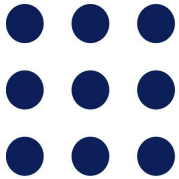


Introduction to
Robot Operating System

Why is simple Arduino programming insufficient?

Example: The classic “Fetch a Stapler” task

What would the void loop() function look like?

- How many functions will the loop execute?
- How will the components interact with each other?
- Is it enough to carry out everything step by step, or is simultaneous execution necessary?
- Will the processor handle load?
- Can the robot be simulated?

Recap: Tasks involved in the FetchBot -

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.),....

...more tasks

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**

Further possible scenarios -

Modifications for testing/ upgrading/ redesigning:

- Bluetooth based request was implemented, but a further WiFi based option needs to be included.
- The existing map now needs to be updated constantly
- A better processor is available and the software needs to be shifted to the new processor
- Odometry is not working, so the programs have to be tested in simulation
- Classical image processing algorithm for object detection needs to be replaced by a new Deep Learning based method.

Takeaway:

- The software for the robot needs to be **modular.**
  A module for a specific task should be replaceable by another, without affecting the rest of the code.

- Multiple programs should be able to **execute simultaneously.**

- The programs should be able to **communicate** with each other and with the hardware.
  A (virtual) medium should exist to allow a program to send 'messages' to another program and a framework to manage all communications.

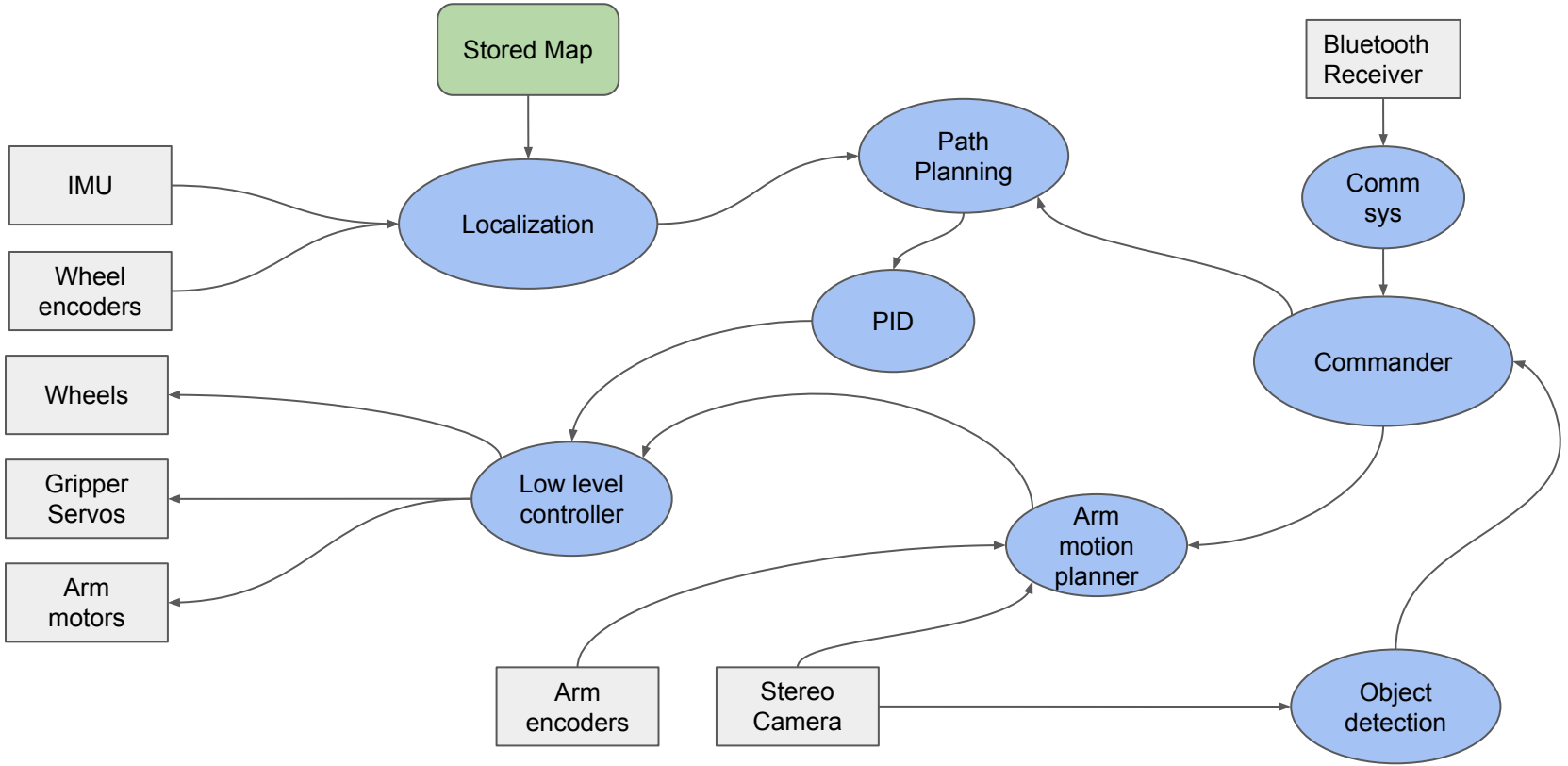
- The software should be as **platform independent** as possible.

