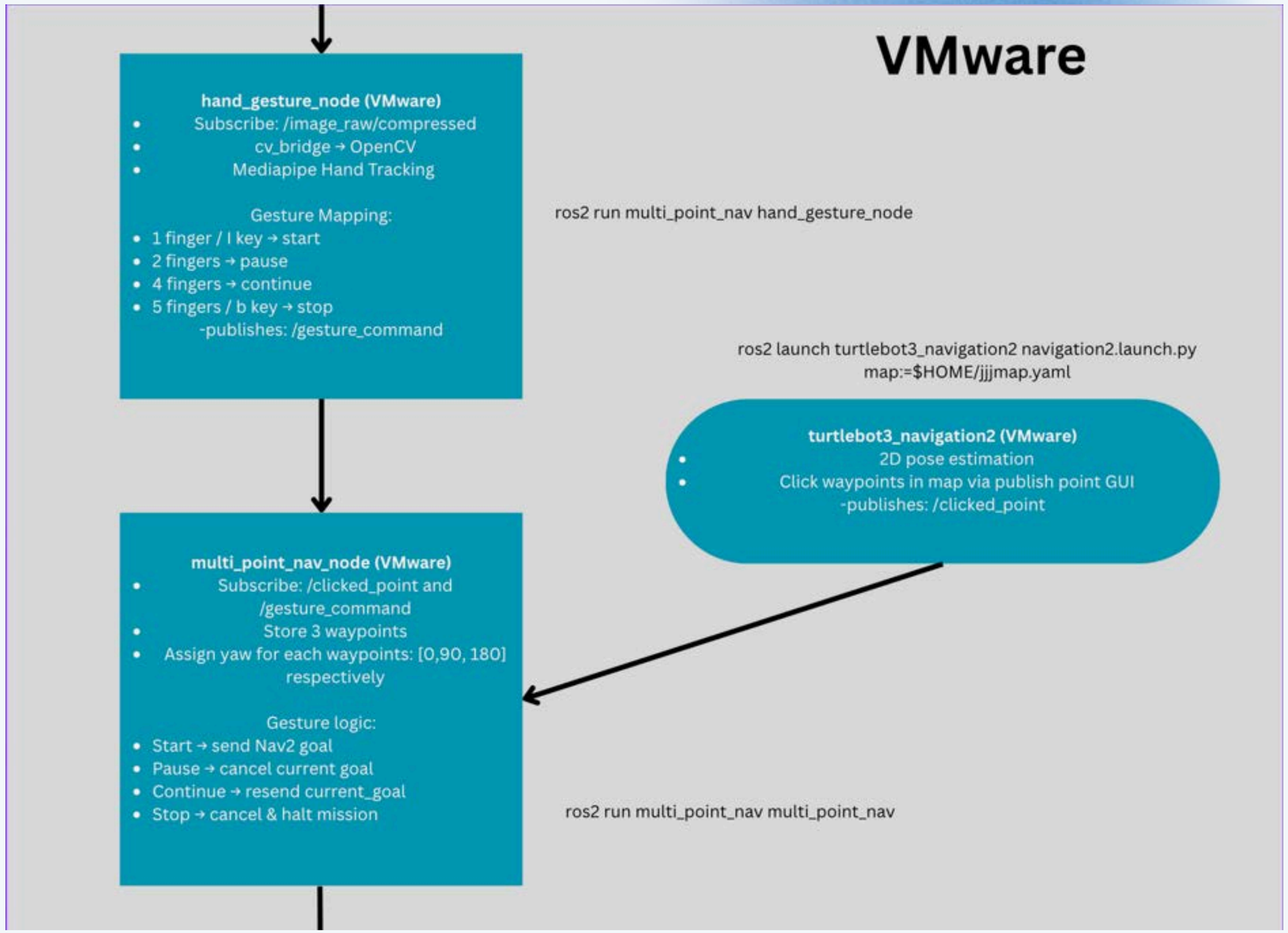
WORKFLOW


Turtlebot3 Burger
USB Camera (turtlebot3)
/dev/video0
ros2 launch turtlebot3_bringup robot.launch.py
V4l2_camera_node (turtlebot3)
-publishes: /image_raw
ros2 run v4l2_camera v4l2_camera_node --ros-args \
-p video_device:=/dev/video0 \
-p image_size:="[640,480]" \
-p pixel_format:="mjpeg" \
-p output_encoding:="bgr8"
image_transport_republisher (turtlebot3)
-raw → compressed
-publishes: /image_raw/compressed
ros2 run image_transport republish raw compressed \
--ros-args \
--remap in:=/image_raw \
--remap out:=/image_raw/compressed
VMware
hand_gesture_node (VMware)
Subscribe: /image_raw/compressed
cv_bridge → OpenCV
Mediapipe Hand Tracking
Gesture Mapping:
1 finger / l key → start
2 fingers → pause
4 fingers → continue
5 fingers / b key → stop
-publishes: /gesture_command
ros2 run multi_point_nav hand_gesture_node
ros2 launch turtlebot3_navigation2 navigation2.launch.py map:=$HOME/jjjmap.yaml
turtlebot3_navigation2 (VMware)
2D pose estimation
Click waypoints in map via publish point GUI
-publishes: /clicked_point
multi_point_nav_node (VMware)
Subscribe: /clicked_point and /gesture_command
Store 3 waypoints
Assign yaw for each waypoints: [0,90, 180] respectively
Gesture logic:
Start → send Nav2 goal
Pause → cancel current goal
Continue → resend current_goal
Stop → cancel & halt mission
ros2 run multi_point_nav multi_point_nav
Turtlebot3 Burger
Nav2 Stack (turtlebot3)
bt_navigator
planner_server
controller_server
amcl / slam_toolbox
turtlebot3 hardware
Motors
Encoders
IMU

