GENERAL GUIDELINES OF PROFESSIONAL CONDUCT:

- Be in the Lab 10 min prior to the schedule. Latecomers will have a penalty of some marks, and an apology letter to be written and submitted.
- Always conduct yourself in a responsible manner in the laboratory.
- Follow all written and verbal instructions carefully. If you do not understand a direction or part of a procedure, ask the faculty before proceeding with the activity.
- Perform only those experiments authorized by your faculty. Carefully follow all instructions, both written and oral. Unauthorized experiments are not allowed.
- Observe good housekeeping practices. Work areas should be always kept clean and tidy.
- Be alert and always proceed with caution in the laboratory. Notify the faculty immediately of any unsafe conditions you observe.
- Labels and equipment instructions must be read carefully before use. Set up and use the equipment as directed by your faculty.
- Experiments must be personally always monitored. Do not wander around the room, distract other students, startle other students, or interfere with the laboratory experiments of others.
- Report any accident (breakage, short circuit, etc.) or injury (cut, burn, etc.) to the teacher immediately, no matter how trivial it seems. Do not panic

SAFETY INSTRUCTIONS

Don’t enter the fenced area of the robot while the robot is idle or working.

Wear safety shoes while working in the Laboratory.

Don’t operate /program the Robot at the full speed mode.

No unauthorized experiments/programs should be performed.

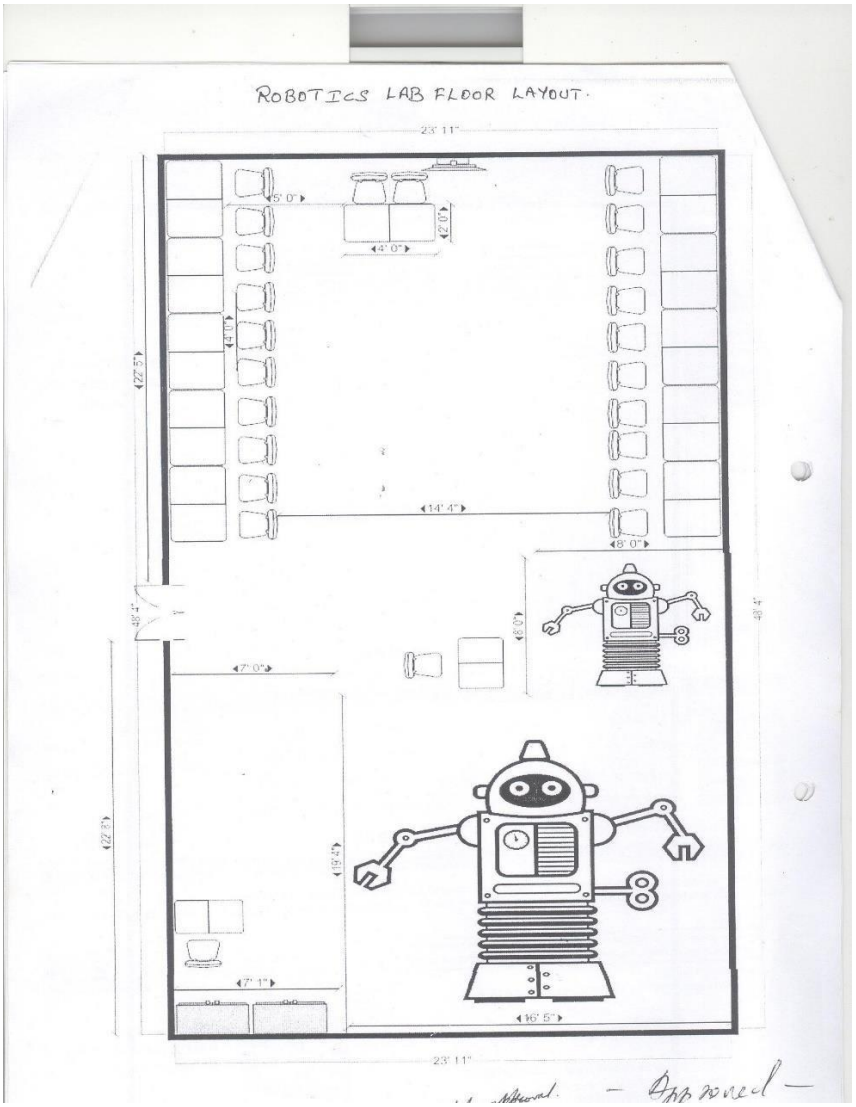
Do not eat or drink in the laboratory.

Before beginning to work in the laboratory you should be familiar with the standard operating procedure of each equipment/component you may be using.

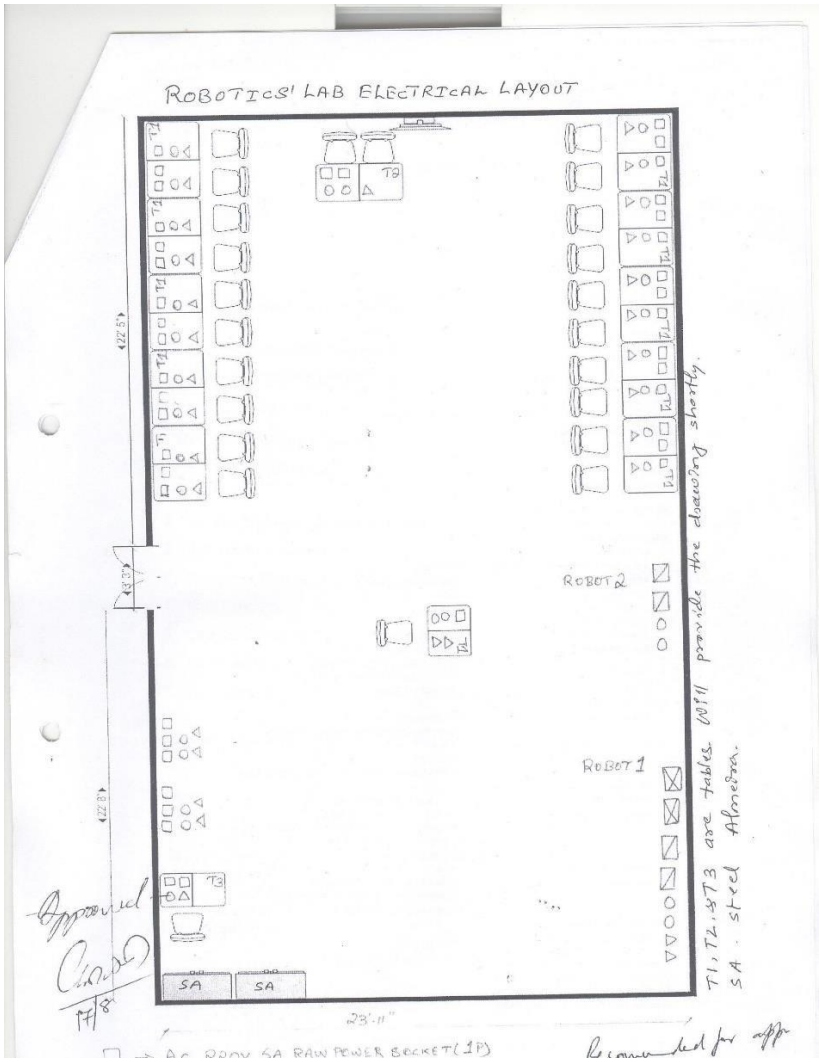
In case of an accident or any unexpected event like a short circuit, burning of components, sparks in or around the workplace, immediately inform the faculty or lab instructor.

EVACUATION ROUTE PLAN

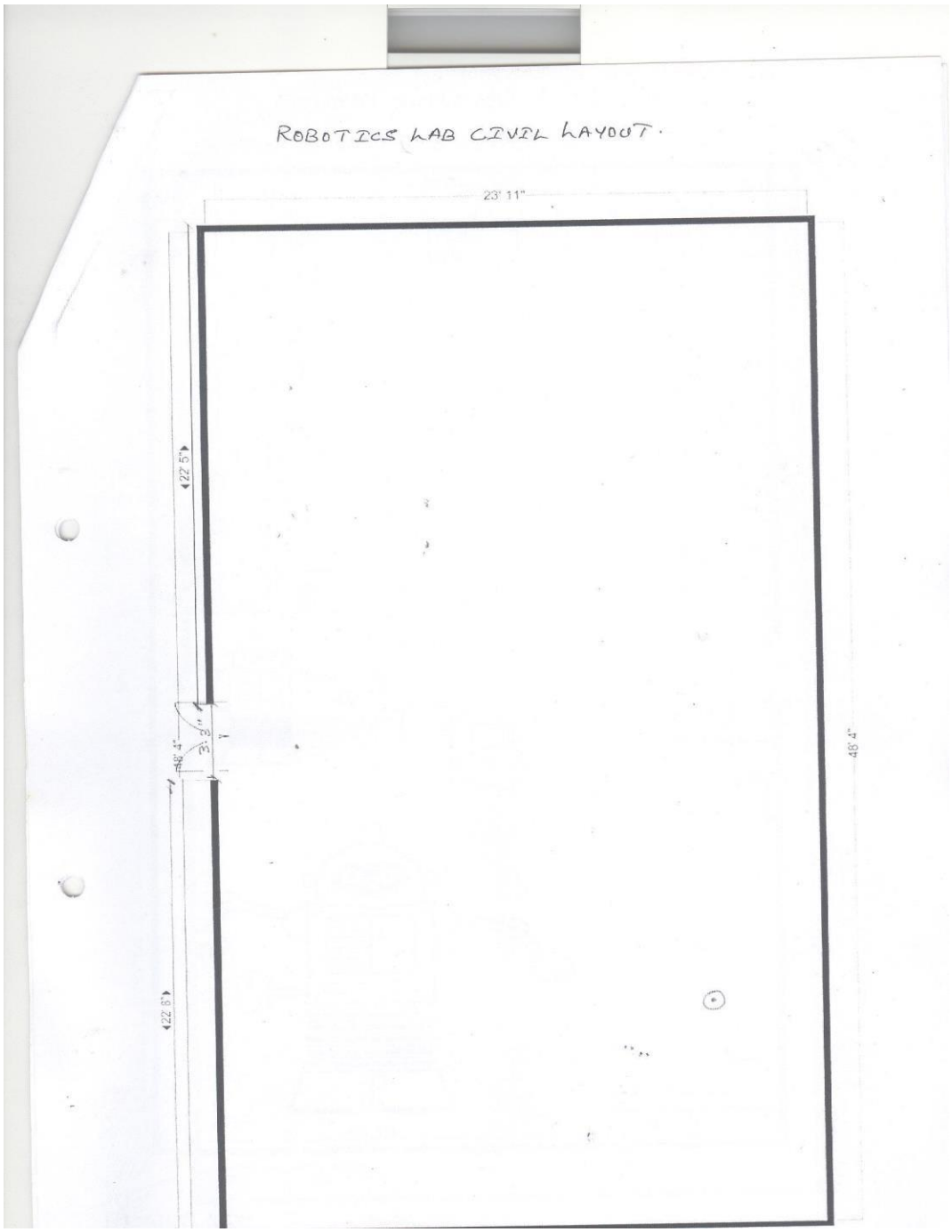
FLOOR PLAN/ELECTRICAL LAYOUT



FLOOR PLAN/ELECTRICAL LAYOUT



CIVIL LAYOUT



COVID SAFETY NORMS

1. Wearing masks, gloves, and shoes is mandatory for all labs.
2. Each student must sit/stand at the Yellow marked place only, both inside and outside the lab.
3. Maintain social distance from each other.
4. Cover your mouth with a tissue/handkerchief (or sleeve if there are no tissues available) when you cough or sneeze and wash your hands afterward.
5. Throw away used tissues in a bin and wash hands after contact with coughing/sneezing.
6. Enter the lab only after using the hand sanitizer which is kept outside each lab. Wash hands with soap and water after leaving the lab even when the hands are visibly clean.
7. Signs & symptoms of cough, fever, or difficulty in breathing should be reported to the lab faculty immediately.

Linux Programming for Robotics

Open Terminal: alt+ ctrl + t
Close Terminal ctrl+ d

Linux File System:

- How to navigate through a Linux file system
- How to interact with a Linux files system
- How to edit files using Shell (gedit editor)
- Manage access to files (permissions)
- Create simple Linux programs (Bash Scripts)
- Manage execution of Linux programs (processes)
- How to connect to the remote computer of a robot (ssh)
- Perception with ROS

Source: \url{https://www.hostinger.in/tutorials/linux-commands}

pwd
pwd command to find out the path of the current working directory (folder) you're in.

cd
cd command is used to navigate through the Linux files and directories. It requires either the full path or the name of the directory.
cd .. (with space and two dots) to move one directory up
cd (with tilde) to go straight to the home folder
cd- (with a hyphen) to move to your previous directory

ls
ls command is used to view the contents of a directory. By default, this command will display the contents of your current working directory.
ls -R will list all the files in the sub-directories as well

ls -a will show the hidden files
ls -al will list the files and directories with detailed information like the permissions, size, owner, etc.

cat
cat command is used to display the contents of a file on the standard on screen.
cat > filename creates a new file
cat filename1 filename2>filename3 joins two files (1 and 2) and stores the output of them in a new file (3)

cp
cp <file/folder we want to copy> <name of the new file/folder>
cp -r my_scripts/ my_scripts_copy/
cp –help
cp command to copy files from the current directory to a different directory.

mv command
mv command is used to move files and rename files.
To rename files, the Linux command is mv oldname.ext newname.ext
mv <file/folder we want to move> <destination>

mkdir
mkdir command is used to make a new directory
There are extra mkdir commands as well:
mkdir dirName
use the p (parents) option to create a directory in between two existing directories. For example,
mkdir -p Music/2020/Newfile will create the new “2020” file.

rmdir
rmdir command is used to delete a directory. However, rmdir only allows you to delete empty directories.

rm
rm command is used to delete directories and the contents within them. If you only want to delete the directory — as an alternative to rmdir — use rm -r.

touch
touch command allows you to create a blank new file through the Linux command line.
Example: touch /home/newfile.txt

locate

locate command is used to locate a file, just like the search command in Windows. Use -i for case insensitive.
locate -i word1*word2 to search for a file that contains two or more words.

find
find is used to search for files and directories. The difference is, you use the find command to locate files within a given directory.
As an example, find /home/ -name notes.txt command will search for a file called notes.txt within the home directory and its sub directories.
Other variations when using the find are:
To find files in the current directory use, find . -name notes.txt
To look for directories use, \/ -type d -name notes.txt

grep
grep is used to search through all the text in each file.
To illustrate, grep blue notepad.txt will search for the word blue in the notepad file. Lines that contain the searched word will be displayed fully.

sudo
Short for “SuperUser Do”, sudo command is used to perform tasks that require administrative or root permissions.

df
df command to get a report on the system’s disk space usage, shown in percentage and kilo bytes. If you want to see the report in megabytes, type df -m.

du
du (Disk Usage) command is to check how much space a file or a directory takes. However, the disk usage summary will show disk block numbers instead of the usual size format. If you want to see it in bytes, kilobytes, and megabytes, add the -h argument to the command line.

head
head command is used to view the first lines of any text file. By default, it will show the first ten lines, but you can change this number to your liking. For example, if you only want to show the first five lines, type head -n 5 filename.ext.

tail
tail command will display the last ten lines of a text file. For example, tail -n filename.ext

diff

diff command compares the contents of two files line by line. After analyzing the files, it will output the lines that do not match. Programmers often use this command when they need to make program alterations instead of rewriting the entire source code.
The simplest form of this command is diff file1.ext file2.ext

tar
tar command is used to archive multiple files into zip format, with compression being optional.

chmod
chmod is used to change the read, write, and execute permissions of files and directories.
chmod octal file – change the permissions of file to octal, which can be found separately for user, group, and world by adding:
4 – read (r)
2 – write (w)
1 – execute (x)
Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
chmod +x read.py give permission to file read.py for execution.

chown
chown command enables you to change or transfer the ownership of a file to the specified username. For instance, chown linuxuser2 file.ext will make linuxuser2 as the owner of the file.ext.

jobs
jobs command will display all current jobs along with their statuses. A job is basically a process that is started by the shell.

kill
kill is used to terminate unresponsive program. It will send a certain signal to the misbehaving app and instructs the app to terminate itself.
SIGTERM (15) — requests a program to stop running and gives it some time to save all of its progress. If you don't specify the signal when entering the kill command, this signal will be used.
SIGKILL (9) — forces programs to stop immediately. Unsaved progress will be lost.
Besides knowing the signals, you also need to know the process identification number (PID) of the program you want to kill. If you don't know the PID, simply run the command ps ux.

After knowing what signal you want to use and the PID of the program, enter the following syntax:
kill [signal option] PID.

ping
ping command is used to check your connectivity status to a server. For example, by simply entering ping google.com, the command will check whether you're able to connect to Google and also measure the response time.

wget
wget command line is used to download files from the internet. Simply type wget followed by the download link.

uname
uname command, short for Unix Name, will print detailed information about your Linux system like the machine name, operating system, kernel, and so on.

top
top is equivalent to Task Manager in Windows, the top command will display a list of running processes and how much CPU each process uses. It's very useful to monitor system resource usage, especially knowing which process needs to be terminated because it consumes too many resources.

history
When you have been using Linux for a certain period of time, you will quickly notice that you can run hundreds of commands every day. As such, running history command is particularly useful if you want to review the commands you have entered before.

man
man gives the detail of manual instruction of the command. Example man tail will show the manual instruction of the tail command.

echo
echo command is used to move some data into a file. For example, if you want to add the text, "Hello, my name is John" into a file called name.txt, you would type echo Hello, my name is John >> name.txt

zip, unzip
Use the zip command to compress your files into a zip archive, and use the unzip command to extract the zipped files from a zip archive.

hostname

If you want to know the name of your host/network simply type hostname. Adding a -i to the end will display the IP address of your network.

useradd, userdel
Since Linux is a multi-user system, this means more than one person can interact with the same system at the same time. useradd is used to create a new user, while passwd is adding a password to that user's account. To add a new person named John type, useradd John and then to add his password type, passwd 123456789.
To remove a user is very similar to adding a new user. To delete the users account type, userdel UserName

Use the clear command to clean out the terminal if it is getting cluttered with too many past commands.
Try the TAB button to autofill what you are typing.
Ctrl+C and Ctrl+Z are used to stop any command that is currently working. Ctrl+C will stop and terminate the command, while Ctrl+Z will simply pause the command.
If you accidental freeze your terminal by using Ctrl+S, simply undo this with the unfreeze Ctrl+Q.
Ctrl+A moves you to the beginning of the line while Ctrl+E moves you to the end.
You can run multiple commands in one single command by using the ";" to separate them. For example Command1; Command2; Command3. Or use if you only want the next command to run when the first one is successful.

