**ROBOTICS**

**DAY - 1**

A machine designined to execute one or more task automatically with speed and precision by Human guidance.

**Need:**

Human safety,High production,work by 24/7

**Isaac Asimov-Three Laws of Robotics**

1.A robot may not injure a human being.

2.A robot must obey the order given by human being.

3.Must protect its own existence.

**3H’s in Robotics:**

Head Hand Heart

Joint, Jetson nano,

Controllet instruction is constructed by the processor.Based on sensor data ,it will instruct the controller.

Reinforcement learning concept is used to short distance finding.Reward learning.

Motion planning-Autonomous vehicle,Path planning

Artificial Intelligence-CSE

Control systems-ECE

General Science: Physis,Mathematics

Mechanical Eng: Kinematics ,Dynamics,Sensing

**Classification of Robotics:**

**1.Robots with fixed base(manipulators)**

Eg:Arm

Serial Parallel

PUMA-programmable universal Steward platform

Machine for Assembly

**2.Mobile Robots**

Wheeled Robots Tracked Robots Multi-legged robots

Low speed High speed

Drone-unmanned aerial vehicles

**Industrial Robot/Manipulator nomenclature**

1.Base

2.Link/Links-connects adjacent joints

3.Joint/Joints-Degree of Freedom

4.Actuator-Orientation

5.End-Effector

6.Robot Control Unit-Robot Computer

**Degre of freedom**: in a mechanics context,are specific,defined models in which a mechanical device or system can move.

**Actuators:** responsible for moving and controlling a mechanism

Hydraulic actuators, Electric Actuator, Pneumatic Acutuators.

**Gripper/End Effector**: device at te end of a robotic arm.

**Control Unit**: CPU,Wireless unit,Sensing,control unit

**Mobile Robot Nomenclature:**

Wheels-standard wheels,Orientable wheel(Centered and off-centered),Ball/Caster wheel,Omni wheels.

Tracked Robot wheels(chain type),Multi legged robot,LIDAR(light detection and ranging)Hydraulic pressure,Stereo vision(camera)

**Applications**

Industrial, automotive, health care, agriculture, aerospace, space, retail, construction , entertainment

Palaticing(arranging),welding, neurosurgery cutting bone, weed cotroller robot, cockpit, amazon warehouse monitor stock level, wall plastering , paying games.

Master and slave robot

ROS-ros.org

**Adv**: **Disadvantages:**

Decreased production cost Capital cost

Shorter cycle times Expertise

Improved quality and reliability Limitatiosn

Reduced waste Poential job losses

Increased safety

Attract more customers

Better floor utilization

**Human-Robot interaction:**

Speech recognition, gestures, facial expression, artificial emotions

**Types of industrial robots based on joint types:**

Articulated, Cartesian, Cylindrical, Polar, SCARA, Delta.

**Articulated**-robot with rotary joints, used for 360 degree, used for arc welding, spray painting.

**Cartesian**: robot whose three principal axes are linear and at right angles to each other. Used for pic and place work. Eg: 3d printer

**Cylindrical**: for shipping the objects in conveyor belt.

**Spherical/Polar robot:** used for handling machine tools,spot welding,gas welding

**SCARA: Selective Compliance Assembly Robot Arm or Selective compliance articulated robot arm** assembly operations (mostly used one)

**Delta robot :** parallel robot maintains orientation of end effector.eg: 3d printing

All interdisciplinary concepts:

**Industrial Robot manufacturing companies**: kuka, fanuc, staubli, abb, Yaskawa, Kawasaki, Mitsubishi electric

**Mobile robotics**: clear path robotics, hushbarion robotics, robotis,

**Introduction to Robotics Frameworks**

Robots.ros.org-for details about robots and category

Framework-collection of libraries and tools. Cohesion, message library etc.,

**Robot Software platform:**

ROS-robot operating system

MSRDS-Microsoft Robotics Developer studio

ERSP-Evoltion Robotics Software platform

OpenRTM,OROCOS, OPROS, NAOqi Os,Player,yarp, marie, urbi,carmen,MOOS and orca

**Linux for Robotics:**

**Open source** , take the linux kernel and combine it with other free software to create comlete packages. Distribution-version in windows, ubuntu, rasbian

Network operationgs, running databases, desktop computing, server , scientific computing tasks.web servers such as apache, running mbl devices with os

**ADV:** Linux distributions: stackware, ubuntu, Debian,fedora,red hat.

many distribution,security,low cost.