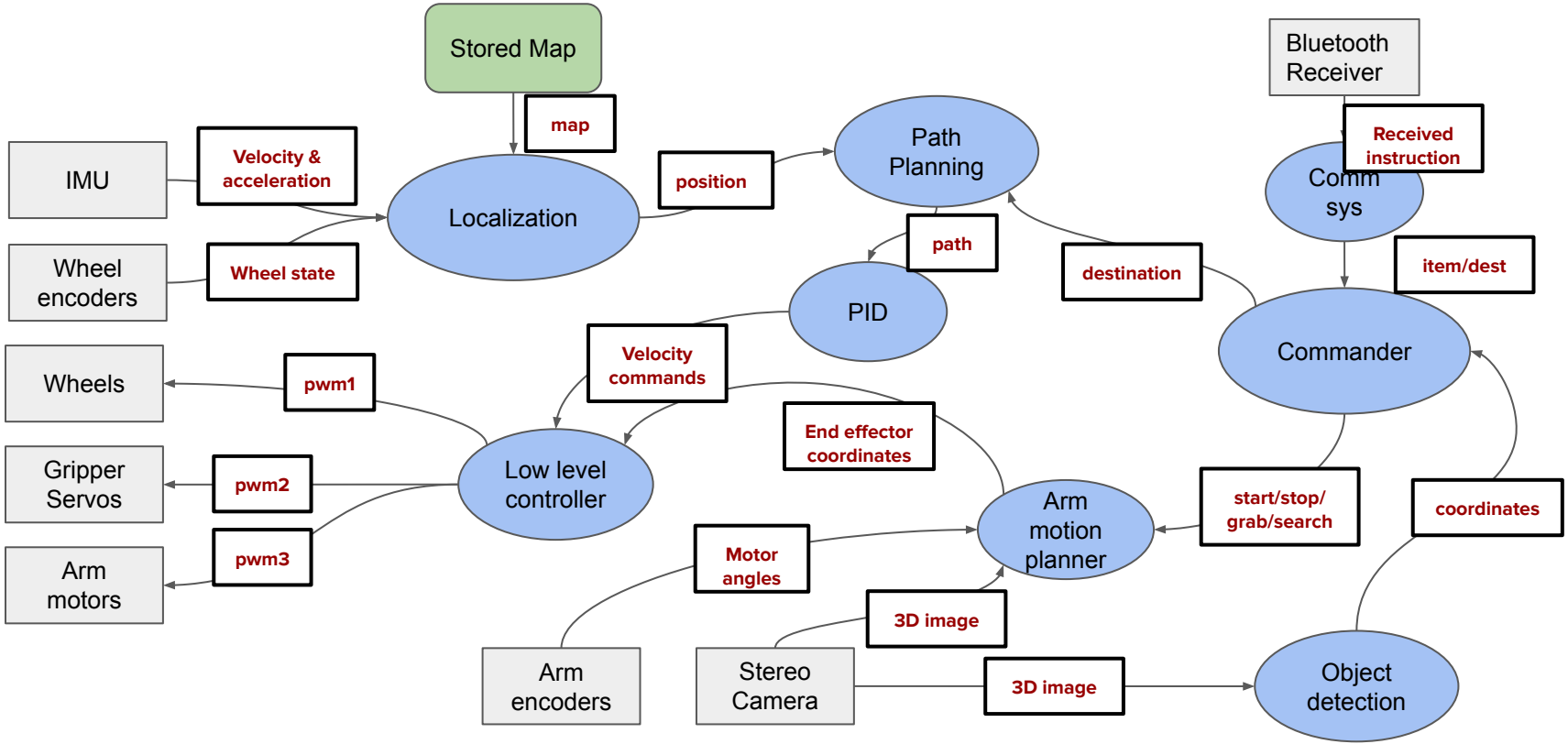
- It should be possible to **simulate** parts (or the entirety) of the robot.