Tab Completion

$ls -l Do
$clear
ctrl + a -beginning of the line
ctrl + e -End of the line
ctrl + ←
ctrl + →
ctrl + u - delete cursor to start
ctrl + k - delete cursor to end
ctrl + shift + c - copy to clipboard
ctrl + shift + v - copy from clipboard
ctrl + r - search command history
ctrl + c - cancel the command
uparrow - previous command
downarrow - next command

Type and note down the operations

$ ls
$ ls -l
long list for more details
$ ls -lh
{command}[options][argument]
$ls -lah /opt
$ls -a - list all
$ls -F list.txt
$man ls
$ls --help
$ls -l --human-readable
q to quit

apropos list - lists all those commands
$file myfile.txt - determines file type
$stat myfile.txt - display ownership
$cd Document/
$pwd - print working directory
$cd .. - previous directory
$cd ~ - goes to home folder

$echo Hello
$echo "Hello"
$cal
$cal 2017
$cal 12 2017
$cal -A 1 12 2021 - after one month
$cal -B 1 12 2021 - before one month
$cal -A 1 -B 1 12 2021 - after and before one month
$cal -y
$date
$date -u - universal time
$date --universal (long names preceded by --)
$history
$!1 - executes first command
$!! - executes previous command
$history -c; history -w clears history
$exit
$echo $PATH - gives the shell script path
$which cal
$which echo

$which which
$cat 1>output.txt - deletes the data written before

standard input - keyboard- 0
standard output - monitor - 1
error - 2

$cat >>output.txt - appends to the data written before
$cat 2>>error.txt - writes the error to error.txt
$cat -k xyz 2>error.txt - writes the error to error.txt
$cat input.txt
"Hello World!"
$cat 0<input.txt
$cat <input.txt
$cat <input.txt 1>output.txt
$tty - gives dev/pts/l
$date 1>date.txt
$cat 0<date.txt --delimiter = " " --fields 1
$date | cut --delimiter = " " --fields 1
$date | cut --delimiter = " " --fields 1 > today.txt
$date | cut >today.txt --delimiter = " " --fields 1
$date | tee full.txt | cut --delimiter = " " --fields 1 > today.txt
tee stores in the file full.txt also pipe helps to store data in today.txt
$date | echo "hello"
$date | xargs echo "hello"
$date | cut --delimiter = " " --fields 1 > today.txt
$date | cut --delimiter = " " --fields 1 | xargs echo

$rm file.txt
$cat file.txt | xargs rm
Aliases - nicknames for pipelines
$cd \texttildelow
$gedit .bash_aliases - shows hidden files

alias getdate = 'date | tee */home/asha/text.txt* | cut --delimiter = " " --fields 1 | tee /home/asha/date.txt | xargs echo hello'
save - close - open terminal
$getdates

open .bashrc_aliases
add the following lines
alias calmagic = "cal -A 1 -B 1 2021 > */home/asha/thing.txt"*
save and close

ctrl+alt+t
$echo 12 2017 | calmagic

wild cards *

ls *
ls D*
ls Do*
ls *.txt - all that ends with .txt
ls ?.txt - ? single letter.txt
ls ??.txt
ls file[1234567890].txt
ls file*.txt
ls file[0-9]/txt
ls file[A-Z].txt
ls file[0-9][0-9].txt
ls file[0-9][A-Z][a-z].txt
ls file[0-9ABC].txt
cd Desktop/
touch file.txt
touch *~/Documents/asha.txt*
echo "Hello" > hello.txt
mkdir fodername
mkdir ~/Pictures/myPics
mkdir -p myFolder
cd ~/Desktop/
mkdir \{jan, feb, mar\}_\{2020, 2021\}
mkdir \{jan, feb, mar\}_\{2020 .. 2025\}
touch \{jan, feb, mar\}_{2020 .. 2025\}/file\{1 ..10\}.txt
ls \{jan, feb, mar\}_\{2020 .. 2025\}/file\{1 ..10\} > output.txt
touch file\{A,B,C\}.txt
touch file\{A .. C\}.txt

Delete files and Folders
rm
rm asha.txt
rm Documents/asha.txt
touch file\{1 ..3\}.txt
rm *.txt - removes all those ends with .txt
rm file*
rm *2*
rm *.jpg
rm *\[2,3\]*
rm -r folderName

```
mkdir -p folder/file\{1,2,3\}
touch folder/file\{1,2,3\}/file\{1,2,3\}.txt
rm -r folder
rm -ri folder
```

Copy files and Folders

```
cp file1.txt file2.txt
rm file*
. - this folder
.. - previous folder
```

Rename files and folders

```
cd Desktop/
mv oldfile.txt newfile.txt
mv oldfolder newfolder
mv file* newfolder
mv newfolder/ ~ /Documents/
```

```
sudo apt install mlocate
locate *.conf
locate *.CONF
locate -i *.CONF
locate -i --limit 3 *.CONF
locate -S
locate -S >~/Desktop/data.txt
locate -i --limit 3 *.CONF>~/Desktop/list.txt
locate -e .com
locate -e .conf
locate --existing .conf
locate --follow .conf
locate --follow *.conf
locate --existing --follow -i --limit 5 *.conf
touch findme.txt
locate findme.txt
updatedb
man updatedb
sudo updatedb
locate findme.txt
cd ..
locate findme.txt
```

```
cd Desktop/
locate -S
locate -S > data_after.txt
locate --existing --follow findme.txt
find
find /home
find /etc
find /home/asha
cd Desktop/
mkdir asha.txt
mkdir asha
cd asha
touch asha.txt
cd ..
cd asha2
mkdir asha2
cd asha2
find
cd ..
find
cd ~/Documents/
mkdir level1
cd level1/
mkdir level2
cd level2
mkdir level3
cd level3
touch level.txt
cd ~/Documents/
find
find . -maxdepth 1
find . -maxdepth 2
find . -maxdepth 3
find . -maxdepth 4
find .
find . -type f
find . -type d
find . -type d -maxdepth 1
find . -maxdepth 1 -type d
find . -maxdepth 1 -type f
find . -name "5.txt"
find . -name "level3.txt"
find . -name "level4.txt"
find . -name "*.txt"
```

```
find . --maxdepth 3 -name "*.txt"
find . -maxdepth 3 -name "*.txt"
find . -maxdepth 2 -name "*.txt"
find . -maxdepth 2 -name "?.txt"
find . -maxdepth 2 -iname "*.TXT"
find /-type f -size +1000k
find /-type f -size +100k
find / -type f -size +100k
find / -type f -size +1000k
sudo find / -type f -size +1000k
sudo find / -type f -size +1000k |wc -l
sudo find / -type f -size +1000k | wc -l
sudo find / -type f -size +100k | wc -l
find / -type f -size +100k | wc -l
find / -type f -size +100k -size -5M
find / -type f -size -100k -size +5M
find / -type f -size -100k -o -size +5M
find / -type f -size -100k -o -size +5M | wc -l
cd Desktop/
mkdir copy
sudo find / -type  f -size +100k ssize -5M
sudo find / -type  f -size +100k size -5M
sudo find / -type  f -size +100k -size -5M
sudo find / -type  f -size +100k -size -5M |wc -l
sudo find / -type  f -size +100k -size -5M | wc -l
sudo find / -type  f -size +100k -size -5M -exec cp WHT WHERE
sudo find / -type  f -size +100k -size -5M -exec cp {} ~/Desktop/copy
sudo find / -type  f -size +100k -size -5M -exec cp {} ~/Desktop/copy  \;
sudo find / -maxdepth 3 -type  f -size +100k -size -5M -exec cp {}
~/Desktop/copy  \;
cd copy/
ls
cd ..
sudo find / -maxdepth 3 -type  f -size +100k -size -5M -ok cp {}
~/Desktop/copy  \;
touch  haystack/folder{1..500}/files{1..500}.txt
touch haystack/folder{1..500}/files{1..500}.txt
touch haystack/folder${shuf -i 1-500 -n l}/needle.txt
find haystack/ -type f -name "needle.txt"
find haystack/ -type f -name "needle.txt" -exec mv {} ~/Desktop \;


echo "Hello" >file1.txt
echo "there" >file2.txt
```

```
echo "Welcome" >file3.txt
cat file1.txt file2.txt file3.txt
cat file1.txt file2.txt file3.txt > file.txt
cat file.txt
cat file[1-3].txt > file.txt
ls
echo "abc" >>new.txt
echo "def" >>new.txt
cat new.txt
tac new.txt
cat file[1-3].txt | tac
cat file[1-3].txt | tac > reverse.txt
cat file[1-3].txt | rev
cat file[1-3].txt | rev | tac
cat file.txt | head
head file.txt | head -n 2
find | head -n 5
find | head -n 5 | tac
cat /etc/cups/cups-browsed.conf
cat /etc/cups/cups-browsed.conf | wc -l
head -n 20 /etc/cups/cups-browsed.conf | wc -l
head -n 20  /etc/cups/cups-browsed.conf
head -n 20 | cat /etc/cups/cups-browsed.conf
head file.txt | tail -n 2
tail -n 20 cat /etc/cups/cups-browsed.conf | wc -l
tail -n 20 /etc/cups/cups-browsed.conf | wc -l
tail -n 20 /etc/cups/cups-browsed.conf
head -n 1 file.txt | tail -n 1
find  | tail -n 3
find  | tail -n 3 > export.txt
cat export.txt
```

echo "asha" >> file.txt
echo "vihan" >> file.txt
echo "laxmi" >> file.txt

```
sort file.txt > sorted.txt
cat sorted.txt
sort file.txt | tac rev_sorted.txt
sort file.txt | tac > rev_sorted.txt
cat rev_sorted.txt
sort -r file.txt | less (sort the text)
sort -r numbers.txt | less (wrong output )
```

sort -n numbers.txt | less (n- number)
sort -nr numbers.txt | less (nr- number and reverse)
sort number.txt
sort -u number.txt | less (u- unique_
ls -l /etc/
ls -l /etc | head -n 20
ls -l /etc | head -n 20 | sort -k 5n
ls -l /etc | head -n 20 | sort -k 5nr (n number)
ls -l /etc | head -n 20 | sort -k 6hr (h human readable)
ls -l /etc | head -n 20 | sort -k 5hr
ls -lh /etc | head -n 20 | sort -k 5hr
ls -lh /etc | head -n 20 | sort -k 6M (M-month)
ls -lh /etc | head -n 20 | sort -k 6Mr (Mr-Month reverse)
ls -lh /etc | head -n 20 | sort -k 7r
ls -lh /etc | head -n 20 | sort -k 7nr
ls -lh /etc | head -n 20 | sort -k 2n
ls -lh /etc | head -n 20 | sort -k 2nr

echo "hello world" > file.txt
cat file.txt
grep e file.txt (search e)
grep -c e file.txt
wc -l file.txt
grep -i hello file.txt (i case insensitive)
grep -i a file.txt
grep -ic a file.txt
grep -i "hello world" file.txt
grep -iv a file.txt (v search without a)
grep -iv o file.txt
grep -iv h file.txt
grep -cv h file.txt
cat file.txt
grep -i "e" file1.txt file.txt
grep -ci "e" file1.txt file.txt
mkdir file
mv hello.txt hello
mv file.txt file
ls file/ | grep file.txt
ls -F
ls -F /etc | grep -v
ls -F /etc | grep -v / >files.txt
ls -F /etc | grep -v / sort -r > files.txt

ls -F /etc
ls -F /etc | grep magic
ls -F /etc | grep -v magic
ls -F /etc | grep -v magic >files.txt
ls -F /etc | grep -v magic| sort -r >files.txt
ls -F /etc | grep -v/ | sort -r >files.txt
ls -F /etc | grep -v / | sort -r >files.txt
ls -F /etc | grep -v /
man -k print | grep files

Archive and Compressing

ls
ls -lh
tar -cvf archive.tar file1.txt files.txt (c for create)
ls
ls -l | grep .tar
file archive.tar
mv archive.tar new.odt
file new.odt
mv new.odt new.tar
ls
tar -tf new.tar
rm file?.txt
ls
tar -xvf new.tar (x for extract)
ls
gzip new.tar
ls
ls -lh
gunzip new.tar.gz
ls
ls -lh
bzip2 new.tar
ls
ls -lh
bunzip2 new.tar.bz2
ls
zip other.zip file1.txt files.txt
ls
unzip other.zip
rm other.zip new.tar

```
ls
tar -cvf new.tar *.txt
ls
tar -cvzf new.tar.gz *.txt
ls
file new.tar.gz
tar -cvjf new.tar.bz2 *.txt
ls
rm *.txt | tar -xvf new.tar
rm *.txt
ls
tar -xvzf new.tar.gz
tar -xvjf new.tar.bz2
```

Introduction to Editors

Bash Scripting

```
nano our_script.sh
file our_script.sh
nano our_script.sh
which bash
nano our_script.sh

#!/bin/bash

echo "Welcome to bash scripting!!"

mkdir ~/Desktop/files
cd ~/Desktop/files
touch file{1..100}.txt
ls -lh ~/Desktop/files > ~/Desktop/file.log

file our_script.sh
nano our_script.sh
which python3
nano our_script.sh
bash our_script.sh
nano our_script.sh

nano backup.sh
which bash
nano backup.sh
```

```
#!/bin/bash
tar -cvzf backup.tar.gz ~/{Pictures} 2>/dev/null

cd ~
mkdir bin
mv ~/Desktop/backup.sh /bin
mv ~/Desktop/backup.sh ~/bin/
cd ~/bin/
ls
chmod +x backup.sh
nano backup.sh
ls
bash backup.sh
ls
cd ..
nano .bashrc
echo $PATH
backup
nano .bashrc
Add the foolowing line at the end
PATH="$PATH:$HOME/bin"

crontab -e

# 20 11 1 1(JUN) 0(SUN)
  * *  *   *   *     echo "Hello World" >> ~/Desktop/hello.txt

cd Desktop/
ls
cat hello.txt

crontab -e

# m h  dom mon dow   command
# 20 11 1 1(JUN) 0(SUN)
#  * *  *   *   *     echo "Hello World" >> ~/Desktop/hello.txt


#   0,15,30,45 * * * * echo "Hello World" >> ~/Desktop/hello.txt

    */15 * */3 * * echo "Hello World"
    59   23  * JAN,DEC SUN echo "we can"
```

```
cd bin
nano backup.sh

#!/bin/bash

tar -cvzf ~/backup/backup.tar.gz ~/{Desktop,Pictures,Videos} 2>/dev/null

date >> ~/backup/backup.log


ls backup/
backup
bash backup.sh

crontab -e

#    */15 * */3 * * echo "Hello World"
#    59   23  * JAN,DEC SUN echo "we can"

* * * * * bash ~/bin/backup.sh

cat backup.log
uname -o
www.GNU.org
```

Download, install, compile from GNU.org

```
https://ftp.gnu.org/gnu/coreutils/


cd /Documents
cd ~/Documents
ls
file coreutils-9.0.tar.xz
tar -xJf coreutils-9.0.tar.xz
ls
cd coreutils-9.0/
ls
cd src
ls
nano ls.c

add after main() {
```

printf("Hello\n");
ls | less
sudo apt-get install gcc
cd ..
bash configure
ls
sudo apt-get install make
ls
make
sudo make install

reverting back
cd ~/Documents/coreutils-9.0/src
nano ls.c
cd ..
make && sudo make install
www.ubuntu.com
lsb_release -a
uname -m
https://packages.ubuntu.com/focal/
apt-cache search docx
apt-cache search docx | grep text
apt-cache show docx2txt
apt-cache search "web server" | less
apt-cache show apache2
apt-cache search textt
apt-cache show apache2
cd /var/lib/apt/lists
ls
ls | less
ls | less -N
cd ~
apt-cache show gcc
cd /var/lib/apt/lists/
ls | less -N
nano archive.ubuntu.com_ubuntu_dists_focal_main_binary-amd64_Packages

Visual Studio

Open terminal
Install Visual Studio using the command
$sudo snap install --classic code

Refer \url{https://linuxhint.com/install_use_vs_code_ubuntu/} for more details.

Install extensions
1. Python
2. XML
3. Terminal

Python Programming for Robotics

>>python

or

>>python3

```
print("Hello")
print(4)
```

and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	async
def	for	lambda	return	await

+	Addition	1 + 1 = 2
-	Subtraction	2 - 1 = 1
*	Multiplication	2 * 2 = 4
/	Division	5 / 2 = 2
%	Modulus	5 % 2 = 1

=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x – 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%	x %= 3	x = x % 3

Expressions

An expression is a combination of values, variables, and operators. A value all by itself is considered an expression, and so is a variable, so the following are all legal expressions.

Order of operations

When more than one operator appears in an expression, the order of evaluation depends on the rules of precedence. For mathematical operators, Python follows mathematical convention. The acronym PEMDAS is a useful way to remember the rules: Parentheses, Exponentiation, Multiplication and Division.

Modulus operator
The modulus operator works on integers and yields the remainder when the first operand is divided by the second. In Python, the modulus operator is a percent sign (%).

String operations
The + operator works with strings, but it is not addition in the mathematical sense. Instead, it performs concatenation, which means joining the strings by linking them end to end.

Asking the user for input
Sometimes we would like to take the value for a variable from the user via their keyboard. Python provides a built-in function called input that gets input from the keyboard. When this function is called, the program stops and waits for the user to type something. When the user presses Return or Enter, the program resumes and input returns what the user typed as a string.

Comments
For this reason, it is a good idea to add notes to your programs to explain in natural language what the program is doing. These notes are called comments, and in Python they start with the # symbol.

Conditional execution
Boolean expressions
A Boolean expression is an expression that is either true or false. The following examples use the operator ==, which compares two operands and produces True if they are equal and False otherwise:

==	Equal	5 == 5
!=	Not Equal	4 != 5
>>	Greater than	5 >> 4
<	Less than	4 < 5
>=	Greater than or equal to	5 >= 4
<=	Less than or equal to	4 <= 5

if loop

```
i=1
if i==1:
    print('first')
elif 4==4:
    print('second')
elif 3==3:
    print('middle')
else:
    print('last')
```

while loop

```
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break
print("ROS")
```

for loop

```
for x in range(5):
    print(x)
```

```
while (1):
    print('enter a digit')
    num=input()
    var=str(num)
    if (ord(var) in range (48,58)):
        break
print('you entered BCD')
```

```
for x in range(10):
    if x % 2 == 0:
        continue
    print(x)
```

Exception Handling in Python

```
print('num')
num=input()
print('den')
den=input()
try:
    res= int(num)/int(den)
except:
    print("den cannot be 0")
else:
    print(res)
```

Functions

Built-in functions
Python provides a number of important built-in functions that we can use without needing to provide the function definition. The creators of Python wrote a set of functions to solve common problems and included them in Python for us to use.

int
len
max
min
type
float
str

Math functions

```
import math
math.sin()
math.sqrt()
math.log10
```

Random numbers

```
import random
random.random()
```

```
random.randint()


def my_func(num):
    return num*2

seq=[2,3,4,5,6,7]
map(my_func,seq)
a= list(map(my_func,seq))
print(a)


main()

if __name__ == "__main__":
 # execute only if run as a script
        main()
```

Classes

```
class Number:
        def __init__(self, val):
        self.val = val

obj = Number(2)
obj.val
```

Practice the codes from following Libraries:

numpy
matplotlib
math
random

Experiment 2: Introduction ROS2 Programming

Installation:

Step 1:

https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debians.html

Step 2:

https://docs.ros.org/en/foxy/Tutorials/Colcon-Tutorial.html

Before creating your first node you need to:

- Create a ROS2 workspace and source it.
- Create a Python package.

ROS organizes the program using packages. A package contains Cpp, Python, setup, compilation, and parameters files. They are:

- package.xml file containing meta-information about the package
- setup.py containing instructions for how to install the package
- setup.cfg is required when a package has executable so that ros2 run can find them
- /<package_name> a directory with the same name as the package, used by ROS2 tools to find the package that contains __init__.py

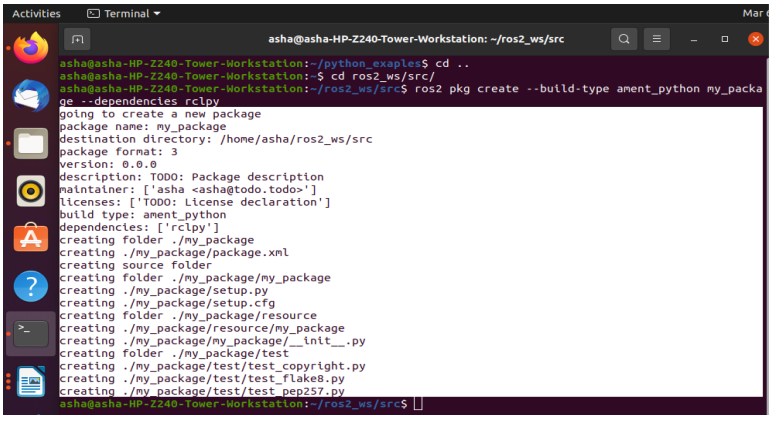
When you want to create packages, you need to work in a particular ROS workspace known as **ROS workspace**. The ROS2 workspace is the directory in your hard disk where your **ROS2. packages reside to be usable by ROS2. Usually, the ROS2 workspace directory is called ros2_ws**

Execute in Terminal #1

```
mkdir -p ~/ros2_ws/src
```

Inside this workspace, there is a directory called **src**. This folder contains all the packages created. Every time you create a package, you have to be in the directory **ros2_ws/src**.

```
cd ~/ros2_ws/src
ros2 pkg create --build-type ament_python my_package --dependencies rclpy
```



ros pkg create <package_name> --build-type ament_python my_package --dependencies <package_dependencies>

The **package_name** is the name of the package you want to create, and **package_dependencies** are the names of other ROS packages that your package depends on.

Execute in Terminal #1

gedit ~/.bashrc

source /opt/ros/foxy/setup.bash
source ~/ros2_ws/install/setup.bash
source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash

Execute in Terminal #1

ros2 pkg list
ros2 pkg list | grep my_package

ros2 pkg list: Gives you a list with all packages in your ROS system.

ros2 pkg list | grep my_package: Filters, from all of the packages located in the ROS system, the package is named my_package.