WORKFLOW


Turtlebot3
Burger
USB Camera (turtlebot3)
/dev/video0
ros2 launch turtlebot3_bringup robot.launch.py
V4l2_camera_node
(turtlebot3)
-publishes: /image_raw
ros2 run v4l2_camera v4l2_camera_node --ros-args \
-p video_device:=/dev/video0 \
-p image_size:="[640,480]" \
-p pixel_format:="mjpeg" \
-p output_encoding:="bgr8"
image_transport_republisher
(turtlebot3)
-raw → compressed
-publishes: /image_raw/compressed
ros2 run image_transport republish raw compressed \
--ros-args \
--remap in:=/image_raw \
--remap out:=/image_raw/compressed

WORKFLOW



VMWARE_LAUNCHFILE

- ros2 launch turtlebot3_navigation2 navigation2.launch.py map:=$HOME/jjjmap.yaml
- ros2 run multi_point_nav multi_point_nav
- ros2 run multi_point_nav hand_gesture_node

navigation2.launch.py + multi_point_nav_node +hand_gesture_node = l**aunch file**

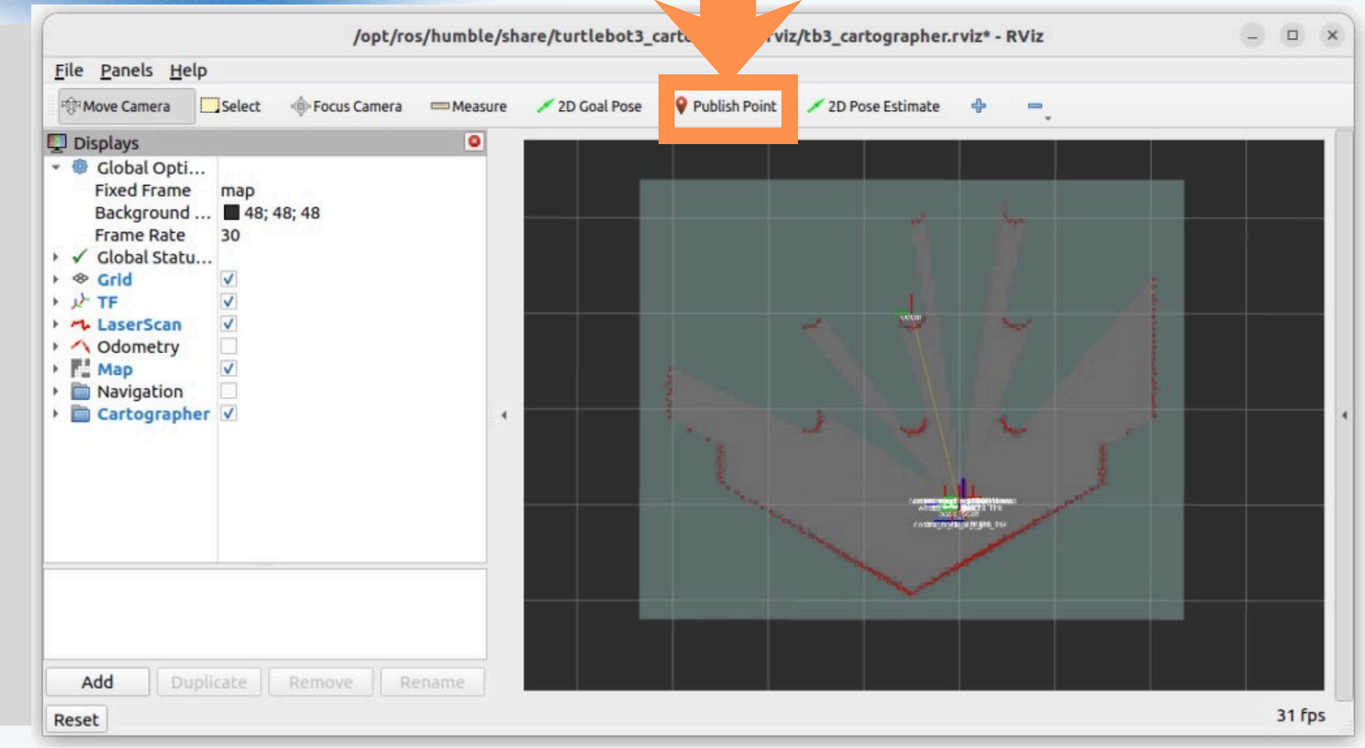
WORKFLOW

ros2 launch turtlebot3_navigation2 navigation2.launch.py
map:=$HOME/jjjmap.yaml

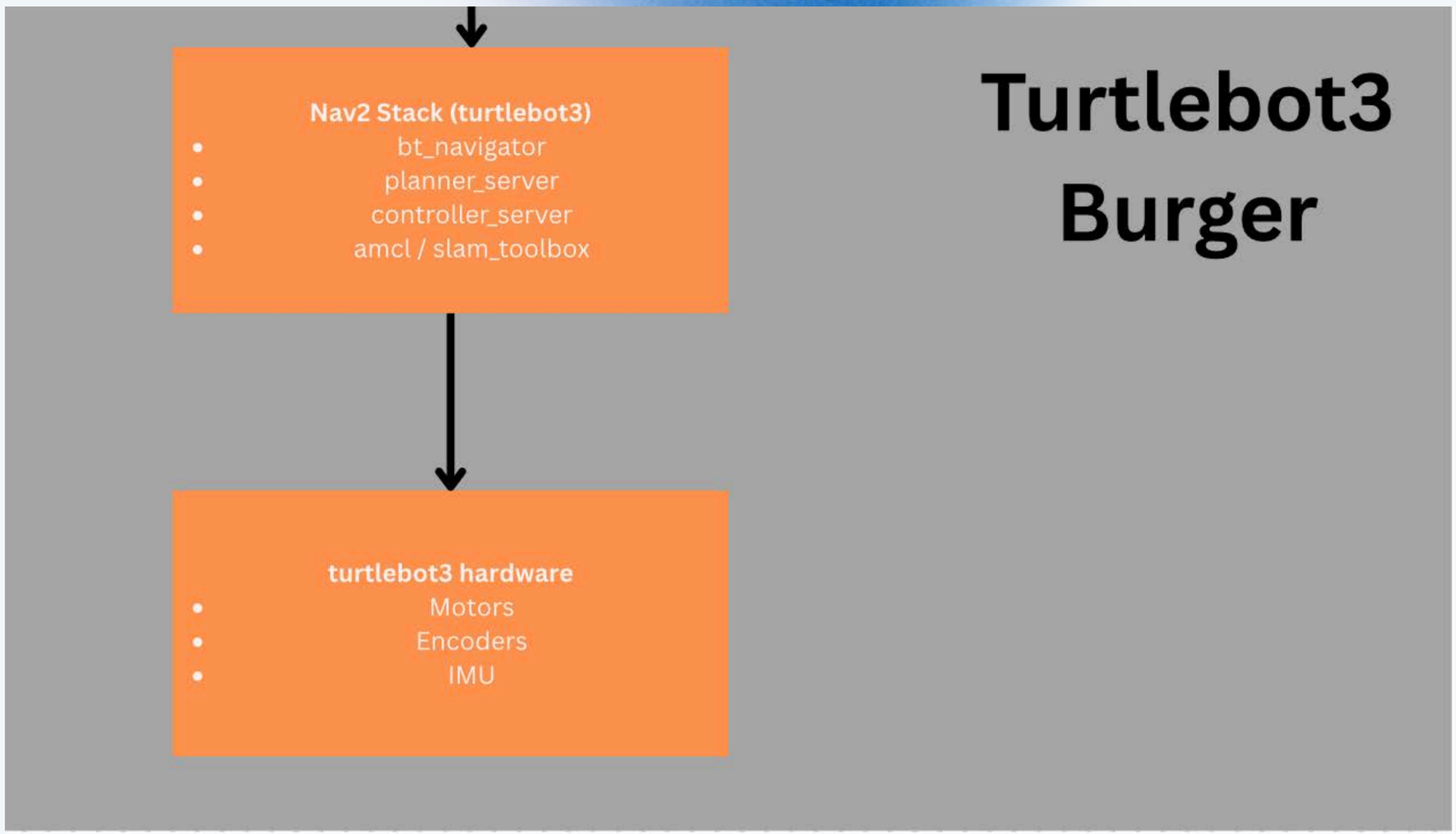
turtlebot3_navigation2 (VMware)

- 2D pose estimation
- Click waypoints in map via publish point GUI
  -publishes: /clicked_point

Collect 3 way points



WORKFLOW


Nav2 Stack (turtlebot3)
bt_navigator
planner_server
controller_server
amcl / slam_toolbox
turtlebot3 hardware
Motors
Encoders
IMU
Turtlebot3
Burger

VIDEO DEMONSTRATION

HTTPS://YOUTU.BE/UW6X6YV6_6K?SI=UW3O-MWZBLI-BDVC

THANK YOU