**DIS**: confusing

**Importance:** framework that sits on top of an existing os such as GNU/linux.Packages are provided for Ubuntu Linux to help get your robot up and rolling.

Github.com/Durgaprasad-SB/

Clone and download copy the **link()**

In terminl hit **git clone link**

If git command gets error type - Sudo apt intall git

If error occurs

Sudo rm/var/cache/apt/archives/lock

Sudo rm/var/lib/dpkg/lock

Sudo apt-get install git

**Ls**-list the directory inside the directory

**Cd**-change directory

**Pwd**-present working directory

Files->computer->home->Bagavathy

**Touch** pythonfilename.py

**Python** filename.py – to execute the python file or program

**Touch** command is used to create a file

To edit the file – **gedit** filename

**Nano** filename.py to edit the code in shell

**Vi** filename.py – another edit option(while using server side)

Insert and command mode

**Ctrl+c** to come out of the command

I-insert mode

Escape-> **:wq** or :q to exit

**cd..** to move back the folder

**cd**-to return to the home directory comepletley

**mkdir**-to create a folder

**cp**- copy from one loation to another

cp <fromloc> to <des>

**rm**- is to remove the file

**cp -r** to copy the whole folder containing files

**rm -r** to remove the folder containing files

**mv** – copy the folder

**mv -t** to move one folder to another

**File permission:**

**Ls -la – to see the permission owner, group, all users**

r- read permission

w- write permission

x- executable

Chmod +x <filename> execute permission

Chmod +w <filename> write permission

Chmod +r <filename> read permission

r->4

w->2

x->1

total-7

7 7 7 4 2 1

Rwx rwx rwx r w x

Owner group alluser owner group alluser

**Python programming**

Guido van hassum

**Theconstructsim.com** - python and linux for robotics

**Day 2**

**ROS**

**It is a flexible framework for writing robot software.** It is a collection of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms.

It is an open-source, meta-os for your robot. It provides hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and packet management. It also provides tools and libraries for obtaining , writing and running code across multiple computers.

Os contains deice drivers, libraries, debugging tools, message passing , execution tools, compile tools, intaller, package create and release.

ROBOT 🡨------🡪 Meta Operating System (ROS) 🡨----🡪 Sensor

APP

**Why ROS?**

Reusability of the program

It is communication- based program

Support of development tools

Active community and open source

Formation of the ecosystem.

Hardware -🡪 Os -🡪 Application 🡪 User

Robot, Sensor 🡨🡪 Ros 🡨🡪 App

**Ros –**

Programming languages

Installer

File systems roscd

Compiler tools catkin\_make

Execution tools rosrun

Message passing rosmaster, rosmsg

Debugging tools rviz

Device drivers camera drivers

Libraries opencv, pcl, tf

Intelligent Modules navigation

Simulators gazebo, player/stage

Applications fetch, beer

**Components of ROS:**

Client layer

Robotics application

Robotics application framework

Communication layer

Hardware interface layer

Software development tools

Simulation

**Active distribution of ros**:

Melodic Moriana( may 2023), Ros kinetic kame(April 2021)

From box turtle to kinetic kame.