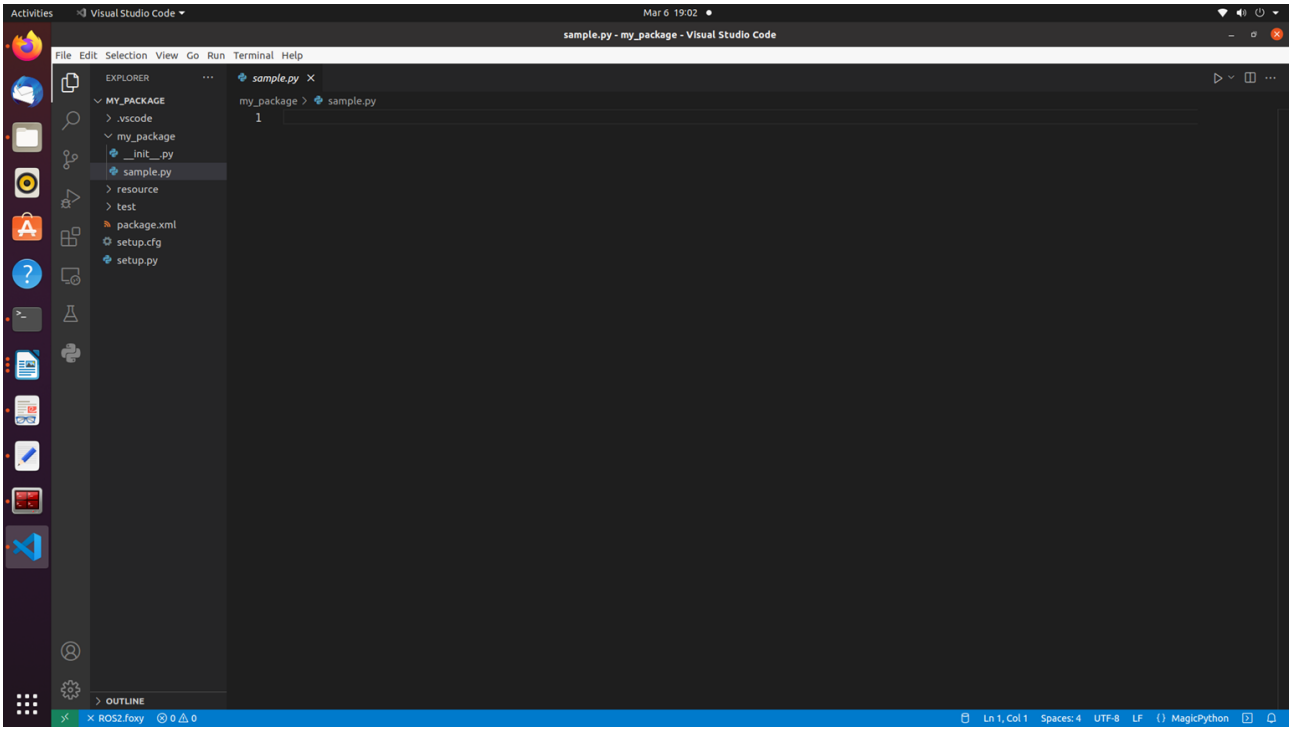
cd ~/ros2_ws
colcon build
colcon build --packages-select <package_name>
colcon build --packages-select my_package

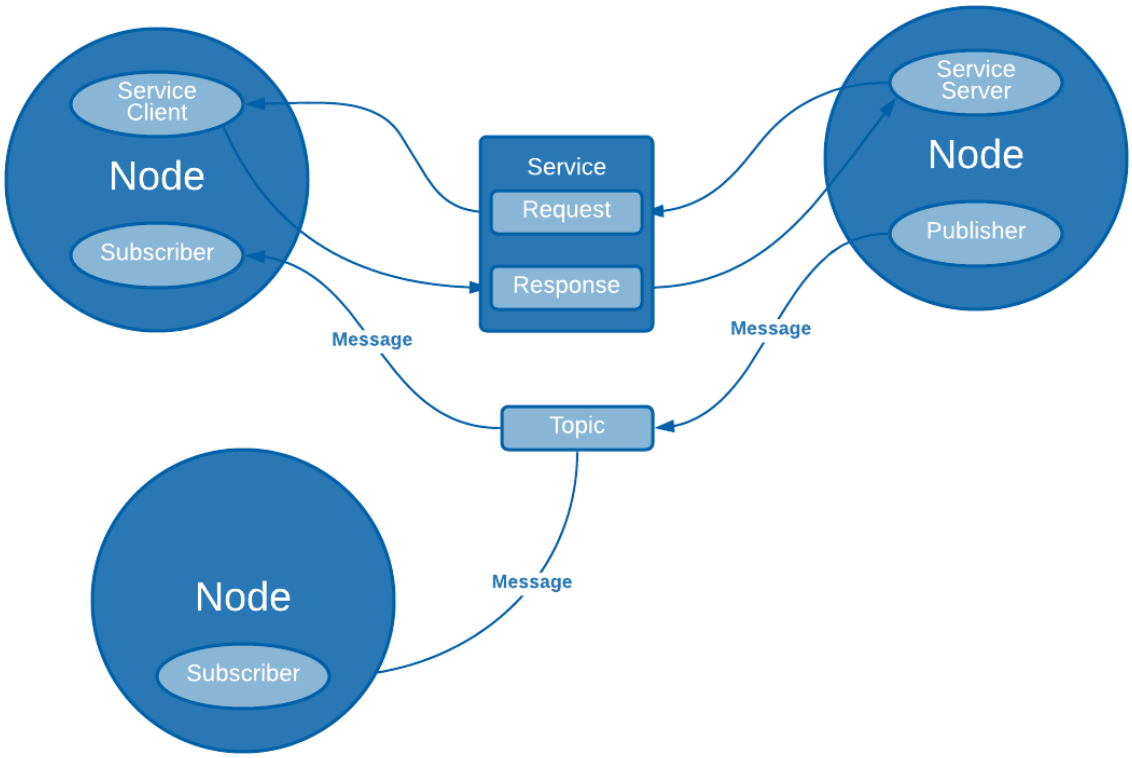
1. Create a Python file that will be executed in the **my_package** (all Python scripts) directory inside **my_package** folder.

Execute in Terminal #1

src/my_package/my_package
touch sample.py
chmod +x sample.py

Right click on the folder my_package and open with Visual Studio Application





Type the following code

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

class MyNode(Node):  #MIDIFY NAME OF THE CLASS
    def __init__(self):
        # call super() in the constructor in order to initialize the Node object
        # the parameter we pass is the node name
        super().__init__('sample') #MIDIFY NAME OF THE NODE
        # create a timer sending two parameters:
        # - the duration between 2 callbacks (0.2 seeconds)
        # - the timer function (timer_callback)
        self.create_timer(0.2, self.timer_callback)

    def timer_callback(self):
        # print a ROS2 log on the terminal with a great message!
        self.get_logger().info("Congratulation for starting your Robot Operating Syatem
Lab!!")


def main(args=None):
    # initialize the ROS communication
    rclpy.init(args=args)
    # declare the node constructor
    node = MyNode() #MIDIFY NAME OF THE NODE
    # pause the program execution, waits for a request to kill the node (ctrl+c)
    rclpy.spin(node)
    # shutdown the ROS communication
    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

Execute in Terminal #2

```
cd ~/ros2_ws/src/my_package
mkdir launch
cd launch
touch my_package_launch_file.launch.py
chmod +x my_package_launch_file.launch.py
```

Type the following in the my_package_launch_file.launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node
```

```
def generate_launch_description():
    return LaunchDescription([
        Node(
            package='my_package',
            executable='sample',
            output='screen'),
    ])
```

Modify the setup.py file to generate an executable from the Python file you just created.

```
from setuptools import setup
from glob import glob
import os

package_name = 'my_package'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launch/*')),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='asha',
    maintainer_email='asha@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        'sample = my_package.sample:main'
        ],
    },
)
```

The main objective of this code is to add an entry point to the script you created a few moments ago. To do that, work with a dictionary named entry_points. Inside it, you find an array called console_scripts. Add the node information to generate the executable. The objective of this code is to install the launch files. For example, with the package named "my_package", this will install all the launch files from the launch/folder, into ~/ros2_ws/install/my_python_pkg/share/my_python_pkg/launch/.

```
import os
from glob import glob
```

```
from setuptools import setup
package_name = 'my_package'
setup(

    #code
    ...


    data_files=[
            ('share/ament_index/resource_index/packages',
                ['resource/' + package_name]),
            ('share/' + package_name, ['package.xml']),
            (os.path.join('share', package_name), glob('launch/*.launch.py'))
        ],

    #code

)
```

Compile your package file as was previously explained.

Execute in Terminal #1

```
cd ~/ros2_ws
colcon build
source ~/ros2_ws/install/setup.bash
```

Finally, Now you must execute it. Use the following command that you already know. Execute the roslaunch command in the terminal to launch your program.

Execute in Terminal #1

```
ros2 launch my_package my_package_launch_file.launch.py
```

```
ctrl+C
```

Execute in Terminal #1

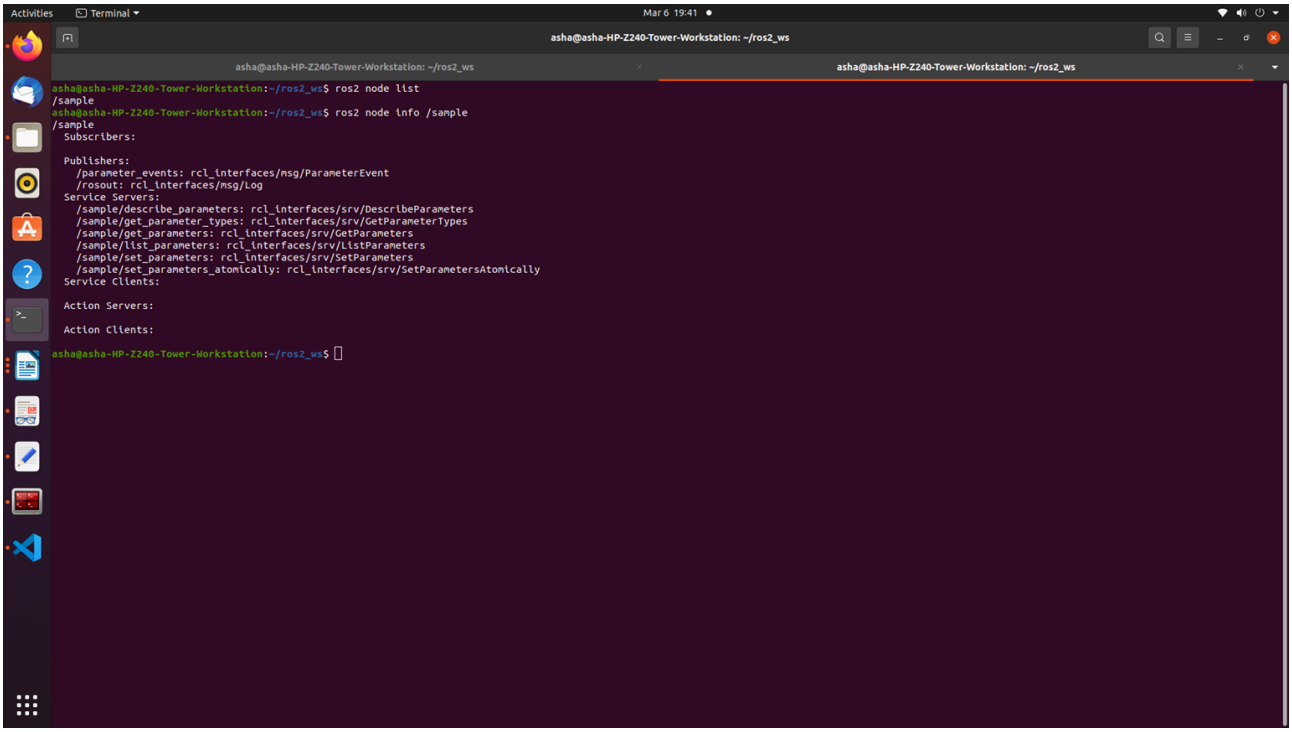
```
ros2 run  my_package sample
```

In main function declare the MyNode class, which you will use to start the node as such. Within this, you have the __init__method and the timer_callback method.
super().__init__('<node_name>')
timer_callback method creates a message with the counter value appended and publishes it to the console with get_logger().info.
self.get_logger().info(""your message to be printed")

Execute in Terminal #2

```
ros2 node list
```

ros2 node info <node_name>
ros2 node info /sample



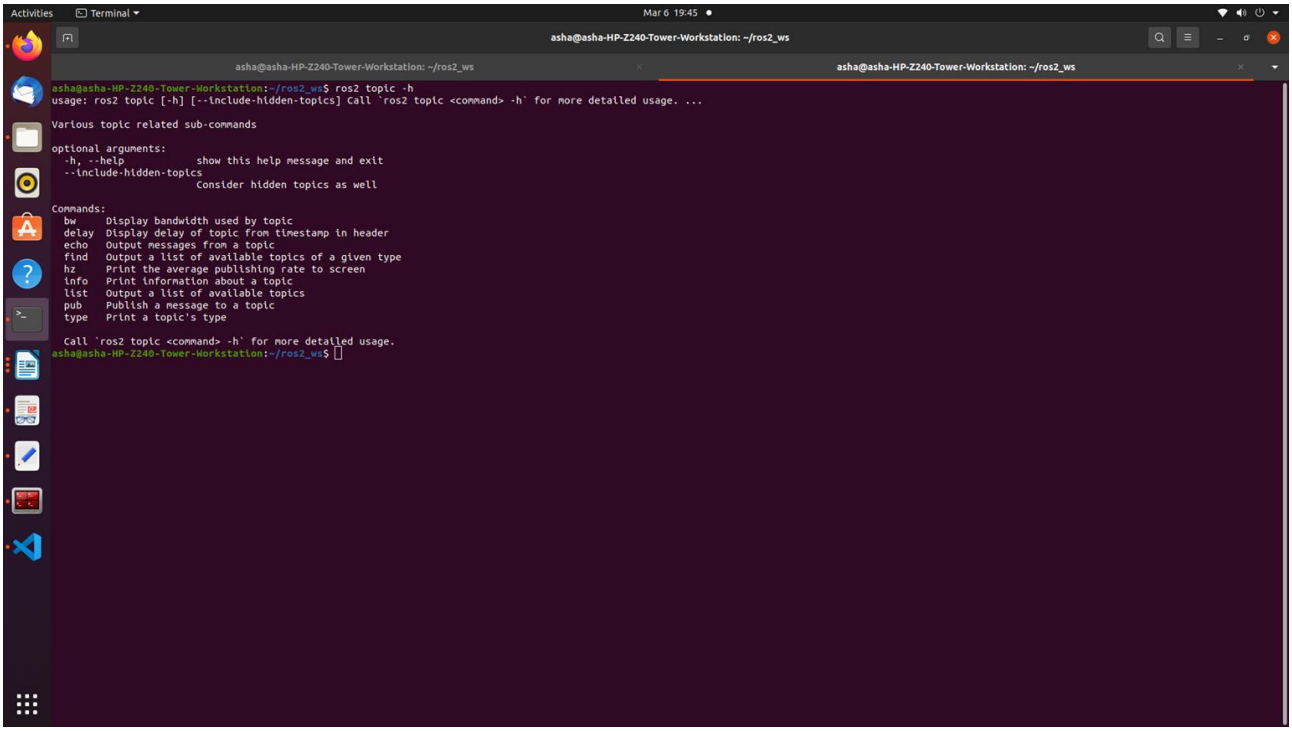
Understanding ROS2 Topics: Publishers & Subscribers

What will you learn about in this unit?

- Topics
- Basic topic commands
- Topic publishers
- Topic subscribers

Execute in Terminal #1

source /opt/ros/foxy/setup.bash
ros2 topic -h



OOP Python Code Template for Nodes

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node


class MyCustomNode(Node): # MODIFY NAME
        def __init__(self):
                super().__init__("node_name") # MODIFY NAME


def main(args=None):
        rclpy.init(args=args)
        node = MyCustomNode() # MODIFY NAME
        rclpy.spin(node)
        rclpy.shutdown()

if __name__ == "__main__":
            main()
```

Execute in Terminal #1

```
cd ros2_ws/src/my_package/my_package/
touch robot_publisher.py
chmod +x robot_publisher.py
ros2 interface show example_interfaces/msg/String
```

open the my_package using Visual Studio and edit the file robot_publisher.py

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.msg import String

class RobotPublisher(Node):

    def __init__(self):
        super().__init__("robot_publisher")
        self.robot_name_="ROBOT"
        self.publisher_ = self.create_publisher(String, "robot_news", 10)
        self.timer_ = self.create_timer(0.5, self.publish_news)
        self.get_logger().info("Node Started")


    def publish_news(self):
        msg = String()
        msg.data = "Hello " + str(self.robot_name_)
        self.publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = RobotPublisher()
    rclpy.spin(node)
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Edit the setup.py

```
from setuptools import setup

package_name = 'my_package '

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
```

```
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='asha',
    maintainer_email='asha@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
             'sample = my_package.sample:main',
            'robot_publisher = my_package.robot_publisher:main'
        ],
    },
)
```

Execute in Terminal #1

```
colcon build --packages-select my_package –symlink-install
```

Execute in Terminal #2

```
source ~/.bashrc
ros2 run my_package robot_publisher
```

Execute in Terminal #3

```
source ~/.bashrc
ros2 topic echo /robot_news
```

Subscriber node

Execute in Terminal #1

```
cd ros2_ws/src/my_package/my_package /
touch robot_subscriber.py
chmod +x robot_subscriber.py
```

Edit the file robot_subscriber.py using Visual Studio Editor

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.msg import String

class RobotSubscriber(Node):
    def __init__(self):
        super().__init__("robot_subscriber")
        self.subscriber_ = self.create_subscription(String, "robot_news",
self.callback_robot_news, 10)
        self.get_logger().info("robot_subscriber Node Started")
```

```
    def callback_robot_news(self, msg):
        self.get_logger().info(msg.data)


def main(args=None):
    rclpy.init(args=args)
    node = RobotSubscriber()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
    main()
```

Edit the setup.py

```
from setuptools import setup
package_name = 'my_package '

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='asha',
    maintainer_email='asha@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
             'sample = my_package.sample:main',
            "robot_publisher = my_package.robot_publisher:main",
            'robot_subscriber = my_package.robot_subscriber:main'
        ],
    },
)
```

Execute in Terminal #1

```
colcon build --packages-select my_package  --symlink-install
```

Execute in Terminal #2

```
source ~./bashrc
ros2 run my_package robot_publisher
```

Execute in Terminal #3

```
source ~./bashrc
ros2 run my_package robot_subscriber
```

Execute in Terminal #4

```
ros2 node list
ros2 topic list
```

Try: Modify the subscriber code to publish number at 1Hz on the topic /number

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.msg import String, Int32
global count

class RobotSubscriber(Node):
    def __init__(self):
        super().__init__("robot_subscriber")
        self.count_ = 0
        self.subscriber_ = self.create_subscription(String, "robot_news",
self.callback_robot_news, 10)
        self.publisher_ = self.create_publisher(Int32, "robot_number", 10)
        self.timer_ = self.create_timer(1, self.send_number)
        self.get_logger().info("robot_subscriber and publisher Node Started")

    def callback_robot_news(self, msg):
        self.get_logger().info(msg.data)

    def send_number(self):
        number = Int32()
        number.data = self.count_
        self.count_ +=1
        self.publisher_.publish(number)

def main(args=None):
    rclpy.init(args=args)
    node = RobotSubscriber()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
    main()
```

Post Lab Exercises - Marks: 4 [CO – 1, LO – 1, 2, 12, PO- 1,2,3, BL – 3,4,5]

Exercise 1: Write a launch file pub_sub.launch.py to run the publisher and subscriber node.
Exercise 2: Create 2 nodes from scratch. First node has 1 publisher, the second has 1 publisher & 1 subscriber.

- The number_publisher node publishes a number on the "/number" topic, with the existing type example_interfaces/msg/Int32.
- The number_counter node subscribes to the "/number" topic. It keeps a counter variable. Every time a new number is received, it's added to the counter. The node also has a publisher on the "/number_count" topic. When the counter is updated, the publisher directly publishes the new value on the topic.

A few hints:

- Check what to put into the example_interfaces/msg/Int32 with the "ros2 interface show" command line tool.
- Use the order as follows: first create the number_publisher node, check that the publisher is working with "ros2 topic". Then create the number_counter, focus on the subscriber. And finally create the last publisher. In the number_counter node, the publisher will publish messages directly from the subscriber callback.

More About Launch File

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        launch_ros.actions.Node(
            package='turtlebot3_teleop',
            executable='teleop_keyboard',
            output='screen'),
    ])
```

```
from launch import LaunchDescription
import launch_ros.actions

def generate_launch_description():
    return LaunchDescription([
        launch_ros.actions.Node(
            package='teleop_twist_keyboard',
            executable='teleop_twist_keyboard',
```

```
        output='screen'),
    ])
```

Within the LaunchDescription object, generate a node where you will provide the following parameters:

1. package='package_name' Name of the package that contains the code of the ROS program to execute
2. executable='cpp_executable_name' Name of the cpp executable file that you want to execute
3. output='type_of_output' Through which channel you will print the output of the program

Create Custom Message

To publish the data that contains multiple data types, one can create a new one.
Create a custom interface in a CMake package and then use it in a Python node.
To create a new message, do the following:

.1 Create a directory in the src folder
.2 Create a directory named **msg** inside your package Inside the directory, create a file named Name_of_message.msg. Modify the CMakeLists.txt file
.3 Modify package.xml file
.4 Compile and source
.5 Use in code

Create an interface to send the Manufacture date of the robot.

Execute in Terminal #1

```
cd ros2_ws/src
ros2 pkg create my_robot_interface
ls
cd my_robot_interface/
rm -rf include/
rm -rf src/
mkdir msg
cd msg
touch ManufactureDate.msg
```

open src with visual studio application:

Enter the following data in ManufactureDate.msg (**It should have the pattern '^[A-Z][A-Za-z0-9]*$')

```
int32 date
string month
int64 year
```

In CmakeLists.txt

Edit two functions inside CMakeLists.txt:

find_package()
This is where all the packages required to COMPILE the messages for the topics, services, and actions go. In package.xml, state them as **build_depend** and **exec_depend**.
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)
find_package(rosidl_default_generators REQUIRED)

rosidl_generate_interfaces()
This function includes all of the messages for this package (in the msg folder) to be compiled. The function should look similar to the following:
rosidl_generate_interfaces(${PROJECT_NAME}
"msg/ManufactureDate.msg"
)

```
cmake_minimum_required(VERSION 3.5)
project(my_robot_interface)

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES
"Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)
find_package(rosidl_default_generators REQUIRED)

rosidl_generate_interfaces(my_robot_interface
"msg/ManufactureDate.msg"
)


ament_package()
```

Modify package.xml

Add the following lines to the package.xml
<build_depend>rosidl_default_generators</build_depend>
<exec_depend>rosidl_default_runtime</exec_depend>
<member_of_group>rosidl_interface_packages</member_of_group>

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_robot_interface</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="asha@todo.todo">asha</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <depend>rclcpp</depend>
  <depend>std_msgs</depend>
  <build_depend>rosidl_default_generators</build_depend>
  <exec_depend>rosidl_default_runtime</exec_depend>
  <member_of_group>rosidl_interface_packages</member_of_group>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

Execute in Terminal #1

```
cd ~/ros2_ws
colcon build --packages-select my_robot_interface
cd install/my_robot_interface/lib/python3.8/site-packages/my_robot_interface/msg
```

This executes this bash file that sets, among other things, the newly generated messages created through the colcon build. If you don't do this, it might give you an import error, saying it doesn't find the message generated.

