Introduction to ROS

(Robot Operating System)

Objective:

1. To understand what ROS is
2. To understand why we need ROS
3. Theory on some fundamental concepts of ROS

Note: Using ROS to automate robots and performing complex tasks on robots will be covered in the next CTE course (Intermediate Robotics) in the next semester

What is ROS?

- **Framework** for creating software for robots.

- **Meta-**operating system.
  (Provides hardware abstraction, low-level device control, parallel processes, communication between processes, device drivers, etc.)

- Collection of **tools, libraries, packages,** etc. to simplify the creation of complex behaviour in robots.

“Fetch an Item” task

Consider an office-assistant robot.

Task - ‘**Fetch a Stapler**’

Imagine you’re building the software for this task.

Question 1: What individual tasks must the robot perform to complete this task?

Question 2: How will these tasks be implemented on hardware and software?

Example from:
Programming Robots with ROS
-Morgan Quigley, Brian Gerkey, William Smart

Tasks involved -

1. **Understanding the Request**
    a. **Audio -** Verbal (NLP)
    b. **WiFi -** Web Interface: App, Email,...
    c. **Mobile phone -** SMS, Bluetooth, Call,....

2. **Finding the room to search**
    a. **Knowing current position -** Camera for Landmarks, GPS, Dead reckoning, Wheel encoders
    b. **Room location -** Stored map, stored room coordinates, random search,...
    c. **Planning a path to the room -** Navigation algorithms, motion planner, PID controller,...

3. **Reaching the Room**
    a. **Locomotion -** Differential Drive, Omni-wheel drive, Stair climbing controller,...
    b. **Obstacle avoidance -** Obstacle-detection (LiDaR, Stereo Camera, etc.),....

Tasks involved (continued)

4. **Locating the item -** Computer Vision

5. **Reaching for the item -** Inverse Kinematics calculations, motion planning

6. **Picking up the item**
7. **Finding the destination**
8. **Detecting & locating the requester**
9. **Delivering the item**