Kinetic kame: it is more powerful

It is supported in ubuntu 15.10 and 16.04

Jessie(Debian 8) for raspberry pi

Areas: pc , vmware workstation , virtual box

ROS development system

**Ros installation steps:**

1. Configure your ubuntu repositories
2. Setup your sources list
3. Set up your keys
4. Installation (desktop-full install)
5. Initialize rosdep
6. Environment setup
7. Dependencies for building packages
8. Creating a ros workspace

Ros.org

Kinetic for version 16

Lsb\_release -a to find the version

In github page

Copy and paste the set up source code in terminal

And enter password

In the same way the set up key

Installation

**Ros workspace**

In workspace folder we will have

\* Build- compiled files

\* Devel – development files

\* Src-source files

Inside the workspace where the ros project will be put

**Ros basic terms:**

**Master:** act as a name server for node-to-node connections and message communicaition.

It communicates with slaves using XML remote procedure call.

**ROS\_MASTER\_URI** address uses the IP address of local pc, and port no is 11311.

**Node:** smallest unit of processor running in ROS. Think of it as executable program.

**Metapackage:** it is a set of packages that have a common purpose.

**Eg:**

**Keyboard\_teleop\_pkg**

**1.topic**

Node 1 topic 1 🡨------------------------------🡪 Node 2 topic 2

Publisher (messages) Subscriber

(talker) (listener)

**Topic: It is like a topic in convo.** The publisher node first registers its topic with the master and then starts publishing messages on a topic.

**Publish and publisher:** the term ‘publish’ stands for the action of transmitting relative messages corresponding to the topic.

**Subscribe and subscriber:** the term ‘subscribe’ stands for the action of receiving relative messages corresponding to the topic.

**2.Service:**  It is synchronous bidirectional communication between the service client that request a service regarding a particular task and the service server that is responsible for responding to request.

**Request the server**

**Node 1 <--------------------------------------------- Node 2**

**Service --------------------------------------------🡪 Service**

**Server topics will be send Client**

**3.action :**

Node

Action server

Add two nos

4 feedback

Action: it is another message comm method used for asynchronous bidirectional comm. Action is

**CMakeList.txt:** catkin , which is the built system of ROS, uses CMake by default. The build environment is specified in the ‘CMakeLists.txt’ file in each package folder.

**Package.xml: an** xml file contains package information that describe the name, author , license and dependent packages.

**Ros master node(roscore):**

Roscore is a coll of nodes and programs that are pre-requisits of a ros-based system.

You must have a roscore running in order for ros nodes to communicate.

It is launched using the roscore command.

**Publisher steps:**

1. Determine a name for the topic to publish.
2. Determine the type of the messages that the topic will publish
3. Determine the frequency of the topic publication.
4. Create a publisher object with parameters chosen
5. Keep publishing the topic message at the selected frequency.

**Subscriber steps:**

1. Identify the name for the topic to listen to.
2. Identify the type of the messages to be received.
3. Defne a callback function that will be automatically executed when a new message is received on the topic.
4. Start listening for the topic messages.
5. Spin to listed for ever.

Ros Service

Service server service client

**Steps to create a client/server for ROS service application:**

1. Define the service message(service file)
2. Create ros server node
3. Create ros client node
4. Execute the service
5. Consume the service by the client.

Ros Action

Action server action client

**Parameter server:**

It will store some parameter like port no, calibration parameter etc.,

**Message communication:**

Message(topics,services,actions,parameters)

Subscriber

Service client

Action client

Publisher

Service server

Action server

Topic

Service request

Service response

Parameter

Server

(Ros master)

Roscd- is used to find the path of ROS

Ros is added to every command for using the ros workshop

Turtle\_sim

Rospack and double press tab

Rospack find ros\_package\_name – to find the path of the package.

Roscd package\_name – to change the directory to the package\_name

Rosls- to list the folders

Rospack find turtlesim –

Roscore- to activate the master

Rosnode list- to see the active nodes /rosout

Rosnode info /rosout- to get publications, subscriptions, services.