```
source install/setup.bash
ros2 interface show my_robot_interface/msg/ManufactureDate
```

Modify robot_publisher.py to transmit the manufacturing date to the subscriber node

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from my_robot_interface.msg import ManufactureDate

class RobotDatePublisher(Node):

    def __init__(self):
        super().__init__("robot_date_publisher")
        self.robot_name_="ROBOT"
        self.publisher_ = self.create_publisher(ManufactureDate,
"robot_manufacturing_date", 10)
        self.timer_ = self.create_timer(0.5, self.publish_news)
        self.get_logger().info("Node Started")


    def publish_news(self):
        msg = ManufactureDate()
        msg.date = 12
        msg.month = "March"
        msg.year = 2022
        self.publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = RobotDatePublisher()
    rclpy.spin(node)
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Execute in Terminal #1

```
colcon build --packages-select my_package
```

Execute in Terminal #2

```
ros2 run my_package robot_publisher
```

Execute in Terminal #3

```
ros2 run my_package robot_subscriber
```

Edit package.xml

```
 <depend>rclpy</depend>
 <depend>example_interfaces</depend>
 <depend>my_robot_interface</depend>
```

Edit robot_subscriber.py code

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from my_robot_interface.msg import ManufactureDate

class RobotDateSubscriber(Node):
    def __init__(self):
        super().__init__("robot_date_subscriber")
        self.subscriber_ = self.create_subscription(ManufactureDate,
"robot_manufacturing_date", self.callback_robot_news, 10)
        self.get_logger().info("robot_subscriber Node Started")

    def callback_robot_news(self, msg):
        information ="Manufacturing Date of the ROBOT is " + str(msg.date) + " " +
str(msg.month) + " " + str(msg.year)
        self.get_logger().info(information)


def main(args=None):
    rclpy.init(args=args)
    node = RobotDateSubscriber()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
    main()
```

Experiment 3: TurtleSim Programming, Publisher, Subscriber, Services, Actions

Recap:

Setup for launch files:

- Create a new package
- Create a launch/ folder at the root of the package.
- Configure CMakeLists.txt to install files from this launch/ folder.
- Create any number of files you want inside the launch/ folder, ending with .launch.py.

Run a launch file:

- use “colcon build” to install the file.
- source your environment
- launch file with “ros2 launch <package> <name_of_the_file>

First try to design the application by yourself. Don’t write code! Just take a piece of paper and make the design. What nodes should you create? How do the nodes communicate between each other? Which functionality should you add, and where to put them? Etc.

- Directly start on your own (Use the template to start with)
- Work step by step on each functionality/communication.

Client – Server Nodes

Execute in Terminal #1
ros2 interface show example_interfaces/srv/AddTwoInts

Execute in Terminal #1
```
cd ros2_ws/src/my_package/my_package
touch add_two_ints_server.py
chmod +x add_two_ints_server.py
```

Edit add_two_ints_server.py in visual studio editor

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
```

```
from example_interfaces.srv import AddTwoInts

class AddTwoIntsServerNode(Node):
    def __init__(self):
        super().__init__("add_two_ints_server")
        self.server_ = self.create_service(AddTwoInts, "add_two_ints",
self.callback_add_two_ints)
        self.get_logger().info("Add two ints server has been started")

    def callback_add_two_ints(self, request, response):
        response.sum = request.a + request.b
        self.get_logger().info(str(request.a)+ " + " + str(request.b) + " = " +
str(response.sum))
        return response

def main(args=None):
        rclpy.init(args=args)
        node = AddTwoIntsServerNode()
        rclpy.spin(node)
        rclpy.shutdown()

if __name__ == "__main__":
          main()
```

Add executable name in setup.py

```
 entry_points={
        'console_scripts': [
        'sample = my_package.sample:main',
        'robot_publisher = my_package.robot_publisher:main',
        'robot_subscriber = my_package.robot_subscriber:main',
        'add_two_ints_server = my_package.add_two_ints_server:main'
        ],
```

Execute in Terminal #1

```
colcon build –packages-select my_package
```

Execute in Terminal #2

```
ros2 run my_package add_two_ints_server
```

Execute in Terminal #3

ros2 service call /add_two_ints example_interfaces/srv/AddTwoInts "{a: 3, b: 4}"

Ctrl + C in all terminal windows.

Execute in Terminal #1

```
cd ros2_ws/src/my_package/my_package/
touch add_two_ints_client.py
chmod +x add_two_ints_client.py
```

Edit add_two_ints_client.py using visual studio editor

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.srv import AddTwoInts
from functools import partial


class AddTwoIntClientNode(Node):
    def __init__(self):
        super().__init__("add_two_ints_client")
        self.call_add_two_int_server(6, 7)

    def call_add_two_int_server(self, a, b):
        client = self.create_client(AddTwoInts, "add_two_ints")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server Add Two Ints")
        request = AddTwoInts.Request()
        request.a = a
        request.b = b
        future = client.call_async(request)
        future.add_done_callback(
            partial(self.callback_call_two_ints, a=a, b=b))

    def callback_call_two_ints(self, future, a, b):
        try:
            response = future.result()
            self.get_logger().info(str(a) + " + " + str(b) + " = " + str(response.sum))

        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e,))


def main(args=None):
    rclpy.init(args=args)
    node = AddTwoIntClientNode()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
```

```
    main()
```

Add executable name in setup.py

```
entry_points={
        'sample = my_package.sample:main',
        'robot_publisher = my_package.robot_publisher:main',
        'robot_subscriber = my_package.robot_subscriber:main',
        'add_two_ints_server = my_package.add_two_ints_server:main',
        'add_two_ints_client = my_package.add_two_ints_client:main'
        ],
```

Execute in Terminal #1

```
colcon build --packages-select my_package –symlink-install
```

Execute in Terminal #2

```
ros2 run my_package add_two_int_server
```

Execute in Terminal #3

```
ros2 run my_package add_two_ints_client
```

Execute in Terminal #4

```
ros2 node list
```

Execute in Terminal #5

```
ros2 service list
ros2 service type /add_two_ints
ros2 interface show example_interfaces/srv/AddTwoInts
ros2 service call /add_two_int example_interfaces/srv/AddTwoInts
ros2 service call /add_two_int example_interfaces/srv/AddTwoInts "{a: 3, b: 4}"
rqt
plugins→services→service caller
service - /add_two_ints
Enter the values under Expression for a and b
Click call
```

Response is viewed in the second window

Exercise 1: Create a service-client operation to reset the counter value in the number_counter nodes.

The node “number_publisher” publishes a number on the /”number” topic.

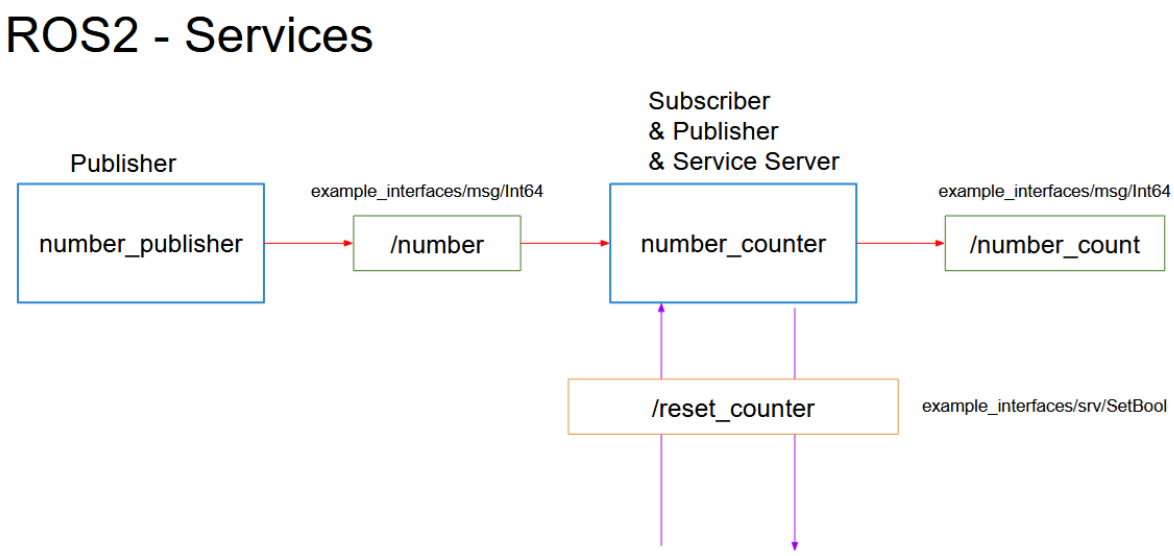
The node “number_counter” gets the number, adds it to a counter, and publishes the counter on the “/number_count” topic.

Add the following ros2 services

Add a functionality to reset the counter to zero:

- Create a service server inside the “number_counter” node.
- Service name: “/reset_counter”
- Service type: example_interfaces/srv/SetBool. Use “ros2 interface show” to discover what’s inside!
- When the server is called, you check the boolean data from the request. If true, you set the counter variable to 0.

We will then call the service directly from the command line. You can also decide - for more practice - to create your own custom node to call this “/reset_counter” service.



Add a functionality to reset the counter to zero:

- Create a service server inside the “number_counter” node.
- Service name: “/reset_counter”
- Service type: example_interfaces/srv/SetBool. Use “ros2 interface show” to discover what’s inside!
- When the server is called, you check the boolean data from the request. If true, you set the counter variable to 0.

We will then call the service directly from the command line. You can also decide - for more practice - to create your own custom node to call this “/reset_counter” service.

```
#!/usr/bin/env python3
```

```
import rclpy
from rclpy.node import Node
from example_interfaces.msg import Int64
from example_interfaces.srv import SetBool

class NumberCounterNode(Node):
    def __init__(self):
        super().__init__("number_counter")
        self.counter_ = 0
        self.number_count_publisher_ = self.create_publisher(Int64, "number_count", 10)
        self.number_subscriber_ = self.create_subscription(Int64, "number",
self.callback_number, 10)
        self.reset_counter_service_ = self.create_service(SetBool, "reset_counter",
self.callback_reset_counter)
        self.get_logger().info("Node started")


    def callback_number(self, msg):
        self.counter_ += msg.data
        new_msg = Int64()
        new_msg.data = self.counter_
        self.number_count_publisher_.publish(new_msg)
        self.get_logger().info(str(self.counter_))

    def callback_reset_counter(self, request, response):
        if request.data:
            self.counter_ = 0
            response.success = True
            response.message = "Counter is reset"
        else:
            response.success = False
            response.message = "Counter is not reset"
        return response


def main(args=None):
        rclpy.init(args=args)
        node = NumberCounterNode()
        rclpy.spin(node)
        rclpy.shutdown()

if __name__ == "__main__":
            main()
```

Execute in Terminal #1

```
ros2 interface show example_interfaces/srv/SetBool
```

Execute in Terminal #1

```
cd ros2_ws/
colcon build --packages-select my_package
```

Execute in Terminal #1
ros2 run my_package number_publisher

Execute in Terminal #2
ros2 run my_package number_counter

Execute in Terminal #3
ros2 topic list
ros2 topic echo /number_count

Execute in Terminal #4
ros2 service call /reset_counter example_interfaces/srv/SetBool "{data: False}"
ros2 service call /reset_counter example_interfaces/srv/SetBool "{data: True}"

Custom Services

cd ros2_ws/src/my_robot_interface
mkdir srv
cd srv
touch SetDate.srv

SetDate.srv

string robot_name
int64 date
---
bool success

Change CmakeLists.txt as

rosidl_generate_interfaces(my_robot_interface
"msg/ManufactureDate.msg"
"srv/SetDate.srv"
)

colcon build –packages-select my_robot_interface

Change robot_publisher.py code as

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from my_robot_interface.msg import ManufactureDate
from my_robot_interface.srv import SetDate

class RobotDatePublisher(Node):

    def __init__(self):
        super().__init__("robot_date_publisher")
```

```
        self.robot_name_="ROBOT"
        self.publisher_ = self.create_publisher(ManufactureDate,
"robot_manufacturing_date", 10)
        self.timer_ = self.create_timer(0.5, self.publish_news)
        self.set_date_ = self.create_service(SetDate, "set_date", self.callback_set_date)
        self.get_logger().info("Node Started")

    def callback_set_date(self, request, response):
        name = request.robot_name
        date = request.date

        if (name =="ROBOT") and (date==12):
            response.success = True
        else:
            response.success = False
        return response

    def publish_news(self):
        msg = ManufactureDate()
        msg.date = 12
        msg.month = "March"
        msg.year = 2022
        self.publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = RobotDatePublisher()
    rclpy.spin(node)
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Execute in Terminal #1

```
colcon build –packages-select my_package
```

Execute in Terminal #1

```
ros2 run my_package robot_publisher
```

Execute in Terminal #2

```
ros2 service list
ros2 service call /set_date my_robot_interface/srv/SetDate "{name: "ROBOT", date: 12}"
```

change robot_subscriber.py code as

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.msg import String, Int32
```

```
from my_robot_interface.msg import ManufactureDate
from my_robot_interface.srv import SetDate
from functools import partial

class RobotDateSubscriber(Node):
    def __init__(self):
        super().__init__("robot_date_subscriber")
        self.subscriber_ = self.create_subscription(ManufactureDate,
"robot_manufacturing_date", self.callback_robot_news, 10)
        self.get_logger().info("robot_subscriber Node Started")

    def callback_robot_news(self, msg):
        information ="Manufacturing Date of the ROBOT is " + str(msg.date) + " " +
str(msg.month) + " " + str(msg.year)
        self.get_logger().info(information)
        self.check_date_server("ROBOT", 12)

    def check_date_server(self, robot_name, date):
        client = self.create_client(SetDate, "set_date")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server")
        request = SetDate.Request()
        request.robot_name = robot_name
        request.date = date
        future = client.call_async(request)
        future.add_done_callback(partial(self.callback_date_response,
robot_name=robot_name, date=date))

    def callback_date_response(self, future, robot_name, date):
        try:
            response = future.result()
            self.get_logger().info(str(response.success))
        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e,))

def main(args=None):
    rclpy.init(args=args)
    node = RobotDateSubscriber()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
    main()
```

Execute in Terminal #1

```
ros2 run colcon build --packages-select my_package
```

Execute in Terminal #2

```
ros2 run my_package robot_publisher
```

Execute in Terminal #3

ros2 run my_package robot_subscriber

TurtleSim Programming

Execute in Terminal #1

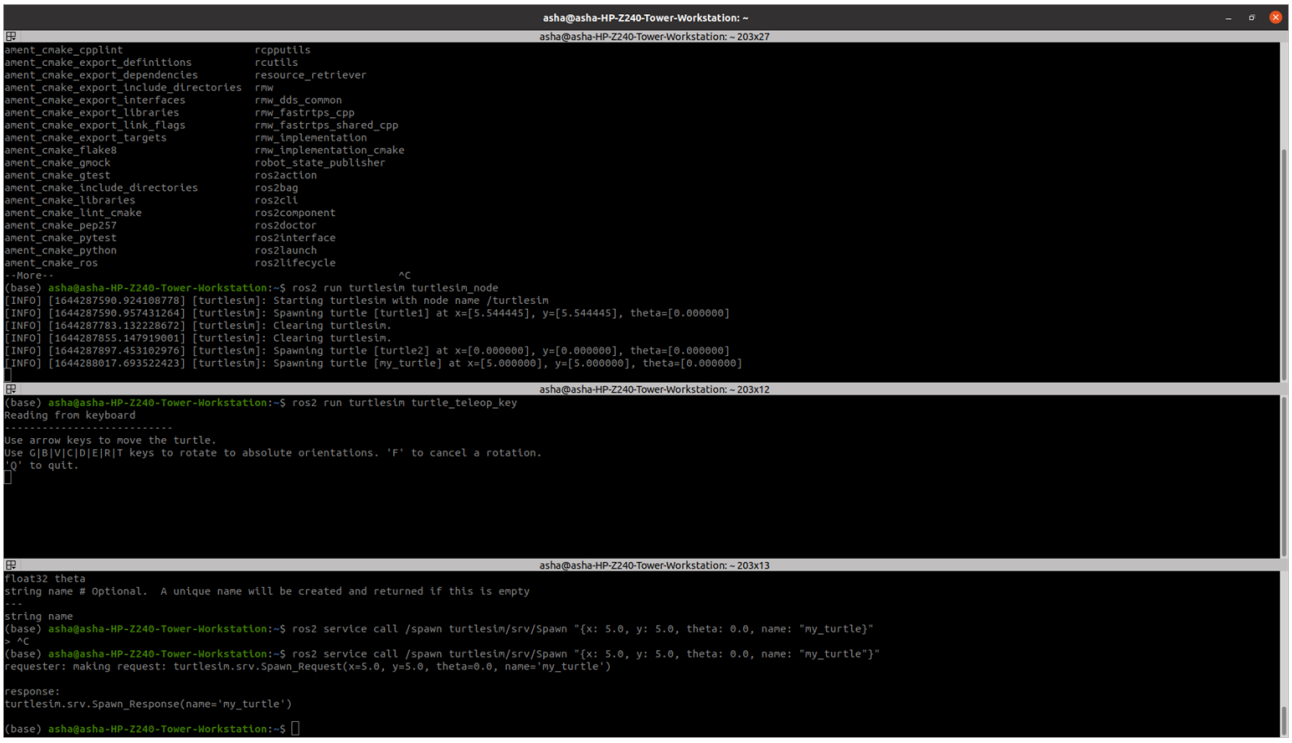
ros2 run turtlesim turtlesim_node

Execute in Terminal #2

ros2 run turtlesim turtle_teleop_key

Execute in Terminal #3

ros2 service list
ros2 service type /clear
ros2 interface show std_srvs/srv/Empty
ros2 service call /clear std_srvs/srv/Empty
ros2 service type /spawn
ros2 interface show turtlesim/srv/Spawn
ros2 service call /spawn turtlesim/srv/Spawn
ros2 service call /spawn turtlesim/srv/Spawn "{x: 5.0, y: 5.0, theta: 0.0, name: "my_turtle"}"



ROS2 interfaces:

https://github.com/ros2/example_interfaces
https://github.com/ros2/common_interfaces

You will use 3 nodes:

- The turtlesim_node from the turtlesim package

- A custom node to control the turtle (named “turtle1”) which is already existing in the turtlesim_node. This node can be called turtle_controller.
- A custom node to spawn turtles on the window. This node can be called turtle_spawner.

Execute in Terminal #1

```
cd ~/ros2_ws/src/my_package/my_package
```

Execute in Terminal #2

```
touch turtle_controller.py
chmod + turtle_controller.py
```

Open src with Visual Studio Application

Enter the code in turtle_controller.py

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from turtlesim.msg import Pose
from geometry_msgs.msg import Twist
import math


class TurtleControllerNode(Node):
    def __init__(self):
        super().__init__("turtle_controller")
        self.target_x = 8.0
        self.target_y = 4.0
        self.pose_ = None
        self.cmd_vel_publisher_ = self.create_publisher(Twist, "turtle1/cmd_vel", 10)
        self.pose_subscriber_ = self.create_subscription(Pose, "turtle1/pose",
self.callback_turtle_pose, 10)
        self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)

    def callback_turtle_pose(self,msg):
        self.pose_ = msg


    def control_loop(self):
        if self.pose_ == None:
            return
        dist_x = self.target_x - self.pose_.x
```

```
        dist_y = self.target_y - self.pose_.y
        distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)
        msg = Twist()
        if distance > 0.5:
            msg.linear.x = distance
            goal_theta = math.atan2(dist_y, dist_x)
            diff = goal_theta - self.pose_.theta

            if diff > math.pi:
                diff -= 2*math.pi
            elif diff < -math.pi:
                diff += 2*math.pi

            msg.angular.z = diff
        else:
            msg.linear.x = 0.0
            msg.angular.z = 0.0

        self.cmd_vel_publisher_.publish(msg)


def main(args=None):
    rclpy.init(args=args)
    node = TurtleControllerNode()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
        main()
```

Modify entry_points setup.py as

```
entry_points={
        'console_scripts': [
        'sample = my_package.sample:main',
        'robot_publisher = my_package.robot_publisher:main',
        'robot_subscriber = my_package.robot_subscriber:main',
        'turtlesim_controller = my_package.turtle_controller:main'
        ],
    },
```

)

Modify he package.xml as

```
  <depend>rclpy</depend>
  <depend>example_interfaces</depend>
  <depend>my_robot_interface</depend>
  <depend>turtlesim</depend>
```

Execute in Terminal #1

```
ros2 run turtlesim turtlesim_node
```

Execute in Terminal #2

```
colcon build --packages-select my_package --symlink-install
```

Execute in Terminal #3

```
ros2 run my_package turtlesim_controller
```

Execute in Terminal #4

```
ros2 service list
ros2 service type /spawn
ros2 interface show turtlesim/srv/Spawn
```

Execute in Terminal #1

```
cd ros2_ws/my_robot_interface/srv
touch  MoveLocation.srv
```

Edit MoveLocation.srv

```
float32 loc_x
float32 loc_y
---
float32 distance
```

Change CmakeLists.txt as

```
rosidl_generate_interfaces(my_robot_interface
"msg/ManufactureDate.msg"
"srv/SetDate.srv"
"srv/MoveLocation.srv"
)
```

Execute in Terminal #1

```
cd ~/ros2_ws
colcon build –packages-select my_robot_interface
```

Send the service request to find the distance between current location and new location.