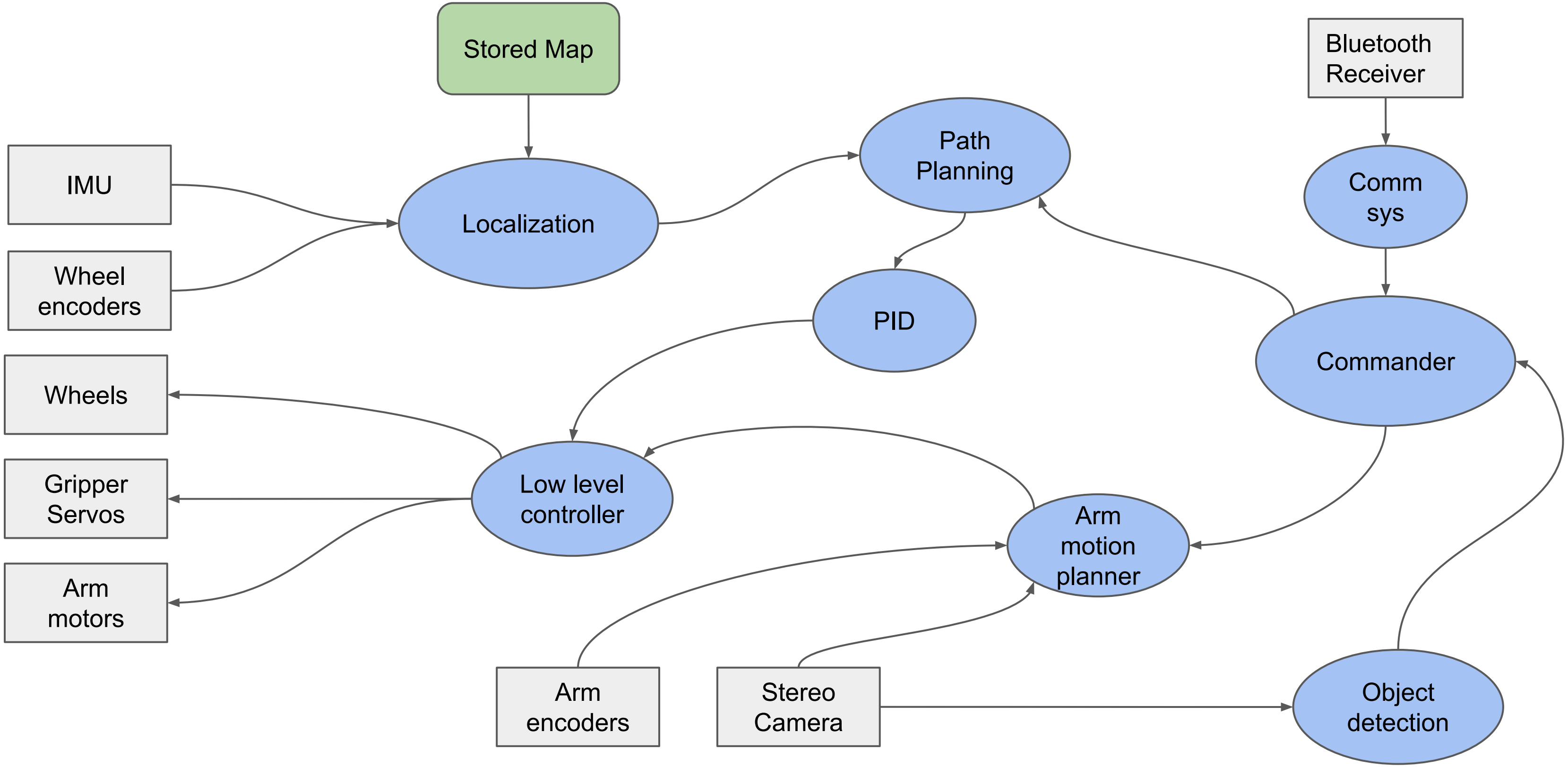
Subsystems involved

- Communication system
- Locomotion
- Odometry/Localization
- Computer Vision
- Path Planning (Navigation)
- Arm motion planning
- Low level Controller
    - Every wheel
    - Every motor of the arm & gripper
    - Feedback from all sensors (cameras, IR sensors, encoders, communication devices, etc.)

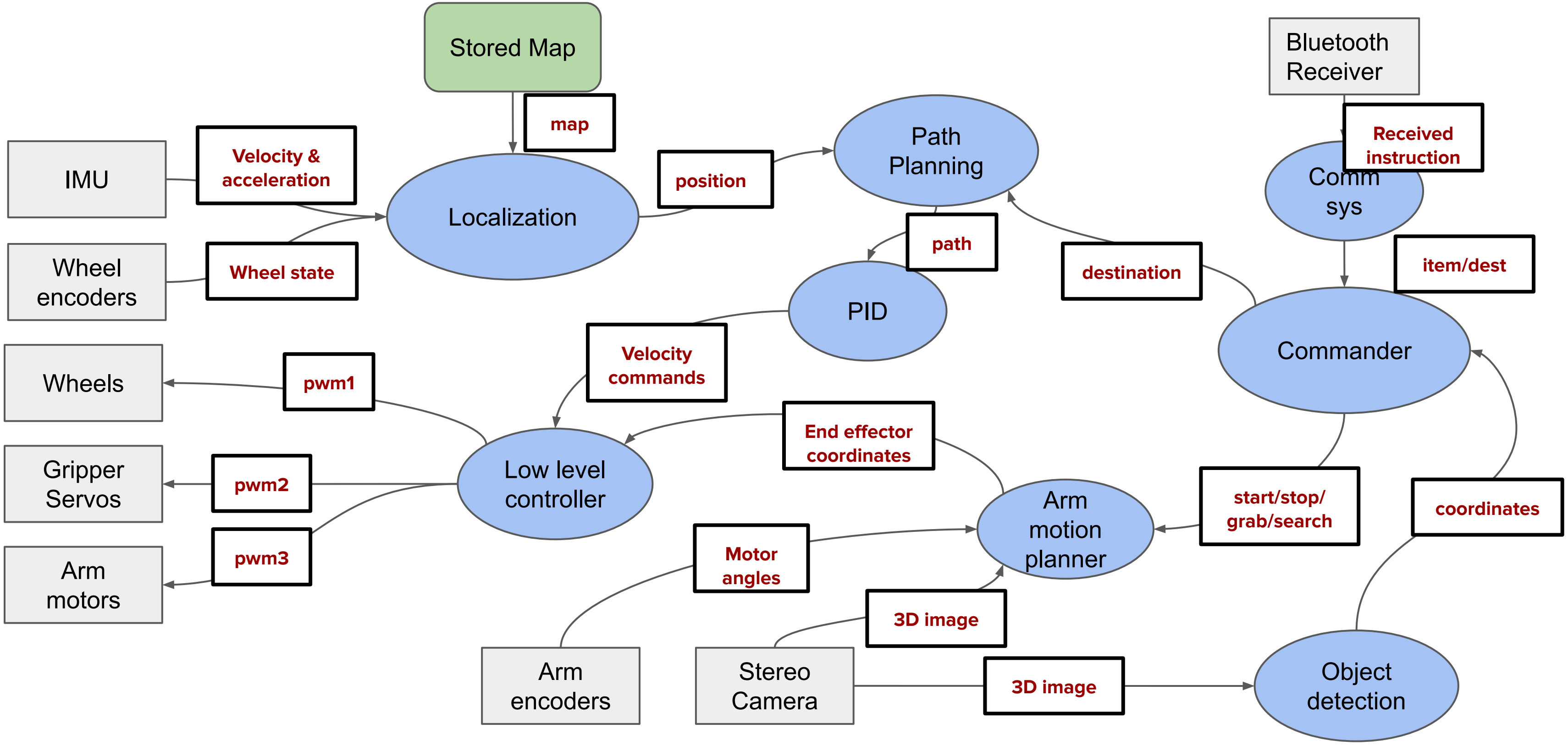
?

What other tasks can such a bot be used for?