Rostopic list – to get the topics

Rostopic info topic\_name – to get the topic information (publisher,subscriber)

Rosmsg show rosgraph\_msg/log - to display the messages

Rosrun packg\_name

Clear – is used to clear the shell

Rosrun <pkg name> <node\_name>

Rosrun turtlesim turtlesim\_node - turtle window will appear

Rosrun turtlesim turtle\_teleop\_key

Rosnode info /turtlesim

Rosrun pckg\_name node\_name

Rosrun turtlesim turtlesim\_node \_\_name:=my\_turtle - to create our own turtle( it will not function because we are copying only the package)

Rosrun turtlesim turtlesim\_node

Rostopic info/turtle/cmd\_vel

Rosmsg show geometry\_msgs/Twist

Rostopic echo /turtle1/cmd\_vel

Rostopic pub -1 /turtle1/cmd\_vel geometry\_msgs/Twists –‘[2.0,0.0,0.0]’

Rosrun turtlesim turtlesim\_node – to run the turtle

Reservice list

Rosservice info /clear

Rossrv show std\_srvs/Empty

Rosservice call/clear- to clear the turtle service

Rosservice info /spawn

Rossrv show turtlesim/spawn

Rosservice call /spawn 5.0 5.0 20 ‘turtle2’ - to get another turtle

Exercise: ros.org prepare installation document,

**DAY -3**

Catkin- is a tool to compile the code.

In git hub go ros\_ws

[Sudo apt-get install ros-kinetic-rqt]

Cd catkin\_ws

And in that src folder will be there

Catkin\_make

Ls – build devil src

Cd src – to put all the source files in that folder python,msg,text, etc,.

Using the cakin\_make command we will ge cmakelis.txt contain the required info about files

**GUI TOOLS:**

Rviz – robot visualisation – 3d visualisation tool (default while installation)

Rqt- Qt – based ROS GUI development tool

Rqt\_image\_view – Image display tool ( a type of rqt)

Rqt\_graph – a tool that visualises the correlation between nodes and messages as a graph

Rqt\_plot – 2d data plot tool ( a type of rqt)

Rqt\_bag – GUI based bag data analysis tool ( a type of rqt)

Type rviz in shell

In add we will add the sensors

Sudo apt-get install ros-kinetic-rqt to install rqt

If any error , then type

sudo rm /var/cache/apt/archives/lock

Sudo rm /var/lib/dpkg/lock-frontend

We have to install the plugins

Sudo apt-get install ros-kinetic-rqt-common-plugins

Roscore

Rosrun turtle turtlesim\_node

Rostopic pub /turtle1/cmd\_vel geometru\_msgs/Twist -r 1 ‘[2.0,0.0,0.0]’ ‘[0.0,0.0,1.8];

Rosrun rqt\_graph rqt\_graph

Rosrun rqt\_plot rqt\_plot

In topic box we have to give /turtle

**For creating new project:**

Roscd

In new terminal type cd catkin\_ws/devel/ to create the development libraries

Pwd gives home/bagu/catkin\_ws/devel

In new terminal give

Gedit ~/.bashrc we will get text document – it gives the shell script document , in that we are changing the workspace as default.The last line will be set as default location. So we are going to add the command at the last line.

Go to the end and give

Source /home/Bagavathy/catkin\_ws/devel/setup.bash

Source ~/.bashrc

Close all terminal and type roscd

Above all steps are created to set up the environment to create the project.

Catkin\_create\_pkg <pkg\_name> <dependencies> for creating the package in the source folder

Rospy – python as language

Std\_msgs – for message communication ( int ,float etc)

Cd ../src to get into the src folder

Nodes – multiple nodes

Topics –> publisher and subscriber

Msgs 🡪subscriber

-🡪 publisher

Catkin\_create\_pkg simple\_node rospy std\_msgs for creating node

Ls

Cd simple\_node

Gedit package.xml

Cd ../..

Catkin\_make

It will create folders in built and devel also.

Cd src

Cd simplenode/src/

No files will be there . and create one using

Touch node\_one.py

Gedit node\_one.py

In github go to simple\_node

And in src folder

And node\_one

And copy paste the code in text editer

Ls -la

Chmod +x node\_one.py

It will change to executable file (color will be changed)

Cd

Cd catkin\_ws

Catkin\_make to compile to save the changes

Source devel/setup.bash sourcing devel file to bash file

The node we created in the src folder will also save in the built and devel folder

In new terminal enter rosrun simple\_node node\_one.py

Rosnode list gives /rosout

We rosrun to execute the python file of a node instead of python command.