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from turtlesim.msg import Pose
from geometry_msgs.msg import Twist
from my_robot_interface.srv import MoveLocation
import math


class TurtleControllerNode(Node):
    def __init__(self):
        super().__init__("turtle_controller")
        self.target_x = 9.0
        self.target_y = 9.0
        self.pose_ = None
        self.cmd_vel_publisher_ = self.create_publisher(Twist, "turtle1/cmd_vel", 10)
        self.pose_subscriber_ = self.create_subscription(Pose, "turtle1/pose",
self.callback_turtle_pose, 10)
        self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)
        self.servce_ = self.create_service(MoveLocation, "move_location",
self.callback_get_distance)
    def callback_turtle_pose(self,msg):
        self.pose_ = msg



    def control_loop(self):
        if self.pose_ == None:
            return
        dist_x = self.target_x - self.pose_.x
        dist_y = self.target_y - self.pose_.y
        distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)
        msg = Twist()
        if distance > 0.5:
            msg.linear.x = distance
            goal_theta = math.atan2(dist_y, dist_x)
            diff = goal_theta - self.pose_.theta

            if diff > math.pi:
                diff -= 2*math.pi
            elif diff < -math.pi:
                diff += 2*math.pi

            msg.angular.z = diff
        else:
            msg.linear.x = 0.0
```

```
            msg.angular.z = 0.0

        self.cmd_vel_publisher_.publish(msg)

    def callback_get_distance(self, request, response):
        x = request.loc_x - self.pose_.x
        y = request.loc_y - self.pose_.y

        response.distance = math.sqrt(x * x + y * y)
        return response


def main(args=None):
    rclpy.init(args=args)
    node = TurtleControllerNode()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
        main()
```

Execute in Terminal #1

```
colcon build –packages-select my_package
```

Execute in Terminal #2

```
ros2 run turtlesim turtlesim_node
```

Execute in Terminal #3

```
ros2 run my_package turtlesim_controller
```

Execute in Terminal #1

```
ros2 service call /move_location my_robot_interface/srv/MoveLocation "{loc_x: 5.0, loc_y: 5.0}"
```

Exercise2: Create two new files named movement_server.py and movement_client.py.

1. Create a directory named **srv** inside my_robot_interface package
2. Inside this directory, create a file named **MyCustomServiceMessage**.srv

```
string move  # Signal to define movement
             # "Turn right" to make the robot turn in right direction.
             # "Turn left" to make the robot turn in left direction.
             # "Stop" to make the robot stop the movement.
---
bool success
```

3. Modify CMakeLists.txt file
4. Modify package.xml file
5. Compile and source

6 Use in code

```
ros2 interface show my_robot_interface/srv/MyCustomServiceMessage
```

Action Server – Action Client Nodes

Execute in Terminal #1

```
cd ~/ros2_ws/src/my_robot_interface
mkdir action
touch Navigate2D.action

#Goal
int32 secs
---
#Result
string status
---
#Feedback
string feedback
```

package.xml

```
  <depend>rclcpp</depend>
  <depend>std_msgs</depend>
  <depend>action_msgs</depend>
```

CMakeLists.txt

```
rosidl_generate_interfaces(my_robot_interface
"msg/ManufactureDate.msg"
"srv/SetDate.srv"
"srv/MoveLocation.srv"
"action/Navigate2D.action"
)
```

Execute in Terminal #1

```
colcon build –packages-select my_robot_interface
```

Execute in Terminal #1

```
cd ~/ros2_ws/src/my_package/my_package
touch  action_client.py
chmod +x  action_client.py

import rclpy
from rclpy.action import ActionClient
from rclpy.node import Node
from rclpy.executors import MultiThreadedExecutor
from my_robot_interface.action import Navigate2D
```

```
class MyActionClient(Node):
    def __init__(self):
        super().__init__('action_client')
        self._action_client = ActionClient(self, Navigate2D, "navigate")

    def send_goal(self, secs):
        goal_msg = Navigate2D.Goal()
        goal_msg.secs = secs
        self._action_client.wait_for_server()
        self._send_goal_future = self._action_client.send_goal_async(goal_msg,
self.feedback_callback)
        self._send_goal_future.add_done_callback(self.goal_response_callback)

    def goal_response_callback(self, future):
        goal_handle = future.result()
        if not goal_handle.accepted:
            self.get_logger().info('Goal rejected')
            return
        self.get_logger().info('Goal accepted')
        self._get_result_future = goal_handle.get_result_async()
        self._get_result_future.add_done_callback(self.get_result_callback)

    def get_result_callback(self, future):
        result = future.result().result
        self.get_logger().info('Result: {0}'.format(result.status))
        rclpy.shutdown()

    def feedback_callback(self, feedback_msg):
        feedback = feedback_msg.feedback
        self.get_logger().info('Received feedback: {0}'.format(feedback.feedback))

def main(args=None):
    rclpy.init(args=args)
    action_client = MyActionClient()
    future = action_client.send_goal(5)
    executor = MultiThreadedExecutor()
    rclpy.spin(action_client, executor=executor)

if __name__ == '__main__':
    main()
```

Execute in Terminal #1

```
cd ~/ros2_ws/src/my_package/my_package
touch  action_server.py
chmod +x  action_server.py
```

Edit the file action_server.py

```
#!/usr/bin/env python3
```

```
import rclpy
from rclpy.node import Node
from turtlesim.msg import Pose
from geometry_msgs.msg import Twist
from rclpy.action import ActionServer
import time
from my_robot_interface.action import Navigate2D


class NavigateAction(Node):
    def __init__(self):
        super().__init__("action_server")
        self.action_server_ = ActionServer(
            self, Navigate2D, "navigate", self.navigate_callback)
        self.cmd = Twist()
        self.publisher_ = self.create_publisher(Twist, "turtle1/cmd_vel", 10)

    def navigate_callback(self, goal_handle):
        self.get_logger().info('Executing goal...')
        feedback_msg = Navigate2D.Feedback()
        feedback_msg.feedback = "Moving to the left ..."
        for i in range(1, goal_handle.request.secs):
            self.get_logger().info(feedback_msg.feedback)
            goal_handle.publish_feedback(feedback_msg)
            self.cmd.linear.x = 0.3
            self.cmd.angular.z = 0.3
            self.publisher_.publish(self.cmd)
            time.sleep(1)
        goal_handle.succeed()
        self.cmd.linear.x = 0.0
        self.cmd.angular.z = 0.0
        self.publisher_.publish(self.cmd)
        feedback_msg.feedback = "Finished action server. Robot moved during 5 seconds"
        result = Navigate2D.Result()
        result.status = feedback_msg.feedback
        return result


def main(args=None):
    rclpy.init(args=args)
    node = NavigateAction()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
    main()
```

Edit CmakeLists.txt

```
  entry_points={
```

```
'console_scripts': [
'sample = my_package.sample:main',
'robot_publisher = my_package.robot_publisher:main',
'robot_subscriber = my_package.robot_subscriber:main',
'add_two_int_server = my_package.add_two_int_server:main',
'add_two_ints_client = my_package.add_two_ints_client:main',
'turtlesim_controller = my_package.turtle_controller:main',
'action_client = my_package.action_client:main',
'action_server = my_package.action_server:main'
],
```

Execute in Terminal #1

```
ros2 run turtlesim turtlesim_node
```

Execute in Terminal #2

```
ros2 run my_package action_client
```

Execute in Terminal #3

```
ros2 run my_package action_server
```

Introduction to URDF: Mobile Robot Design and Control

Install the necessary packages required
Execute in Terminal #1

```
sudo apt-get install ros-foxy-teleop-twist-keyboard
sudo apt-get install ros-foxy-joint-state-publisher*

sudo apt install gazebo11 libgazebo11-dev
sudo apt install ros-foxy-gazebo-ros-pkgs
sudo apt install ros-foxy-robot-state-publisher*
```

Execute in Terminal #1

```
cd ros2_ws/src
ros2 pkg create lab4 --build-type ament_python --dependencies rclpy
cd src/lab4/
mkdir urdf
mkdir launch
cd urdf
touch three_wheeled_robot.urdf
```

```
<?xml version="1.0" ?>

<robot name = "three_wheeled_robot">

    <link name="base">
        <visual>
        <geometry>
            <box size="0.75 0.4 0.1"/>
        </geometry>
        <material name="gray">
            <color rgba=".2 .2 .2 1" />
        </material>
        </visual>

        <inertial>
            <mass value="1" />
            <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01" iyz="0" izz="0.01" />
        </inertial>

        <collision>
        <geometry>
            <box size="0.75 0.4 0.1"/>
        </geometry>
        </collision>

    </link>

    <link name="wheel_right_link">
        <inertial>
                <mass value="2" />
                <inertia ixx="0.01" ixy="0.0" ixz="0"
                iyy="0.01" iyz="0" izz="0.01" />
        </inertial>

        <visual>
            <geometry>
              <cylinder radius="0.15" length="0.1"/>
            </geometry>
            <material name="white">
            <color rgba="1 1 1 1"/>
            </material>
        </visual>

        <collision>
            <geometry>
              <cylinder radius="0.15" length="0.1"/>
            </geometry>
          <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
        </collision>
    </link>
```

```
<joint name="wheel_right_joint" type="continuous">
    <origin xyz="0.2 0.25 0.0" rpy="1.57 0.0 0.0"/>
    <parent link="base"/>
    <child link="wheel_right_link"/>
    <axis xyz="0.0 0.0 1.0"/>
</joint>



<link name="wheel_left_link">
    <inertial>
        <mass value="2" />
        <inertia ixx="0.01" ixy="0.0" ixz="0"
            iyy="0.01" iyz="0" izz="0.01" />
    </inertial>

    <visual>
      <geometry>
          <cylinder radius="0.15" length="0.1"/>
      </geometry>
      <material name="white">
        <color rgba="1 1 1 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
          <cylinder radius="0.15" length="0.1"/>
      </geometry>
      <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
    </collision>
</link>

<joint name="wheel_left_joint" type="continuous">
    <origin xyz="0.2 -0.25 0.0" rpy="1.57 0.0 0.0"/>
    <parent link="base"/>
    <child link="wheel_left_link"/>
    <axis xyz="0.0 0.0 1.0"/>
</joint>



<link name="caster">
    <inertial>
            <mass value="1" />
            <inertia ixx="0.01" ixy="0.0" ixz="0"
            iyy="0.01" iyz="0" izz="0.01" />
    </inertial>

    <visual>
      <geometry>
```

```
      <sphere radius=".08" />
    </geometry>
    <material name="white" />
  </visual>

  <collision>
    <origin/>
    <geometry>
      <sphere radius=".08" />
    </geometry>
  </collision>
</link>

<joint name="caster_joint" type="continuous">
  <origin xyz="-0.3 0.0 -0.07" rpy="0.0 0.0 0.0"/>
  <axis xyz="0 0 1" />
  <parent link="base"/>
  <child link="caster"/>
</joint>

<link name="camera">
  <inertial>
        <mass value="0.1" />
        <inertia ixx="0.01" ixy="0.0" ixz="0"
        iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

  <visual>
    <geometry>
      <box size="0.1 0.1 0.05"/>
    </geometry>
    <material name="white">
      <color rgba="1 1 1 1"/>
    </material>
  </visual>

  <collision>
    <geometry>
       <box size="0.1 0.1 0.05"/>
    </geometry>
  </collision>
</link>

<joint name="camera_joint" type="fixed">
  <origin xyz="-0.35 0 0.01" rpy="0 0.0 3.14"/>
  <parent link="base"/>
  <child link="camera"/>
  <axis xyz="0.0 0.0 1.0"/>
</joint>

<link name="lidar">
```

```
    <inertial>
            <mass value="0.5" />
            <inertia ixx="0.01" ixy="0.0" ixz="0"
            iyy="0.01" iyz="0" izz="0.01" />
    </inertial>

    <visual>
      <geometry>
        <cylinder radius="0.1" length="0.05"/>
      </geometry>
      <material name="white">
        <color rgba="1 1 1 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
          <box size="0.1 0.1 0.1"/>
      </geometry>
    </collision>
</link>

<joint name="lidar_joint" type="fixed">
    <origin xyz="-0.285 0 0.075" rpy="0 0.0 1.57"/>
    <parent link="base"/>
    <child link="lidar"/>
    <axis xyz="0.0 0.0 1.0"/>

</joint>

<!--http://wiki.ros.org/simulator_gazebo/Tutorials/ListOfMaterials-->
<gazebo reference="base">
    <material>Gazebo/WhiteGlow</material>
</gazebo>
<gazebo reference="wheel_left_link">
    <material>Gazebo/SkyBlue</material>
</gazebo>
<gazebo reference="wheel_right_link">
    <material>Gazebo/SkyBlue </material>
</gazebo>
<gazebo reference="caster">
    <material>Gazebo/Grey</material>
</gazebo>
<gazebo reference="lidar">
    <material>Gazebo/Blue</material>
</gazebo>
<gazebo reference="camera">
    <material>Gazebo/Red</material>
</gazebo>
```

```
<!-- differential robot-->
<gazebo>
   <plugin filename="libgazebo_ros_diff_drive.so" name="gazebo_base_controller">
     <odometry_frame>odom</odometry_frame>
     <commandTopic>cmd_vel</commandTopic>
     <publish_odom>true</publish_odom>
     <publish_odom_tf>true</publish_odom_tf>
     <update_rate>15.0</update_rate>

     <left_joint>wheel_left_joint</left_joint>
     <right_joint>wheel_right_joint</right_joint>

     <wheel_separation>0.5</wheel_separation>
     <wheel_diameter>0.3</wheel_diameter>
     <max_wheel_acceleration>0.7</max_wheel_acceleration>
     <max_wheel_torque>8</max_wheel_torque>
     <robotBaseFrame>base</robotBaseFrame>
   </plugin>
</gazebo>

<!-- camera plugin-->
<gazebo reference="camera">
  <sensor type="camera" name="camera1">
    <visualize>true</visualize>
    <update_rate>30.0</update_rate>
    <camera name="head">
      <horizontal_fov>1.3962634</horizontal_fov>
      <image>
        <width>800</width>
        <height>800</height>
        <format>R8G8B8</format>
      </image>
      <clip>
        <near>0.02</near>
        <far>300</far>
      </clip>
    </camera>
    <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
      <alwaysOn>true</alwaysOn>
      <updateRate>60.0</updateRate>
      <cameraName>/camera1</cameraName>
      <imageTopicName>image_raw</imageTopicName>
      <cameraInfoTopicName>info_camera</cameraInfoTopicName>
      <frameName>camera</frameName>
      <hackBaseline>0.07</hackBaseline>
    </plugin>
  </sensor>
</gazebo>

<!--lidar plugin-->
```

```
    <gazebo reference="lidar">
      <sensor name="lidar" type="ray">
       <visualize>true</visualize>
        <update_rate>12.0</update_rate>
        <plugin filename="libgazebo_ros_ray_sensor.so" name="gazebo_lidar">
        <output_type>sensor_msgs/LaserScan</output_type>
        <frame_name>lidar</frame_name>
        </plugin>
        <ray>
          <scan>
            <horizontal>
              <samples>360</samples>
              <resolution>1</resolution>
              <min_angle>0.00</min_angle>
              <max_angle>3.14</max_angle>
            </horizontal>
          </scan>
          <range>
            <min>0.120</min>
            <max>3.5</max>
            <resolution>0.015</resolution>
          </range>
        </ray>
      </sensor>
    </gazebo>

</robot>
```

```
cd ..
cd launch
touch gazebo.launch.py
```

```
from launch import LaunchDescription
from launch_ros.actions import Node
from launch.actions import  ExecuteProcess
def generate_launch_description():
    urdf = '/home/asha/ros2_ws/src/lab4/urdf/three_wheeled_robot.urdf'
    return LaunchDescription([

        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            arguments=[urdf]),
        Node(
            package='joint_state_publisher',
            executable='joint_state_publisher',
            name='joint_state_publisher',
            arguments=[urdf]),
#  Gazebo related stuff required to launch the robot in simulation
```

```
        ExecuteProcess(
            cmd=['gazebo', '--verbose', '-s', 'libgazebo_ros_factory.so'],
            output='screen'),
        Node(
            package='gazebo_ros',
            executable='spawn_entity.py',
            name='urdf_spawner',
            output='screen',
            arguments=["-topic", "/robot_description", "-entity", "lab4 "])
    ])
```

touch rviz.launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    urdf = '/home/asha/ros2_ws/src/lab4/urdf/three_wheeled_robot.urdf'
    # rviz_config_file=os.path.join(package_dir,'config.rviz')

    return LaunchDescription([
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            arguments=[urdf]),
        Node(
            package='joint_state_publisher_gui',
            executable='joint_state_publisher_gui',
            name='joint_state_publisher_gui',
            arguments=[urdf]),

        Node(
        package='rviz2',
        executable='rviz2',
        name='rviz2',
        # arguments=['-d',rviz_config_file],
        output='screen'),


    ])
```

Edit the setup.py file as

```
from setuptools import setup
import os
from glob import glob
package_name = 'lab4 '
```

```
setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
          (os.path.join('share', package_name), glob('urdf/*')),
          (os.path.join('share', package_name), glob('launch/*'))
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='asha',
    maintainer_email='asha@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        ],
    },
)
```

Execute in Terminal #1

```
colcon build --packages-select lab4
ros2 launch lab4 rviz.launch.py
```

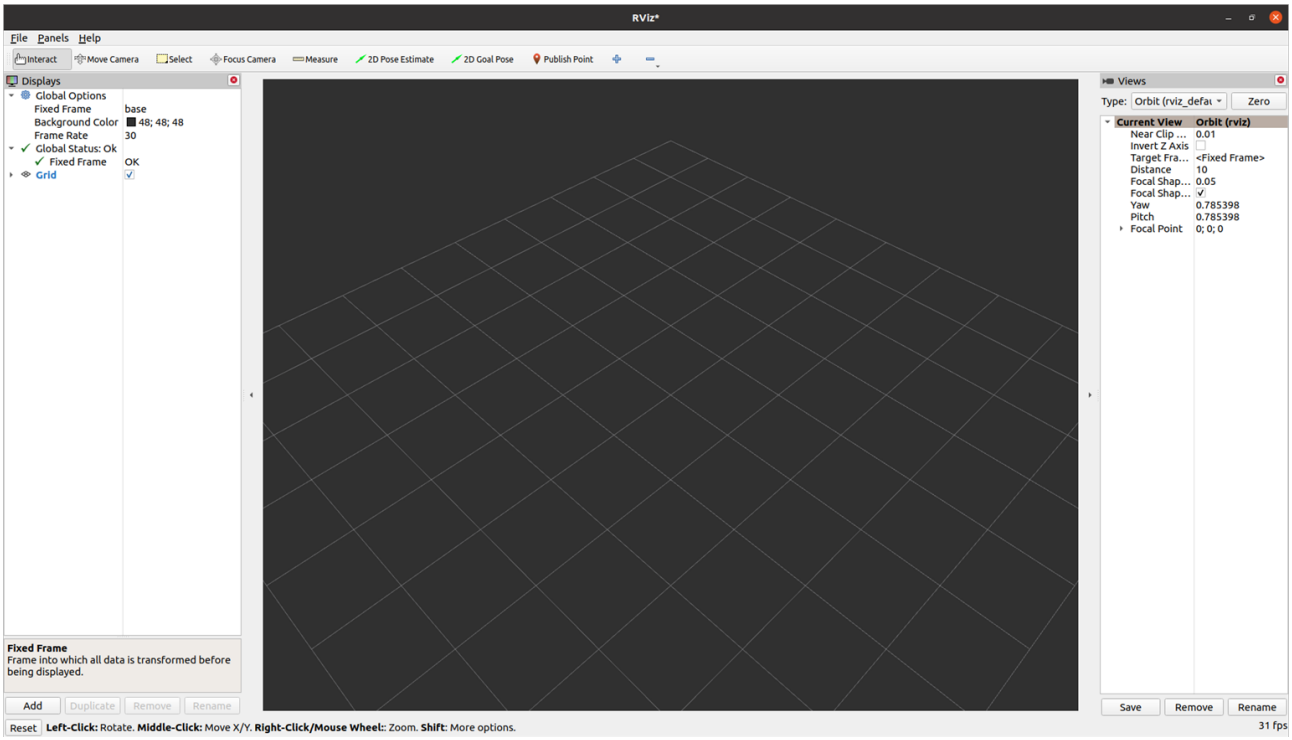
Execute in Terminal #2

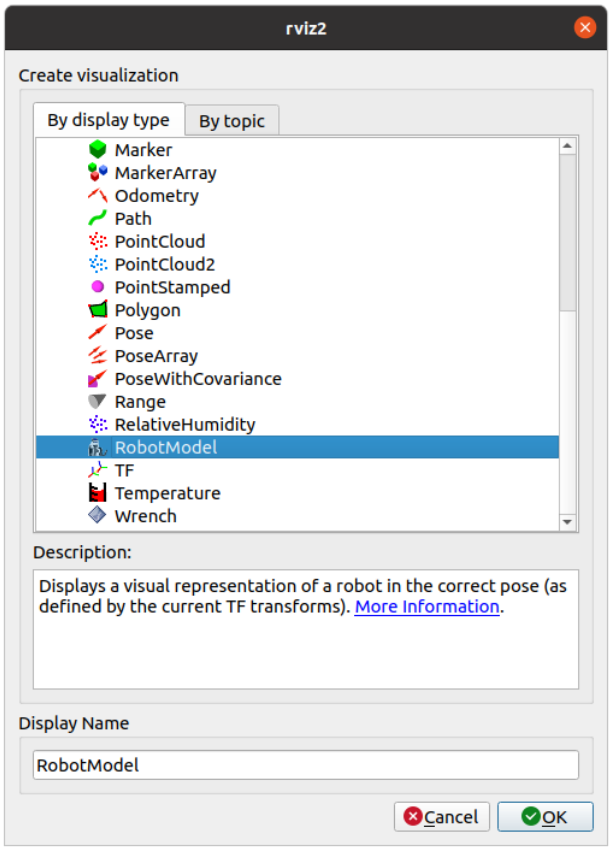
```
killall gzserver
ros2 launch lab4 gazebo.launch.py
```