Interaction between Subsystems



Messages Passed

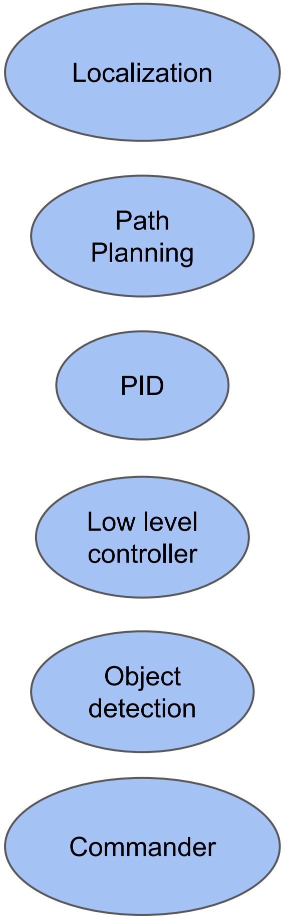


ROS Nodes

- Individual parallelly running processes (written in C++ or Python)

- A node usually does the following:
  - Takes input data **(Subscriber)**
  - Gives an output of processed data **(Publisher)**

- Eg: The Object detection Node
  Input: 3D image (Matrix)
  Output: Coordinates of the detected object (ordered pair)



Localization

Path Planning

PID

Low level controller

Object detection

Commander

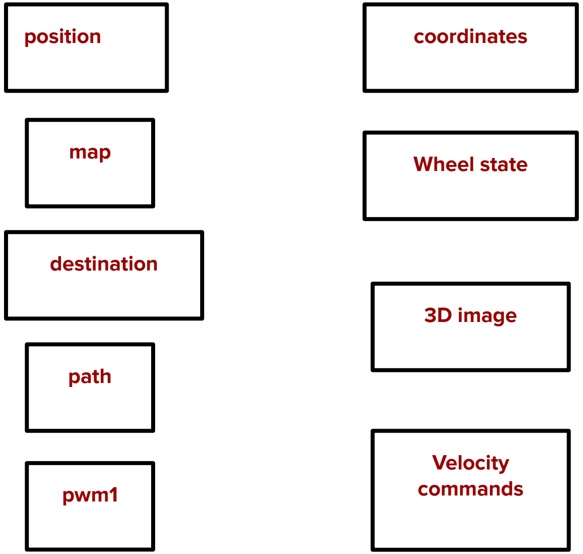
ROS Topics

- Communication between nodes takes place through **‘Topics’.**

- Output of a node is **Published** on a particular Topic.

- Any node that has **Subscribed** to this topic gets this data as input.

- Eg:
    - At every new frame, the **camera** publishes a 3D image to the topic **3D_image**
    - 2 nodes subscribe to this topic: Arm motion planner and object detection



position

coordinates

map

Wheel state

destination

3D image

path

Velocity commands

pwm1

ROS Messages

- **Message:** The data published on a particular topic

  Eg: The message (**x: 2.34   y: 1.61**) published on the topic ‘position’

- **Message type:** Type/Format of data published

  Eg: The message type for the topic ‘position’:

  float64 x

  float64 y

Overall Process

1. For each input hardware component, a **node** takes the input and **publishes** it to the respective **topic** as a **message.**

2. Different nodes **subscribe** to these topics and process the data.

3. Intermediate nodes communicate with each other through topics & messages

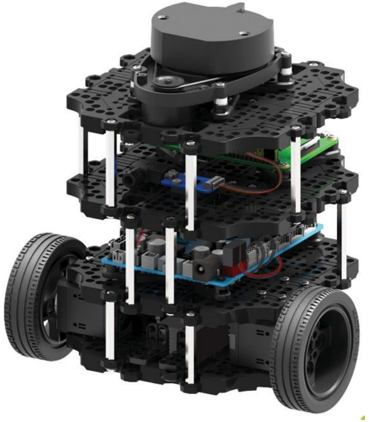
4. Finally, a low level controller node sends data as output to the actuators via GPIO or Pyserial.

Pre-built ROS Packages

- A **package** contains various interdependent codes (including nodes, topics, message types, etc.) & other files, built and optimized for a specific purpose.

- Nodes from installed packages can directly be used in your project

- Eg: The package *turtlebot_teleop* allows us to control the movement of a turtlebot using a keyboard



Other (relatively) Advanced Concepts

- ROS Master
- Catkin
- Services
- Actions
- Parameter Server
- Launch files

These will be taught in detail in the next CTE course, Intermediate Robotics

Installation & Resources

Installing ROS:

Ubuntu 16.04: http://wiki.ros.org/kinetic/Installation/Ubuntu
Ubuntu 18.04: http://wiki.ros.org/melodic/Installation/Ubuntu

Tutorials:

http://wiki.ros.org/ROS/Tutorials

Books:

- Morgan Quigley, Brian Gerkey & William Smart - **Programming Robots with ROS**
- Carol Fairchild, Thomas Harman - **ROS Robotics by Example**
- Wyatt S. Newman - **A systematic Approach to Learning Robot Programming with ROS**

THANK YOU!