It will run from anywhere whereas python cmmnd will run only in the crct directory.

Likewise create another node node\_one\_loop.py

For executing multiple nodes we using launch filed

<URDF> 3d method

In the source file inside the package , we have src and we have to add launch file.

Cd

Cd catkin\_ws/src/

Catkin\_create\_package launch\_package rospy std\_msgs

Cd ..

Catkin\_make

Create two or more launch nodes launch\_node\_one.py and launch\_node\_two.py . And give permission using

Chmod +x launch\_node\_one.py

Go out of the src folder cd ..

Mkdir launch

Cd launch

Touch my\_first\_launch.launch

Gedit my\_first\_launch.launch

In github page copy the launch code and edit the launch node names

<node pgk=”pkg\_name” type=”filename.py” name=”node\_name” output=”screen”>

</node>

Cd

Cd catkin\_ws

Catkin\_make

Roslaunch <pckg\_name> <launch\_file\_name>

In catkin\_ws path give

Roslaunch launch\_package my\_first\_launch.launch

**TOPIC:**

we have to create a packagefor topic

catkin\_create\_package topics\_package rospy std\_msgs

cd topics\_package/src

ls

touch publisher.py

chmod +x publisher.py

gedit publisher.py

copy and paste the topic code

likewise create the subscriber file

and copy paste the code.

In new terminal

Rosrun topic\_package publisher.py

Run subscriber and publisher in different shells

Run publisher and then subscriber

In ros.org🡪wiki🡪ros tutorials

(surge protection😊 ) ☹ 😐 😊 😊 😊

**DAY-4**

We need to create the msg folder inside the source folder.

For using dht sensor , we have to give id, name of sensor, temp, humidity

For creating a new workshop give the command

Mkdir -p ~/ros\_ws/src

Cd ros\_ws

Catkin\_make

Ls – have to give three folders

Cd src

Catkin\_create\_pkg dht\_package rospy std\_msgs

Cd ../ to get out of the source file

Catkin\_make – used to create and compile the package.

Again move to the dht\_package

Inside that cmakelist.txt src and package.xml

Cmakelist- contains the supported files in devel and build

Mkdir msg

Cd msg

Touch IoTsensor.msg

Gedit IoTsensor.msg

Cd ../

Gedit package.xml

To create our customised library for message generation

We have to give

<build\_depend>message\_generation</build\_depend>

Under the <build\_depend>std\_msgs</build\_depend>

<exec\_depen>message\_runtime</exec\_depend>

Save and close it

Gedit CMakelist.txt

In the find\_package (

Std\_msg

Rospy

Message\_generation

)

**Generte message in the ‘msg’ folder**

Uncomment the block and

Give IoTsensor.msg

In catkin\_package(

Uncomment the CATKIN\_DEPENDS ROSPY MSG\_STDS MESSAGE\_RUNTIME

)

**Generate add messages and services with any dependencies listed here**

Uncommend that

Save and clos

In ros\_ws directory give catkin\_make

We will get include folder in

Ros\_ws🡪devel🡪include🡪Iotsensor.h

It will get some code don’t modify it

In new terminal

Give cd ros\_ws

Source devel/setup.bash

Rosmsg show Iotsensor

It will display the mssg

Cd ros\_ws/src/dht\_package/src

Touch dht\_publisher.py

Gedit dht\_pub.py

Modify from dht\_package.msg import Iotsensor

Roscore

Cd

Cd ros\_ws

Source devel/setup.bash

Rosrun dht\_package dht\_publisher.py

Likewise create subscriber

And run the two files in a separate terminals

**Service:**

There is no motions in service. oNly it generate the action.

Cd ros\_ws/src

Catkin\_create\_pkg on\_off\_service std\_msgs rospy

Cd ../

Catkin\_make

Cd ros\_ws/src/onoff services/

Mkdir srv

Srv and msg are stadar d name we should not give by our own.