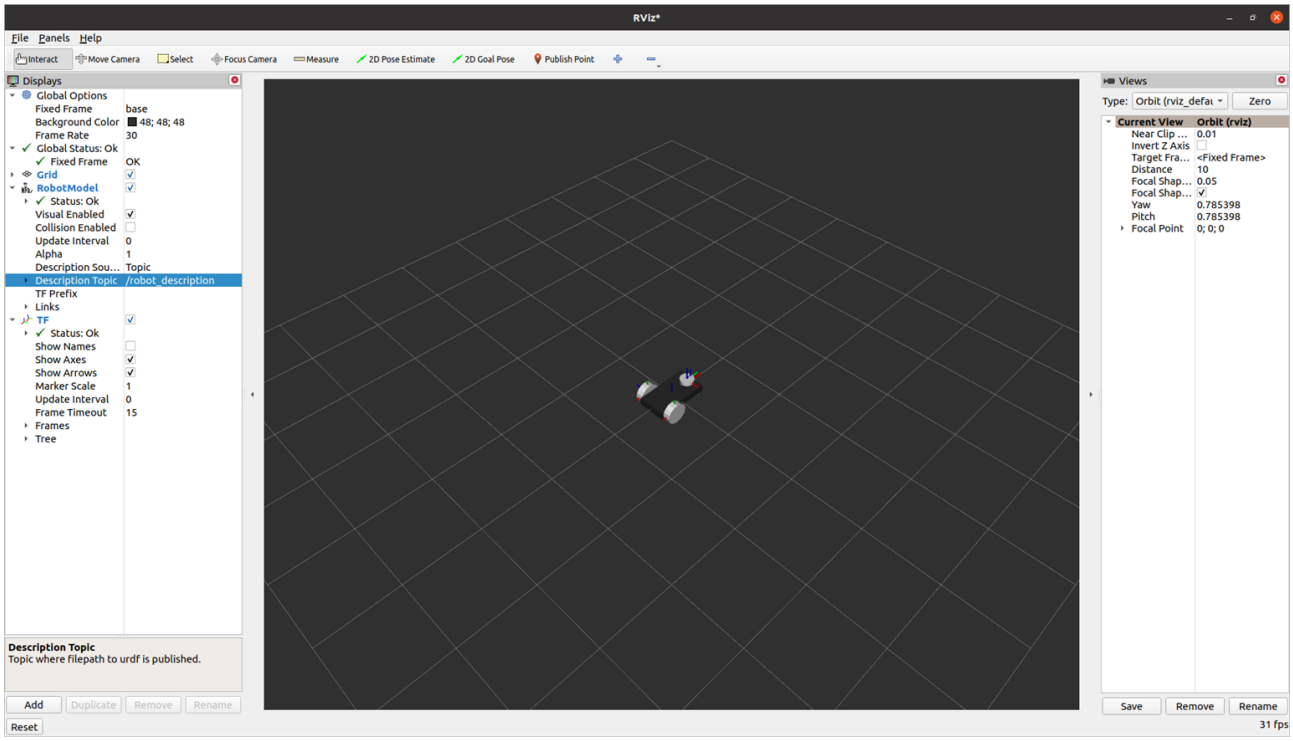
Execute in Terminal #3

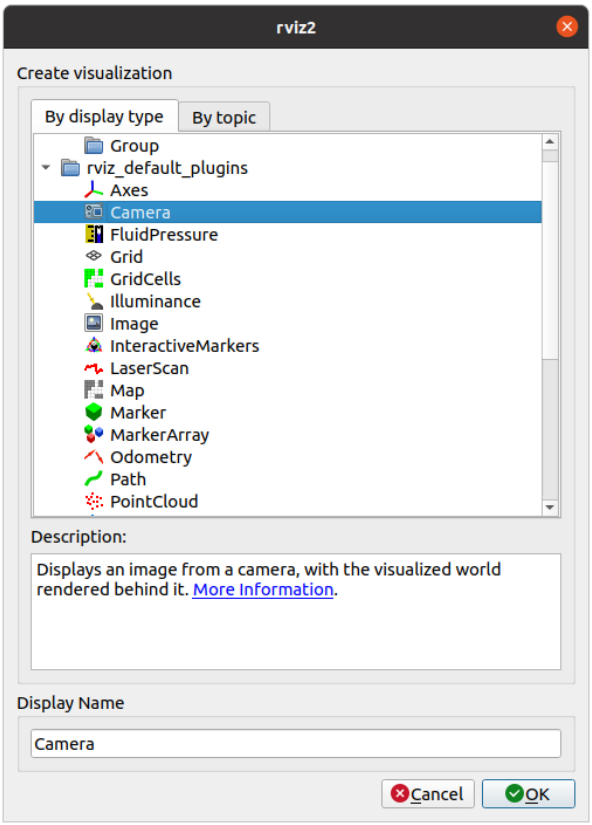
```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

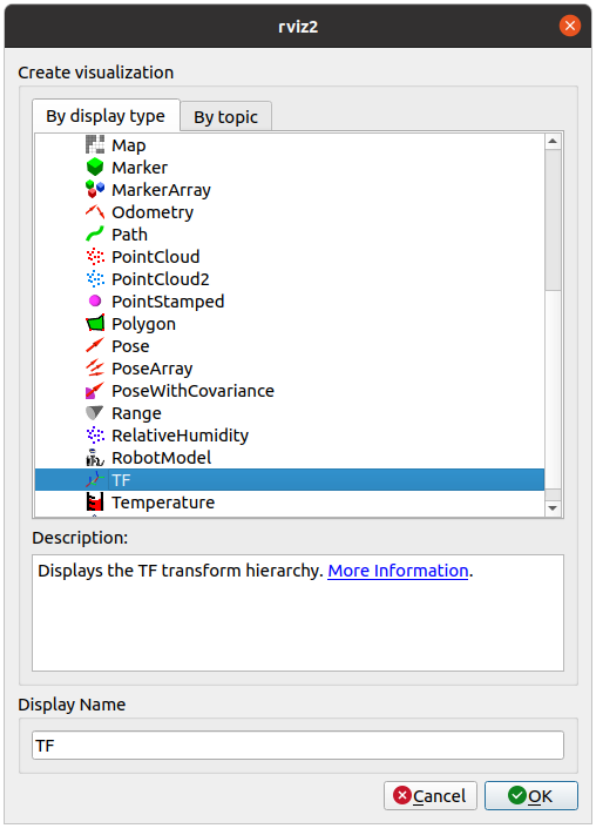
Do these changes in the rviz window:

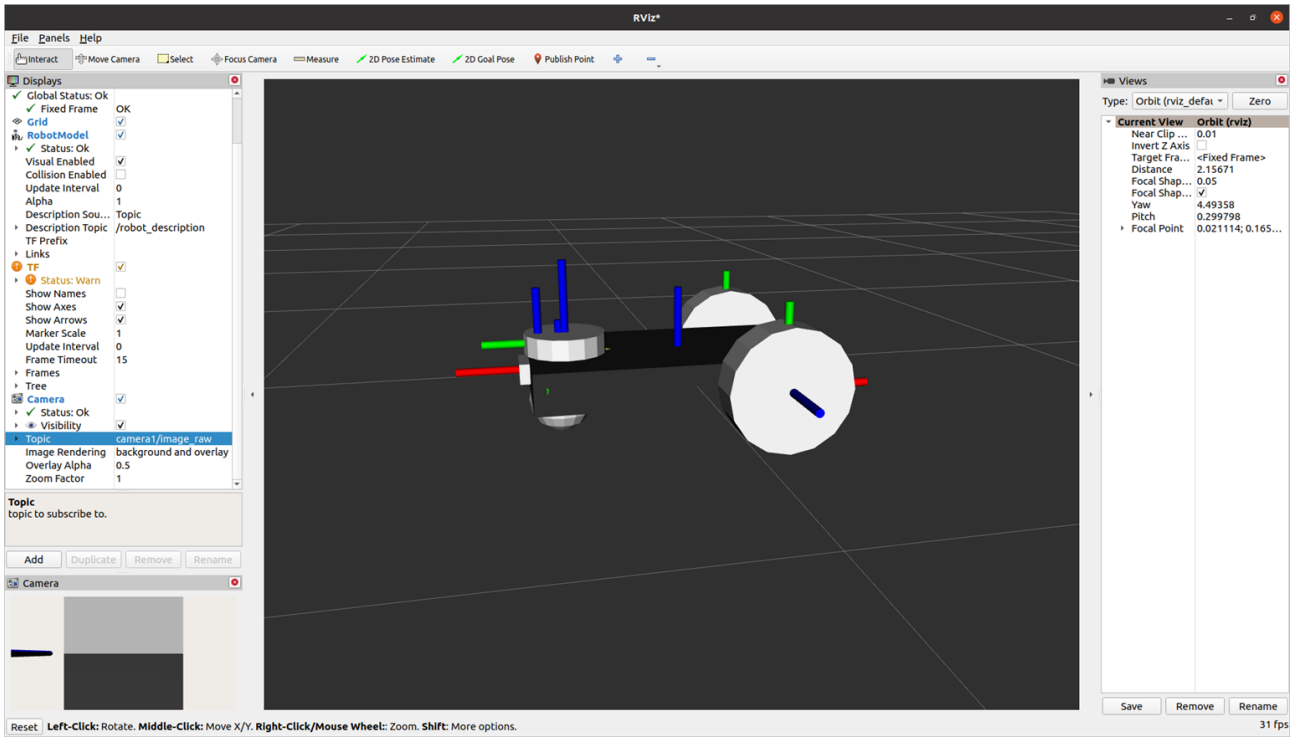

RViz*
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Displays
Global Options
Fixed Frame base
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
Fixed Frame
Frame into which all data is transformed before being displayed.
Add
Duplicate
Remove
Rename
Views
Type: Orbit (rviz_defa
Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis
Target Fra... <Fixed Frame>
Distance 10
Focal Shap... 0.05
Focal Shap...
Yaw 0.785398
Pitch 0.785398
Focal Point 0; 0; 0
Save
Remove
Rename
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps

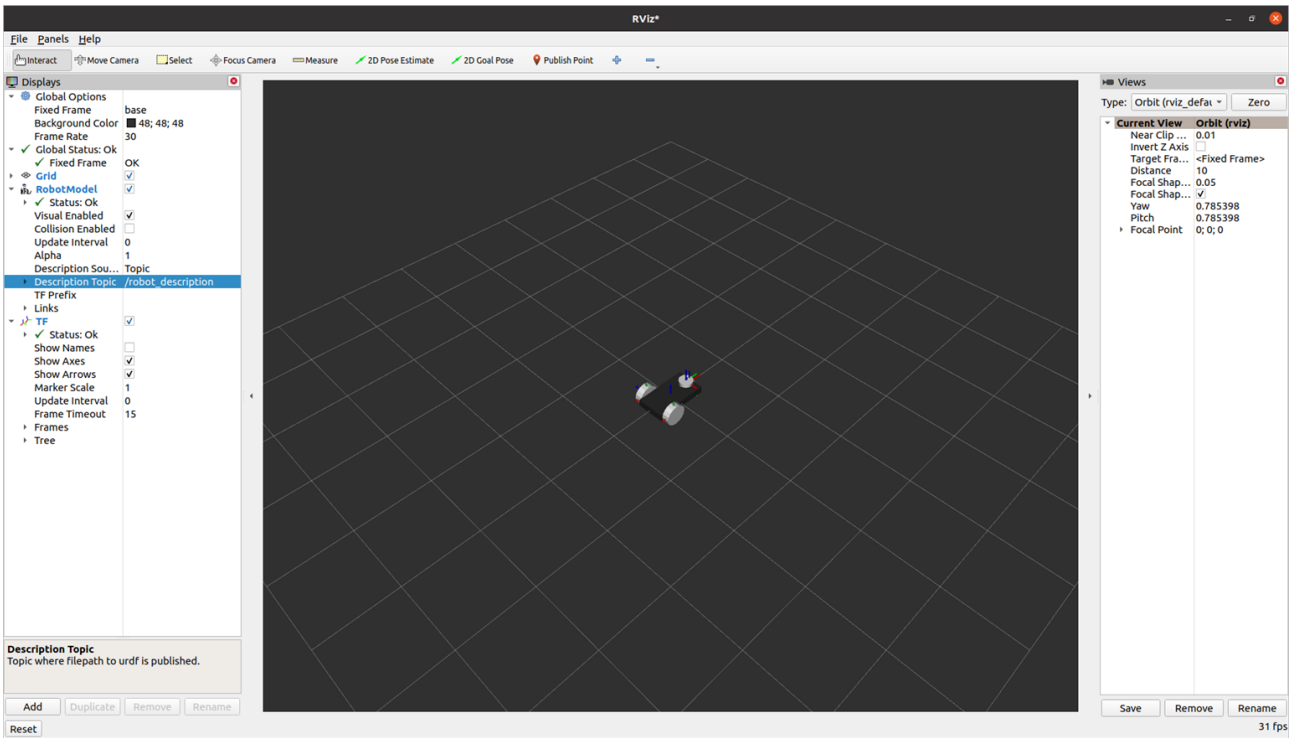

rviz2
Create visualization
By display type
By topic
Marker
MarkerArray
Odometry
Path
PointCloud
PointCloud2
PointStamped
Polygon
Pose
PoseArray
PoseWithCovariance
Range
RelativeHumidity
RobotModel
TF
Temperature
Wrench
Description:
Displays a visual representation of a robot in the correct pose (as defined by the current TF transforms). More Information.
Display Name
RobotModel
Cancel
OK

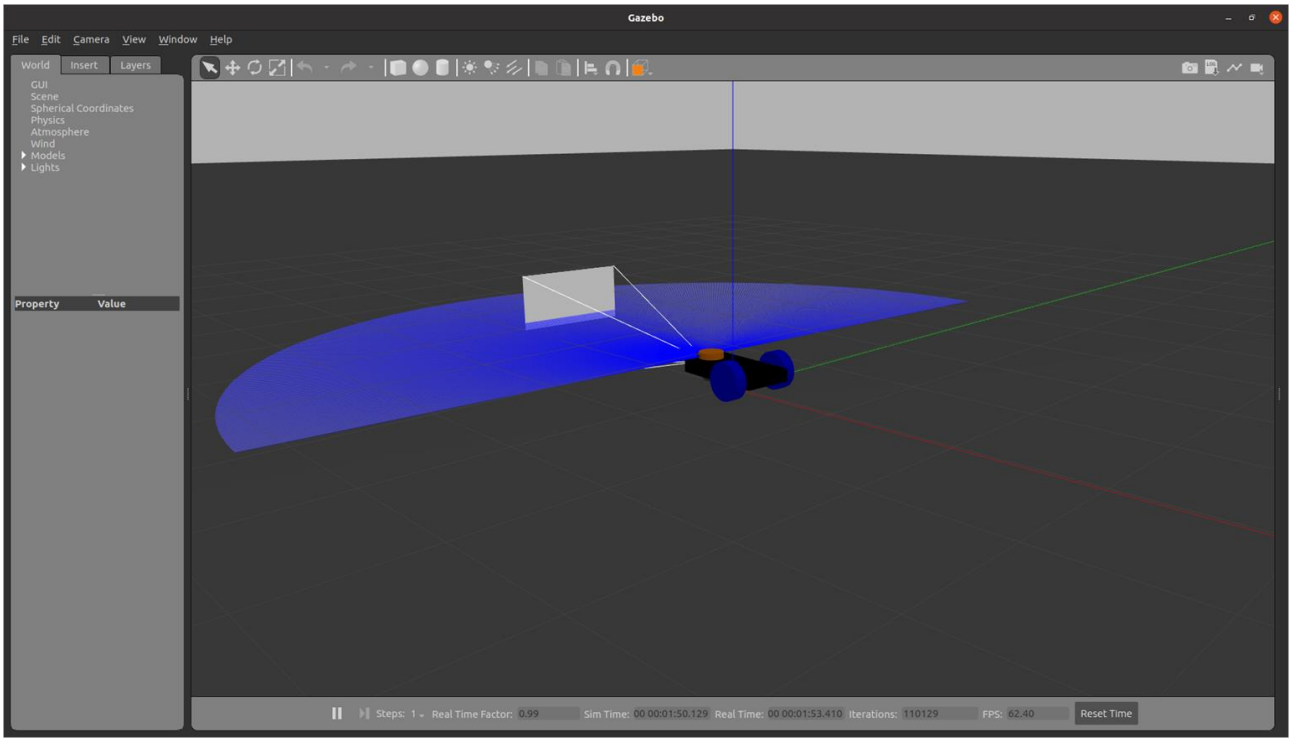

RViz*
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Displays
Global Options
Fixed Frame
base
Background Color
48; 48; 48
Frame Rate
30
Global Status: Ok
Fixed Frame
OK
Grid
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
Update Interval
0
Alpha
1
Description Sou...
Topic
Description Topic
/robot_description
TF Prefix
Links
TF
Status: Ok
Show Names
Show Axes
Show Arrows
Marker Scale
1
Update Interval
0
Frame Timeout
15
Frames
Tree
Description Topic
Topic where filepath to urdf is published.
Add
Duplicate
Remove
Rename
Reset
Views
Type:
Orbit (rviz_defa
Zero
Current View
Orbit (rviz)
Near Clip ...
0.01
Invert Z Axis
Target Fra...
<Fixed Frame>
Distance
10
Focal Shap...
0.05
Focal Shap...
Yaw
0.785398
Pitch
0.785398
Focal Point
0; 0; 0
Save
Remove
Rename
31 fps

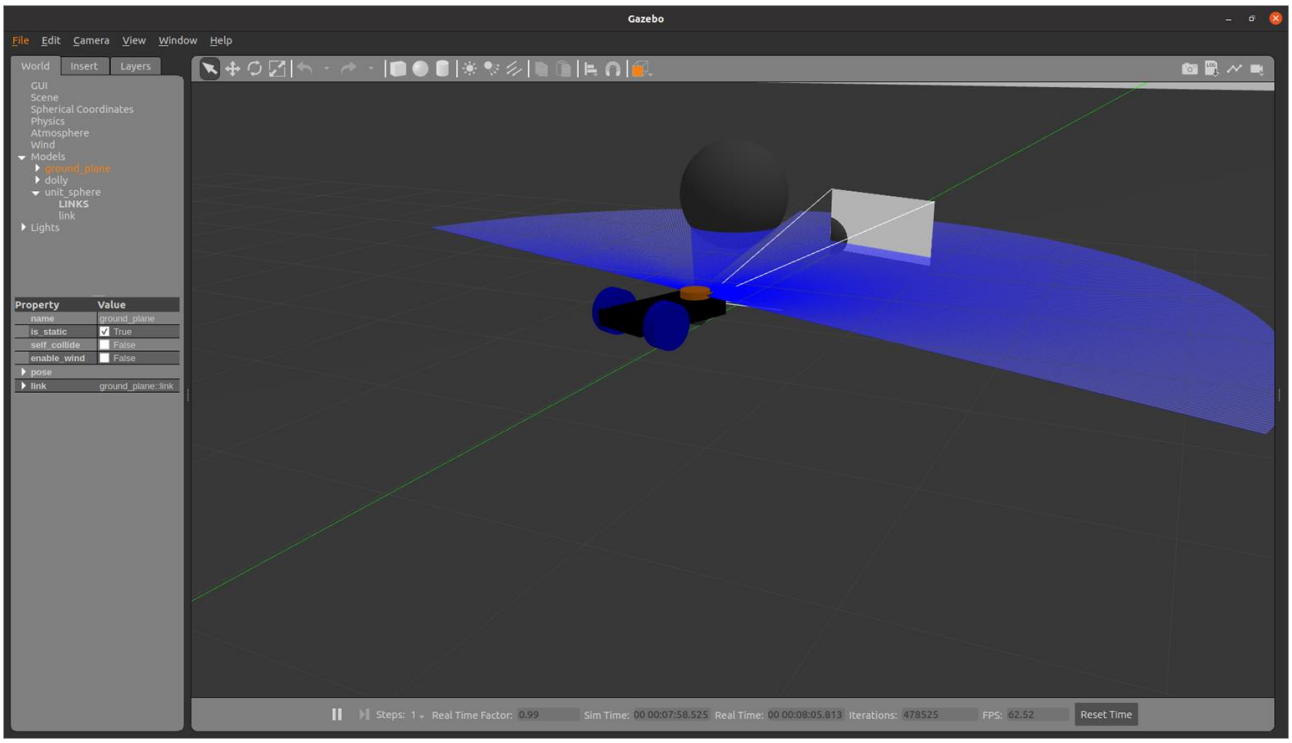

rviz2
Create visualization
By display type
By topic
Group
rviz_default_plugins
Axes
Camera
FluidPressure
Grid
GridCells
Illuminance
Image
InteractiveMarkers
LaserScan
Map
Marker
MarkerArray
Odometry
Path
PointCloud
Description:
Displays an image from a camera, with the visualized world rendered behind it. More Information.
Display Name
Camera
Cancel
OK


rviz2
Create visualization
By display type
By topic
Map
Marker
MarkerArray
Odometry
Path
PointCloud
PointCloud2
PointStamped
Polygon
Pose
PoseArray
PoseWithCovariance
Range
RelativeHumidity
RobotModel
TF
Temperature
Description:
Displays the TF transform hierarchy. More Information.
Display Name
TF
Cancel
OK


RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point
Displays
Global Status: Ok
Fixed Frame OK
Grid
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
Update Interval 0
Alpha 1
Description Sou… Topic
Description Topic /robot_description
TF Prefix
Links
TF
Status: Warn
Show Names
Show Axes
Show Arrows
Marker Scale 1
Update Interval 0
Frame Timeout 15
Frames
Tree
Camera
Status: Ok
Visibility
Topic camera1/image_raw
Image Rendering background and overlay
Overlay Alpha 0.5
Zoom Factor 1
Topic
topic to subscribe to.
Add Duplicate Remove Rename
Camera
Views
Type: Orbit (rviz_defa Zero
Current View Orbit (rviz)
Near Clip … 0.01
Invert Z Axis
Target Fra… <Fixed Frame>
Distance 2.15671
Focal Shap… 0.05
Focal Shap…
Yaw 4.49358
Pitch 0.299798
Focal Point 0.021114; 0.165…
Save Remove Rename
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps


RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point
Displays
Global Options
Fixed Frame base
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
Update Interval 0
Alpha 1
Description Sou… Topic
Description Topic /robot_description
TF Prefix
Links
TF
Status: Ok
Show Names
Show Axes
Show Arrows
Marker Scale 1
Update Interval 0
Frame Timeout 15
Frames
Tree
Description Topic
Topic where filepath to urdf is published.
Add Duplicate Remove Rename
Views
Type: Orbit (rviz_defa Zero
Current View Orbit (rviz)
Near Clip … 0.01
Invert Z Axis
Target Fra… <Fixed Frame>
Distance 10
Focal Shap… 0.05
Focal Shap…
Yaw 0.785398
Pitch 0.785398
Focal Point 0; 0; 0
Save Remove Rename
Reset
31 fps


Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:01:50.129 Real Time: 00 00:01:53.410 Iterations: 110129 FPS: 62.40 Reset Time


Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
ground_plane
dolly
unit_sphere
LINKS
link
Lights
Property Value
name ground_plane
is_static True
self_collide False
enable_wind False
pose
link ground_plane::link
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:07:58.525 Real Time: 00 00:08:05.813 Iterations: 478525 FPS: 62.52 Reset Time

sudo apt-get install python3-pip
pip3 install transforms3d

Edit move_robot.py

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
import transforms3d
import math


class GotoGoalNode(Node):
    def __init__(self):
        super().__init__("move_robot")
        self.target_x = 2
        self.target_y = 2
        self.publisher = self.create_publisher(Twist, "cmd_vel", 10)
        self.subscriber = self.create_subscription(Odometry, "odom", self.control_loop, 10)

    def control_loop(self, msg):

        dist_x = self.target_x - msg.pose.pose.position.x
        dist_y = self.target_y - msg.pose.pose.position.y
        print('current position: {}
{}'.format(msg.pose.pose.position.x,msg.pose.pose.position.y))
        distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)
        print('distance : {}'.format(round(distance, 3)))


        goal_theta = math.atan2(dist_y, dist_x)
        quat = msg.pose.pose.orientation
        roll, pitch, yaw = transforms3d.euler.quat2euler([quat.w,quat.x,quat.y,quat.z])
        diff = math.pi - round(yaw, 2) + round(goal_theta, 2)
        print('yaw: {}'.format(round(yaw, 2)))
        print('target angle: {}'.format(round(goal_theta, 2)))

        if diff > math.pi:
            diff -= 2*math.pi
        elif diff < -math.pi:
            diff += 2*math.pi
        print('orientation : {}'.format(round(diff, 2)))

        vel = Twist()

        if abs(diff) > 0.2:
            vel.linear.x = 0.0
            vel.angular.z = 0.4*round(diff, 2)
```

```
        else:
            if abs(distance) > 0.2:
                vel.linear.x = 0.3*round(distance, 3)
                vel.angular.z = 0.0

            else:
                vel.linear.x = 0.0
                vel.angular.z = 0.0

        print('speed : {}'.format(vel))
        self.publisher.publish(vel)


def main(args=None):
    rclpy.init(args=args)
    node = GotoGoalNode()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
          main()
```