ROS:

- A meta-operating system
- Collection of frameworks, SDK’s, software tools, libraries, packages, etc.
- Manages the interaction and simultaneous execution of all modules of the robot software for the developer.
- Contains pre-built, ready-to-use packages for most commonly used robots, sensors, functions, etc.
- *“Provides a robotic standard so you don’t need to reinvent the wheel”*

Ideal structure of the FetchBot



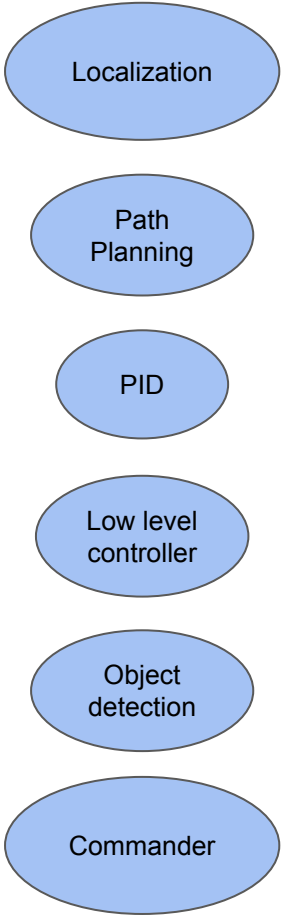

Stored Map
map
Bluetooth Receiver
Received instruction
IMU
Velocity & acceleration
Localization
position
Path Planning
Comm sys
path
destination
item/dest
Wheel encoders
Wheel state
PID
Commander
Wheels
pwm1
Velocity commands
End effector coordinates
Gripper Servos
pwm2
Low level controller
Arm motion planner
start/stop/ grab/search
coordinates
Arm motors
pwm3
Motor angles
3D image
Arm encoders
Stereo Camera
3D image
Object detection

Basic components in ROS:

- **Nodes -** The programs that execute together and communicate with each other
- **Topics -** The media of communication for specific purposes
- **Messages -** The information passed between nodes
- **The ROSMaster -** Program that enables the communication of nodes through messages over topics

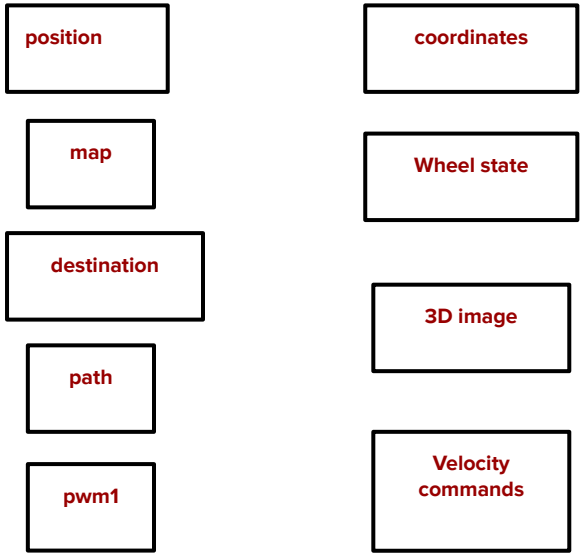
ROS Nodes

- 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)



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



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.

Further concepts:

ROS Services:

- Remote procedure calls
- Allow one node (client) to call a function that executes in another node (server)
- *Eg: Update Waypoint*

Actions:

- Goal oriented remote procedures
- The procedure keeps on executing and uses feedback to know when the goal is accomplished.
- *Eg: Go to waypoint*

Installing ROS

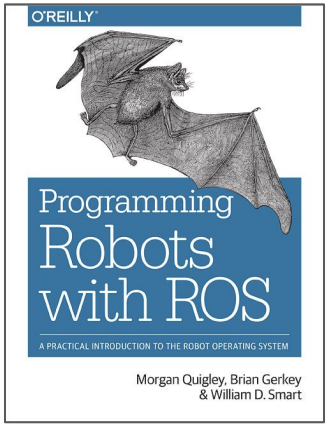
Requires Ubuntu or other Linux based OS

For Ubuntu 16.04: **ROS Kinetic**
http://wiki.ros.org/kinetic/Installation/Ubuntu

For Ubuntu 18.04: **ROS Melodic**
http://wiki.ros.org/melodic/Installation/Ubuntu

Follow the step by step instructions on the link on your linux terminal to install ROS.
Make sure you have a reliable internet connection.

Resources for Learning ROS



Book for ROS:

Programming Robots with ROS - Morgan Quigley, Brian Gerkey, William D. Smart

Online Resources:

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

BITS Goa QSTP (2019):

https://github.com/hardesh/QSTP-Introduction_to_ROS/