Request and response will be generated from the same file

So create one file

Touch ServiceExample.srv

Gedit ServiceExample.srv

Line above --- client request and below is server response

Cd ../

Gedit package.xml

Edit the build depend and execte depend

Gedit cmakelist.txt

Perform above four steps

Cd

Cd ros\_ws

Catkin\_make

Roscore

Rosrun on off service service\_server.py

Rosservice list

Rosservice call /service\_example “onezero: 1”

ON

**BUILDING TWO WHEEL ROBOT:**

**URDF:** Unified Robot Description Format

Robot model🡪 Simulate 🡪Interact

Urdf , xacro(xml macro), rviz

Insert the robot in gazebo world

Control the robot using a gazebo plugin

<http://wiki.ros.org/urdf/xml>

we have to create the urdf and xml in the package folder

cd ros\_ws/src/

catkin\_create\_pkg m2wr\_description urdf rospy stdmsgs

(m2wr – my two wheel robot)

Catkin\_make

Move to m2wr description folder

Mkdir urdf

Cd urdf

Ls – empty

Touch m2wr.xacro

Gedit it

Mkdir launch

Cd launch

Touch rviz.launch

Gedit

Joint\_state\_publisher package is used for action

Cd

Cd ros\_ws

Catkin\_make

Roscore

Cd ros\_ws

Source devel/setup.bash

Roslaunch m2wr\_description rviz.launch - for visualisation in rviz

For launching in gazebo give the command as

Roslaunch m2wr\_description spawn.launch – for simulation in gazebo

In fixed frame choose link chasis

In add , selec robot model and ok

We will get robot

Enable collision enabled option

Rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.py

Sudo apt-get install ros-kinetic-teleop-twist-keyboard

**DAY-5**

Open gazebo

In insert we will get models

Create and save the model

It will create file of .sdf extension . it continsxml code

Save the file world in the src folder in workspace in the launch file.

For robot we have to give a static map by the algorithm gmapping in ROS

In urdf folder create touch m2wr.gazebo

Copy and paste the 4 folders

M2.gazebo m2.xacro macros.xacro material.xacro

In launch folder copy and paste the rviz.launch and spawn.launch

Then compile it using catkin\_make

Sdk will create for a sensor

Here we are using LDAR sensor

Roscore

Cd

Cd ros\_ws

Source devel/setup.bash

Roslaunch m2wr\_desc rviz.launch

In fixed frame enable sensor\_light

And add robot\_model, laser scan

In wiki ros.org/gmapping

Slam=simultaneous localization and mapping

<http://wiki.ros.org/gmapping>

souce using

source devel/setup.bash

roslaunch m2wr\_des gmapping.launch

we have to install the package outside the workspace

sudo apt-get install ros-kinetic-slam-gmapping – for installing gmapping

In rviz enable map from add and in topic give /map

**Turtlebot 3 Burger:**

Servo motor work with TTL logic

Emanual.robotic.com/docs/en/platform/turtlebot3/overview

openCR – used for device to device communication

Processor🡪Controller 🡪Actuator

IMU – inertial measuring unit sensor

**Emanual.robotis.com**

Remote pc Turtlebot PC (Rsberry pi)

To establish the conn the two device should connect in the same pc

1. Connect the remote pc and turtlebot in the same network
2. Using ifconfig as the command check the ip address in wlps20 . Internet address will be displayed. Ip add of pc is the master (on- board intel,rba camera) intel real sense camera

**Create a workspace:**

**Mkdir -p ~/turtlebot3/src**

In simulation create turtlebot workspace

In src copy paste the github command from ros simulation

In ros setup

Go to pc setup

In ros simulation

Ros package for gazebo

Roscore

In simulation give the command in collision avoidance

In workspace give the commands

In simulation follow the commands

Sudo apt-get install ros-kinetic-map-server

Install dependent ROS packages

And in simulation follow the steps