Execute in Terminal #1

```
colcon build --packages-select lab4
ros2 launch lab4 rviz.launch.py
```

Execute in Terminal #2

```
killall gzserver
ros2 launch lab4 gazebo.launch.py
```

Execute in Terminal #3

```
ros2 run lab4 controller
```

Edit setup.py

```
 entry_points={
        'console_scripts': [
            'controller = lab4.move_robot:main'
        ],
```

Exercise 1: Write a python code to move the robot to along the wall using the laser scan data.

Exercise 2: Modify the URDF code to change to 4 wheeled robot and run the program.

MANIPULATOR DESIGN AND CONTROL

http://wiki.ros.org/urdf/Tutorials/Adding%20Physical%20and%20Collision%20Properties%20to%20a%20URDF%20Model

Execute in Terminal #1

```
sudo apt-get install ros-foxy-teleop-twist-keyboard
sudo apt-get install ros-foxy-joint-state-publisher*
sudo apt-get install ros-foxy-joint-trajectory-controller
sudo apt-get install ros-foxy-controller-manager
sudo apt install ros-foxy-gazebo-*
sudo apt install ros-foxy-gazebo-msgs
sudo apt install ros-foxy-gazebo-ros
sudo apt install ros-foxy-gazebo-ros2-control-demos
sudo apt install ros-foxy-ros2 control
sudo apt install ros-foxy-ros2-control
sudo apt install ros-foxy-ros2-controllers
sudo apt install ros-foxy-ros2controlcli
sudo apt install ros-foxy-xacro
sudo apt install ros-foxy-gazebo-dev
sudo apt install ros-foxy-gazebo-plugins
```

```
cd ros2_ws/src/urdf_tutorial/urdf
touch manipulator.urdf
```

```
<?xml version="1.0"?>
<robot name="arm">

    <link name="world"/>
    <link name="base_link">
        <visual>
            <geometry>
                <cylinder length="0.05" radius="0.2"/>
            </geometry>
            <material name="Black">
                <color rgba="0 0 0 1"/>
            </material>
            <origin rpy="0 0 0" xyz="0 0 0.025"/>
        </visual>

        <collision>
            <geometry>
                <cylinder length="0.05" radius="0.2"/>
            </geometry>
            <origin rpy="0 0 0" xyz="0 0 0.025"/>
```

```
    </collision>

    <inertial>
        <origin rpy="0 0 0" xyz="0 0 0.025"/>
        <mass value="5.0"/>
        <inertia ixx="0.0135" ixy="0.0" ixz="0.0" iyy="0.0135" iyz="0.0" izz="0.05"/>
    </inertial>
</link>

<joint name="fixed" type="fixed">
    <parent link="world"/>
    <child link="base_link"/>
    <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_1">
    <visual>
        <geometry>
            <cylinder length="0.5" radius="0.08"/>
        </geometry>
        <material name="blue">
            <color rgba="0 0 0.8 1"/>
        </material>
        <origin rpy="0 0 0" xyz="0 0 0.25"/>
    </visual>

    <collision>
        <geometry>
            <cylinder length="0.5" radius="0.08"/>
        </geometry>
        <origin rpy="0 0 0" xyz="0 0 0.25"/>
    </collision>

    <inertial>
        <origin rpy="0 0 0" xyz="0 0 0.25"/>
        <mass value="5.0"/>
        <inertia ixx="0.107" ixy="0.0" ixz="0.0" iyy="0.107" iyz="0.0" izz="0.0125"/>
    </inertial>

</link>

<joint name="joint_1" type="continuous">
    <axis xyz="0 0 1"/>
    <parent link="base_link"/>
    <child link="link_1"/>
    <origin rpy="0 0 0" xyz="0.0 0.0 0.05"/>
    <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_2">
    <inertial>
```

```
      <origin rpy="0 0 0" xyz="0 0 0.2"/>
      <mass value="2.0"/>
      <inertia ixx="0.027" ixy="0.0" ixz="0.0" iyy="0.027" iyz="0.0" izz="0.0025"/>
    </inertial>

    <visual>
      <geometry>
      <cylinder length="0.1" radius="0.08"/>
      </geometry>
      <material name="Red">
        <color rgba="1 0 0 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
        <cylinder length="0.1" radius="0.08"/>
      </geometry>
    </collision>
  </link>

  <joint name="joint_2" type="continuous">
    <axis xyz="0 0 1"/>
    <parent link="link_1"/>
    <child link="link_2"/>
    <origin rpy="0 1.5708 0" xyz="0.0 -0.005 0.58"/>
    <limit lower="-0.25" upper="3.34" effort="10" velocity="0.5"/>
    <dynamics damping="10" friction="1.0"/>
  </joint>

  <link name="link_3">
    <inertial>
      <origin rpy="0 0 0" xyz="0 0 0.2"/>
      <mass value="0.01"/>
      <inertia ixx="0.027" ixy="0.0" ixz="0.0" iyy="0.027" iyz="0.0" izz="0.0025"/>
    </inertial>

    <visual>
      <geometry>
      <cylinder length="0.4" radius="0.05"/>
      </geometry>
      <material name="blue">
        <color rgba="0.5 0.5 0.5 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
        <cylinder length="0.4" radius="0.05"/>
      </geometry>
    </collision>
```

```
</link>

<joint name="joint_3" type="fixed">
   <parent link="link_2"/>
   <child link="link_3"/>
   <origin rpy="1.57 0 0" xyz="0.0 0.2 0 "/>
   <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_4">
   <inertial>
      <origin rpy="0 0 0" xyz="0 0 0.2"/>
      <mass value="0.01"/>
      <inertia ixx="0.027" ixy="0.0" ixz="0.0" iyy="0.027" iyz="0.0" izz="0.0025"/>
   </inertial>

   <visual>
      <geometry>
      <cylinder length="0.1" radius="0.06"/>
      </geometry>
      <material name="Red">
         <color rgba="1 0 0 1"/>
      </material>
   </visual>

   <collision>
      <geometry>
         <cylinder length="0.1" radius="0.06"/>
      </geometry>
   </collision>

</link>

<joint name="joint_4" type="continuous">
   <parent link="link_3"/>
   <child link="link_4"/>
   <origin rpy="1.57 0 0" xyz=" 0 0 -0.25"/>
   <axis xyz=" 0 0 1"/>
   <limit lower="-1.92" upper="1.92" effort="10" velocity="0.5"/>
   <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_5">
   <inertial>
      <origin rpy="0 0 0" xyz="0 0 0.2"/>
      <mass value="0.01"/>
      <inertia ixx="0.027" ixy="0.0" ixz="0.0" iyy="0.027" iyz="0.0" izz="0.0025"/>
   </inertial>

   <visual>
      <geometry>
```

```
        <cylinder length="0.3" radius="0.03"/>
        </geometry>
        <material name="yello">
            <color rgba="0 1 0.5 1"/>
        </material>
        </visual>

    <collision>
        <geometry>
            <cylinder length="0.3" radius="0.03"/>
        </geometry>
    <dynamics damping="0.0" friction="0.0"/>

    </collision>
</link>

<joint name="joint_5" type="fixed">
    <parent link="link_4"/>
    <child link="link_5"/>
    <origin rpy="1.57 0 0" xyz="0.0 -0.2 0 "/>
    <dynamics damping="10" friction="1.0"/>
</joint>

<gazebo reference="base_link">
    <material>Gazebo/Black</material>
</gazebo>

<gazebo reference="link_1">
    <material>Gazebo/White</material>
</gazebo>

<gazebo reference="link_3">
    <material>Gazebo/White</material>
</gazebo>

<gazebo reference="link_2">
    <material>Gazebo/Blue</material>
</gazebo>

<gazebo reference="link_4">
    <material>Gazebo/Blue</material>
</gazebo>

<gazebo reference="link_5">
    <material>Gazebo/White</material>
</gazebo>



<gazebo>
<plugin filename="libgazebo_ros2_control.so" name="gazebo_ros2_control">
```

```
      <robot_sim_type>gazebo_ros2_control/GazeboSystem</robot_sim_type>

<parameters>/home/asha/ros2_ws/src/urdf_tutorial/config/control.yaml</parameters>
   </plugin>
   </gazebo>

   <ros2_control name="GazeboSystem" type="system">
      <hardware>
      <plugin>gazebo_ros2_control/GazeboSystem</plugin>
   </hardware>


   <joint name="joint_1">
      <command_interface name="position">
      <param name="min">-3.14</param>
      <param name="max">3.14</param>
      </command_interface>
      <state_interface name="position"/>
      <param name="initial_position">0.0</param>
   </joint>
   <joint name="joint_2">
      <command_interface name="position">
      <param name="min">-3.14</param>
      <param name="max">3.14</param>
      </command_interface>
      <state_interface name="position"/>
      <param name="initial_position">-1.57</param>
   </joint>
   <joint name="joint_4">
      <command_interface name="position">
      <param name="min">-3.14</param>
      <param name="max">3.14</param>
      </command_interface>
      <state_interface name="position"/>
      <param name="initial_position">0.0</param>
   </joint>

   </ros2_control>

</robot>
```

```
cd ..
cd launch
touch arm_rviz.launch.py
```

```
from launch import LaunchDescription
from launch_ros.actions import Node
import os
```

```
def generate_launch_description():

    urdf_file = urdf = '/home/asha/ros2_ws/src/urdf_tutorial/urdf/manipulator.urdf'



    joint_state_publisher_node = Node(
        package="joint_state_publisher_gui",
        executable="joint_state_publisher_gui",
    )
    robot_state_publisher_node = Node(
        package="robot_state_publisher",
        executable="robot_state_publisher",
        output="both",
        arguments=[urdf_file]
    )
    rviz_node = Node(
        package="rviz2",
        executable="rviz2",
        name="rviz2",
        output="log"
    )

    nodes_to_run = [
        joint_state_publisher_node,
        robot_state_publisher_node,
        rviz_node
    ]

    return LaunchDescription(nodes_to_run)
```

touch arm_gazebo.launch.py

```
import os
from launch import LaunchDescription
from launch.actions import ExecuteProcess
from launch_ros.actions import Node

def generate_launch_description():
    urdf_file = '/home/asha/ros2_ws/src/urdf_tutorial/urdf/manipulator.urdf'

    return LaunchDescription(
        [
            ExecuteProcess(
                cmd=["gazebo","-s","libgazebo_ros_factory.so",],
                output="screen",
            ),
            Node(
                package="gazebo_ros",
                executable="spawn_entity.py",
```

```
            arguments=["-entity","urdf_tutorial","-b","-file", urdf_file],
        ),
        Node(
            package="robot_state_publisher",
            executable="robot_state_publisher",
            output="screen",
            arguments=[urdf_file],
        ),


    ]
)
```

```
cd ~/ros2_ws/src/urdf_tutorial
mkdir config
cd config
touch control.yaml
```

```
controller_manager:
  ros__parameters:
    update_rate: 100
    joint_state_broadcaster:
      type: joint_state_broadcaster/JointStateBroadcaster
    joint_trajectory_controller:
      type: joint_trajectory_controller/JointTrajectoryController

joint_trajectory_controller:
  ros__parameters:
    joints:
      - joint_1
      - joint_2
      - joint_4

    command_interfaces:
      - position

    state_interfaces:
      - position

    state_publish_rate: 50.0
    action_monitor_rate: 20.0

    allow_partial_joints_goal: false
    open_loop_control: true
    constraints:
      stopped_velocity_tolerance: 0.01
      goal_time: 0.0
      joint1:
        trajectory: 0.05
        goal: 0.03
```

touch arm_control.launch.py
Edit arm_control.launch.py

```
import os
from launch import LaunchDescription
from launch.actions import ExecuteProcess, IncludeLaunchDescription,
RegisterEventHandler
from launch_ros.actions import Node
from launch.event_handlers import OnProcessExit
from launch.launch_description_sources import PythonLaunchDescriptionSource
from ament_index_python.packages import get_package_share_directory

import xacro

def generate_launch_description():

    urdf_file = '/home/asha/ros2_ws/src/lab5/urdf/manipulator.urdf'
    controller_file = '/home/asha/ros2_ws/src/lab5/config/control.yaml'
    robot_description = {"robot_description": urdf_file}

    gazebo = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('gazebo_ros'), 'launch'), '/gazebo.launch.py']),
    )

    doc = xacro.parse(open(urdf_file))
    xacro.process_doc(doc)
    params = {'robot_description': doc.toxml()}

    node_robot_state_publisher = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        output='screen',
        parameters=[params]
    )

    spawn_entity = Node(package='gazebo_ros', executable='spawn_entity.py',
        arguments=["-entity","lab5","-b","-file", urdf_file],
        output='screen'
    )

    load_joint_state_controller = ExecuteProcess(
        cmd=['ros2', 'control', 'load_controller', '--set-state', 'start',
             'joint_state_broadcaster'],
        output='screen'
    )

    load_joint_trajectory_controller = ExecuteProcess(
        cmd=['ros2', 'control', 'load_controller', '--set-state', 'start',
```

```
            'joint_trajectory_controller'],
        output='screen'
    )

    return LaunchDescription(
        [
            RegisterEventHandler(
                event_handler=OnProcessExit(
                    target_action=spawn_entity,
                    on_exit=[load_joint_state_controller],
                )
            ),

            RegisterEventHandler(
                event_handler=OnProcessExit(
                    target_action=load_joint_state_controller,
                    on_exit=[load_joint_trajectory_controller],
                )
            ),

            gazebo,

            node_robot_state_publisher,

            spawn_entity,

            Node(
                package="controller_manager",
                executable="ros2_control_node",
                parameters=[robot_description, controller_file],
                output="screen"
            )
        ]
    )
```

Add extensions in Visual Studio

ros
ros snippet
xml
xml tools
urdf
xml complete
icons

Execute in Terminal #1

```
colcon build --packages-select urdf_tutorial
```

Execute in Terminal #1

ros2 launch urdf_tutorial arm_rviz.launch.py

Execute in Terminal #2

ros2 launch urdf_tutorial arm_gazebo.launch.py

Execute in Terminal #3

ros2 launch urdf_tutorial arm_control.launch.py

cd ros2_ws/src/urdf_tutorial/urdf_tutorial/

touch controller.py

chmod +x controller.py

```
#!/usr/bin/env python3

import rclpy
from rclpy.node import Node
from builtin_interfaces.msg import Duration
from trajectory_msgs.msg import JointTrajectory, JointTrajectoryPoint

class TrajectoryPublisher(Node):

    def __init__(self):
        super().__init__('trajectory_node')
        topic_ = "/joint_trajectory_controller/joint_trajectory"
        self.publisher_ = self.create_publisher(JointTrajectory, topic_, 10)
        self.timer_ = self.create_timer(1,self.timer_callback)
        self.joints = ['joint_1', 'joint_2', 'joint_4']
        self.goal_ =[1.5, 0.5, 1.2]


    def timer_callback(self):
        msg = JointTrajectory()
        msg.joint_names = self.joints
        point = JointTrajectoryPoint()
        point.positions = self.goal_
        point.time_from_start = Duration(sec=2)
        msg.points.append(point)
        self.publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = TrajectoryPublisher()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Edit setup.py as

```
from setuptools import setup
import os
from glob import glob
package_name = 'urdf_tutorial'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
          (os.path.join('share', package_name), glob('urdf/*')),
          (os.path.join('share', package_name), glob('launch/*')),
          (os.path.join('share', package_name), glob('config/*'))


    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='asha',
    maintainer_email='asha@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'controller = urdf_tutorial.controller:main'
        ],
    },
)
```

Execute in Terminal #1

```
colcon build --packages-select urdf_tutorial
```

Execute in Terminal #1

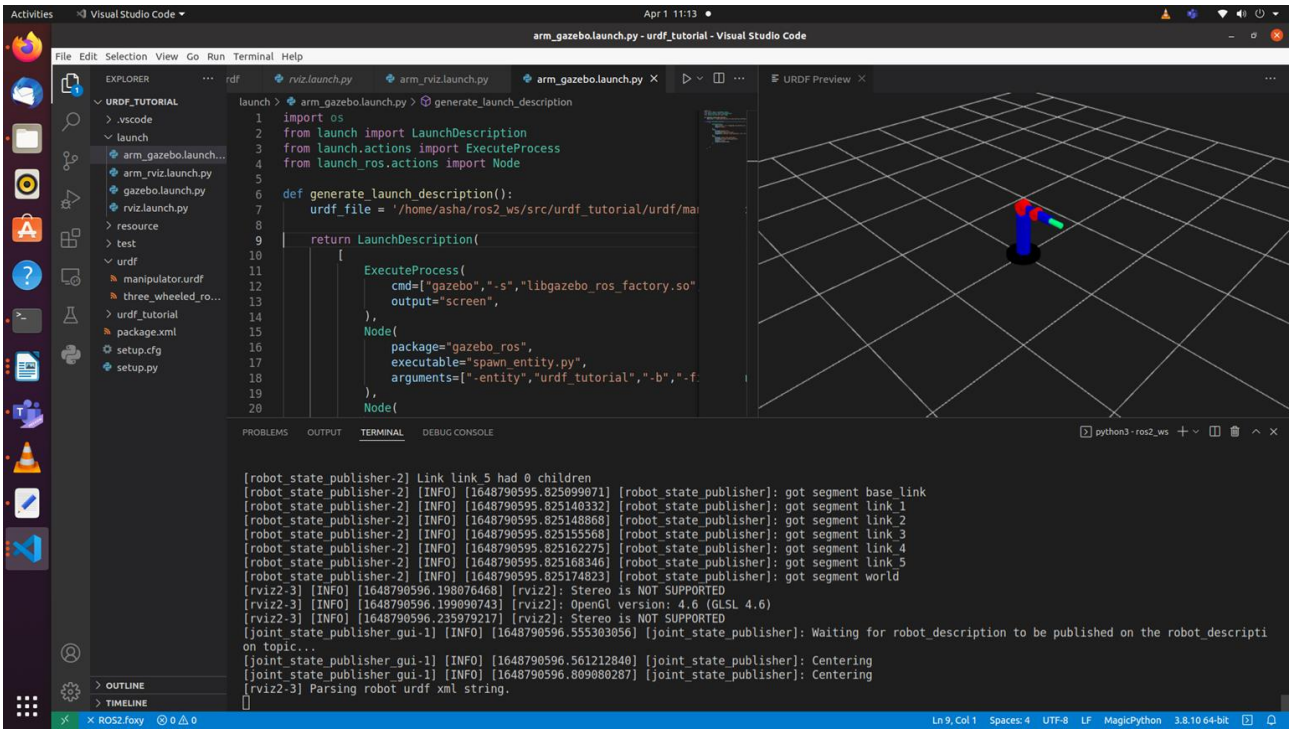
```
ros2 launch urdf_tutorial arm_rviz.launch.py
```

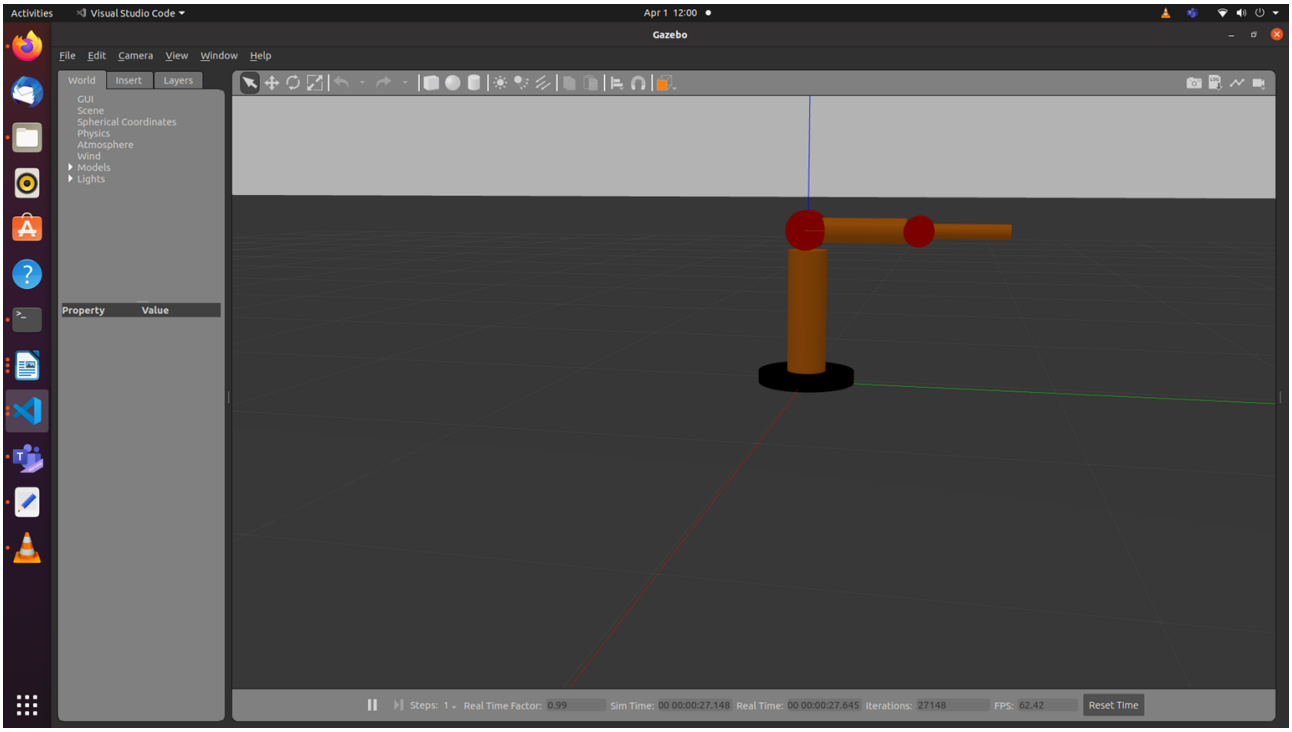
Execute in Terminal #2

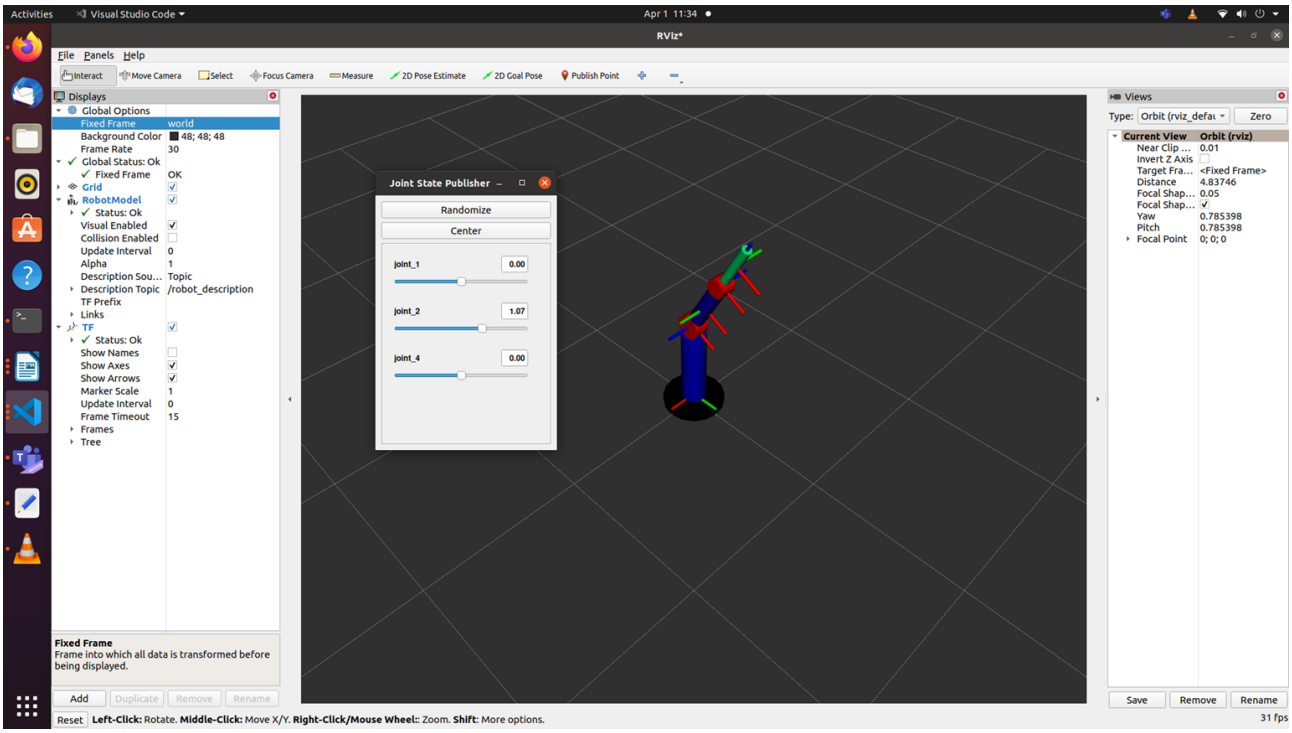
```
ros2 launch urdf_tutorial arm_control.launch.py
```

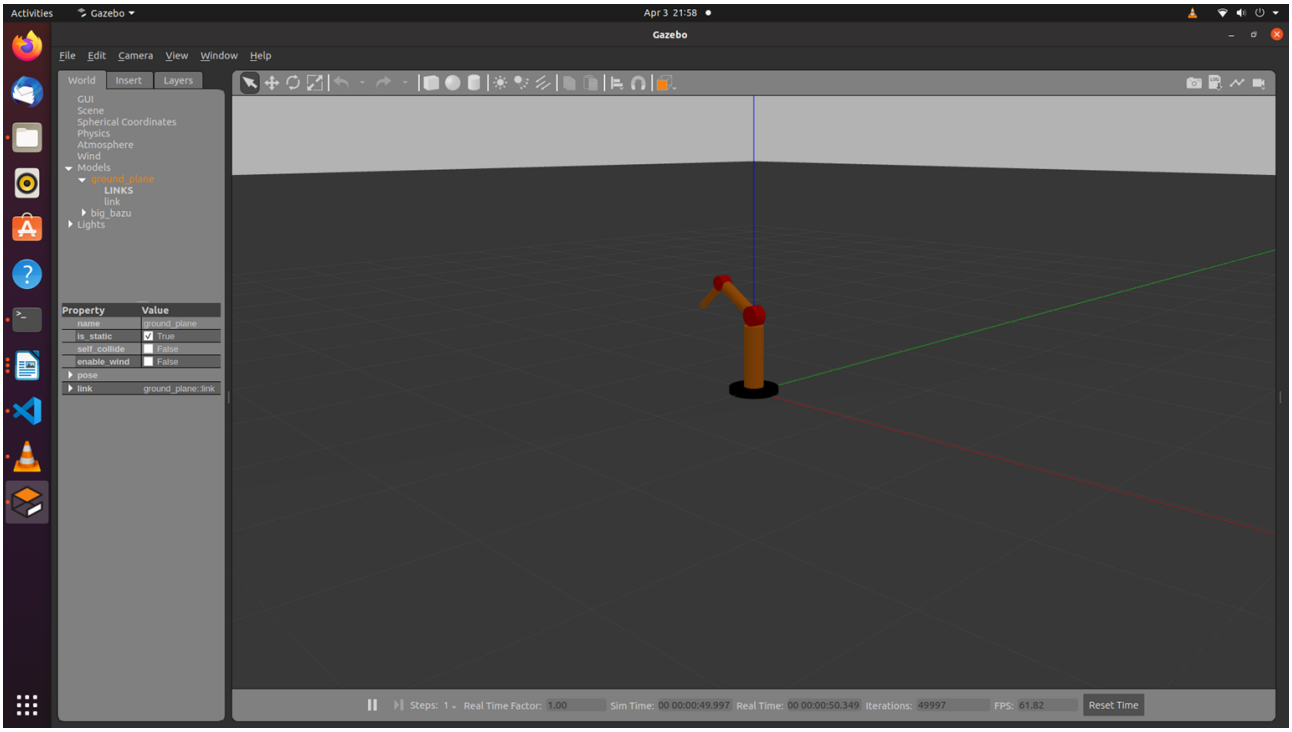
Execute in Terminal #3

```
ros2 run urdf_tutorial controller
```


Activities
Visual Studio Code
Apr 1 11:13
arm_gazebo.launch.py - urdf_tutorial - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
URDF_TUTORIAL
.vscode
launch
arm_gazebo.launch...
arm_rviz.launch.py
gazebo.launch.py
rviz.launch.py
resource
test
urdf
manipulator.urdf
three_wheeled_ro...
urdf_tutorial
package.xml
setup.cfg
setup.py
rviz.launch.py
arm_rviz.launch.py
arm_gazebo.launch.py
launch > arm_gazebo.launch.py > generate_launch_description
import os
from launch import LaunchDescription
from launch.actions import ExecuteProcess
from launch_ros.actions import Node
def generate_launch_description():
urdf_file = '/home/asha/ros2_ws/src/urdf_tutorial/urdf/ma
return LaunchDescription(
[
ExecuteProcess(
cmd=["gazebo","-s","libgazebo_ros_factory.so"
output="screen",
),
Node(
package="gazebo_ros",
executable="spawn_entity.py",
arguments=["-entity","urdf_tutorial","-b","-f
),
Node(
URDF Preview
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
python3 - ros2_ws
[robot_state_publisher-2] Link link_5 had 0 children
[robot_state_publisher-2] [INFO] [1648790595.825099071] [robot_state_publisher]: got segment base_link
[robot_state_publisher-2] [INFO] [1648790595.825140332] [robot_state_publisher]: got segment link_1
[robot_state_publisher-2] [INFO] [1648790595.825148868] [robot_state_publisher]: got segment link_2
[robot_state_publisher-2] [INFO] [1648790595.825155568] [robot_state_publisher]: got segment link_3
[robot_state_publisher-2] [INFO] [1648790595.825162275] [robot_state_publisher]: got segment link_4
[robot_state_publisher-2] [INFO] [1648790595.825168346] [robot_state_publisher]: got segment link_5
[robot_state_publisher-2] [INFO] [1648790595.825174823] [robot_state_publisher]: got segment world
[rviz2-3] [INFO] [1648790596.198076468] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-3] [INFO] [1648790596.199090743] [rviz2]: OpenGl version: 4.6 (GLSL 4.6)
[rviz2-3] [INFO] [1648790596.235979217] [rviz2]: Stereo is NOT SUPPORTED
[joint_state_publisher_gui-1] [INFO] [1648790596.555303056] [joint_state_publisher]: Waiting for robot_description to be published on the robot_descripti
on topic...
[joint_state_publisher_gui-1] [INFO] [1648790596.561212840] [joint_state_publisher]: Centering
[joint_state_publisher_gui-1] [INFO] [1648790596.809080287] [joint_state_publisher]: Centering
[rviz2-3] Parsing robot urdf xml string.
OUTLINE
TIMELINE
ROS2.foxy
Ln 9, Col 1
Spaces: 4
UTF-8
LF
MagicPython
3.8.10 64-bit


Activities
Visual Studio Code
Apr 1 12:00
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:00:27.148 Real Time: 00 00:00:27.645 Iterations: 27148 FPS: 62.42
Reset Time


Activities
Visual Studio Code
Apr 1 11:34
RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point
Displays
Global Options
Fixed Frame world
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
Update Interval 0
Alpha 1
Description Sou... Topic
Description Topic /robot_description
TF Prefix
Links
TF
Status: Ok
Show Names
Show Axes
Show Arrows
Marker Scale 1
Update Interval 0
Frame Timeout 15
Frames
Tree
Fixed Frame
Frame into which all data is transformed before being displayed.
Add Duplicate Remove Rename
Joint State Publisher
Randomize
Center
joint_1 0.00
joint_2 1.07
joint_4 0.00
Views
Type: Orbit (rviz_defa
Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis
Target Fra... <Fixed Frame>
Distance 4.83746
Focal Shap... 0.05
Focal Shap...
Yaw 0.785398
Pitch 0.785398
Focal Point 0; 0; 0
Save Remove Rename
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps



ROS2 parameters:

Parameters help to provide the values while running the code.

ros2 param list

Edit the controller.py

```
#!/usr/bin/env python3


#colcon build --packages-select urdf_tutorial
#ros2 run urdf_tutorial controller --ros-args -p end_location:=[3.5,1.5,-1.2]

import rclpy
from rclpy.node import Node
from builtin_interfaces.msg import Duration
from trajectory_msgs.msg import JointTrajectory, JointTrajectoryPoint
```

```
class TrajectoryPublisher(Node):

    def __init__(self):
        super().__init__('trajectory_node')
        topic_ = "/joint_trajectory_controller/joint_trajectory"
        self.joints = ['joint_1', 'joint_2', 'joint_4']
        #self.goal_ =[1.5, 0.5, 1.2]
        self.declare_parameter("joint_angles", [1.5, 0.5, 1.2])
        self.goal_=self.get_parameter("joint_angles").value
        self.publisher_ = self.create_publisher(JointTrajectory, topic_, 10)
        self.timer_ = self.create_timer(1,self.timer_callback)



    def timer_callback(self):
        msg = JointTrajectory()
        msg.joint_names = self.joints
        point = JointTrajectoryPoint()
        point.positions = self.goal_
        point.time_from_start = Duration(sec=2)
        msg.points.append(point)
        self.publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = TrajectoryPublisher()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Execute in Terminal #1

```
#colcon build --packages-select urdf_tutorial
```

Execute in Terminal #2

```
ros2 launch urdf_tutorial arm_control.launch.py
ros2 control load_controller --set-state start joint_state_broadcaster
ros2 control load_controller --set-state start joint_trajectory_controller
```

Execute in Terminal #3

```
#ros2 run urdf_tutorial controller --ros-args -p joint_angles:=[3.5,1.5,-1.2]
```

Exercise 1: Replicate the process for UR5e robot given its urdf file.

Exercise 2: Write a python code to move the manipulator to end location using inverse kinematics.

Viva Questions: Compute the inertia parameters for the each block used in the three_wheeled_robot and manipulator.

References

https://docs.ros.org/en/foxy/Tutorials/URDF/Using-URDF-with-Robot-State-Publisher.html

https://github.com/benbongalon/ros2-urdf-tutorial/tree/master/urdf_tutorial

https://github.com/cra-ros-pkg/robot_localization/tree/foxy-devel

https://github.com/ros/robot_state_publisher/tree/foxy

https://github.com/ros/joint_state_publisher/tree/foxy

http://gazebosim.org/tutorials?tut=ros_urdf

Lab6: ROS2 Perception – Line Follower

- How to use OpenCV in ROS2
- How to follow a line
- How to find different elements based on color
- Track multiple paths and decide
- Create a basic PID for the line following

OpenCV is the most extensive and complete library for image recognition. With it, you can work with images like never before: applying filters, post-processing, and working with images in any way you want. OpenCV is not a ROS2 library, but it's been integrated nicely into ROS with http://wiki.ros.org/cv_bridge. This package allows the ROS imaging topics to use the OpenCV image variable format.

For example, OpenCV images come in BGR image format, while regular ROS images are in the more standard RGB encoding. OpenCV_bridge provides a nice feature to convert between them. Also, there are many other functions to transfer images to OpenCV variables transparently.

If gazebo is stuck use the following command
killall -9 gzserver
sudo apt install libopencv-dev python3-opencv

cd ros2_ws/src
ros2 pkg create lab6 --build-type ament_python --dependencies rclpy std_msgs
cd ..
colcon build
open lab6 with visual studio application
create urdf folder
create a file car.urdf inside the urdf folder

```
<?xml version="1.0" ?>

<robot name = "car">

   <link name="base">
      <visual>
      <geometry>
         <box size="0.75 0.4 0.1"/>
      </geometry>
      <material name="pink">
```

```
      <color rgba="1 0 1 1" />
    </material>
    </visual>

    <inertial>
      <mass value="1" />
      <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01" iyz="0" izz="0.01" />
    </inertial>

    <collision>
    <geometry>
      <box size="0.75 0.4 0.1"/>
    </geometry>
    </collision>

</link>

<link name="wheel_right_link">
    <inertial>
          <mass value="2" />
          <inertia ixx="0.01" ixy="0.0" ixz="0"
          iyy="0.01" iyz="0" izz="0.01" />
    </inertial>

    <visual>
      <geometry>
        <cylinder radius="0.15" length="0.1"/>
      </geometry>
      <material name="blue">
      <color rgba="0 0 1 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
        <cylinder radius="0.15" length="0.1"/>
      </geometry>
     <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
    </collision>
</link>

<joint name="wheel_right_joint" type="continuous">
    <origin xyz="0.2 0.25 0.0" rpy="1.57 0.0 0.0"/>
    <parent link="base"/>
    <child link="wheel_right_link"/>
    <axis xyz="0.0 0.0 1.0"/>
</joint>



<link name="wheel_left_link">
```

```
    <inertial>
       <mass value="2" />
       <inertia ixx="0.01" ixy="0.0" ixz="0"
          iyy="0.01" iyz="0" izz="0.01" />
    </inertial>

    <visual>
      <geometry>
         <cylinder radius="0.15" length="0.1"/>
      </geometry>
      <material name="blue">
        <color rgba="0 0 1 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
         <cylinder radius="0.15" length="0.1"/>
      </geometry>
      <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
    </collision>
</link>

<joint name="wheel_left_joint" type="continuous">
    <origin xyz="0.2 -0.25 0.0" rpy="1.57 0.0 0.0"/>
    <parent link="base"/>
    <child link="wheel_left_link"/>
    <axis xyz="0.0 0.0 1.0"/>
</joint>



<link name="caster">
    <inertial>
          <mass value="1" />
          <inertia ixx="0.01" ixy="0.0" ixz="0"
          iyy="0.01" iyz="0" izz="0.01" />
    </inertial>

    <visual>
      <geometry>
        <sphere radius=".08" />
      </geometry>
      <material name="white" />
    </visual>

    <collision>
      <origin/>
      <geometry>
        <sphere radius=".08" />
      </geometry>
```

```
    </collision>
</link>

<joint name="caster_joint" type="continuous">
    <origin xyz="-0.3 0.0 -0.07" rpy="0.0 0.0 0.0"/>
    <axis xyz="0 0 1" />
    <parent link="base"/>
    <child link="caster"/>
</joint>

<link name="camera">
    <inertial>
        <mass value="0.5" />
        <inertia ixx="0.01" ixy="0.0" ixz="0"
        iyy="0.01" iyz="0" izz="0.01" />
    </inertial>

    <visual>
        <geometry>
            <box size="0.1 0.1 0.1"/>
        </geometry>
        <material name="red">
            <color rgba="1 0 0 1"/>
        </material>
    </visual>

    <collision>
        <geometry>
            <box size="0.1 0.1 0.1"/>
        </geometry>
    </collision>
</link>


<joint name="camera_joint" type="fixed">
    <origin xyz="-0.32 0 0.1" rpy="0 0.0 3.14"/>
    <parent link="base"/>
    <child link="camera"/>
    <axis xyz="0.0 0.0 1.0"/>
</joint>




<!--http://wiki.ros.org/simulator_gazebo/Tutorials/ListOfMaterials-->
<gazebo reference="base">
    <material>Gazebo/WhiteGlow</material>
</gazebo>
<gazebo reference="wheel_left_link">
    <material>Gazebo/SkyBlue</material>
```

```
</gazebo>
<gazebo reference="wheel_right_link">
   <material>Gazebo/SkyBlue </material>
</gazebo>
<gazebo reference="caster">
   <material>Gazebo/Grey</material>
</gazebo>
<gazebo reference="camera">
   <material>Gazebo/Blue</material>
</gazebo>



<!-- differential robot-->
<gazebo>
   <plugin filename="libgazebo_ros_diff_drive.so" name="gazebo_base_controller">
     <odometry_frame>odom</odometry_frame>
     <commandTopic>cmd_vel</commandTopic>
     <publish_odom>true</publish_odom>
     <publish_odom_tf>true</publish_odom_tf>
     <update_rate>15.0</update_rate>

     <left_joint>wheel_left_joint</left_joint>
     <right_joint>wheel_right_joint</right_joint>

     <wheel_separation>0.5</wheel_separation>
     <wheel_diameter>0.3</wheel_diameter>
     <max_wheel_acceleration>0.7</max_wheel_acceleration>
     <max_wheel_torque>8</max_wheel_torque>
     <robotBaseFrame>base</robotBaseFrame>
   </plugin>
</gazebo>

<!-- camera plugin-->

<gazebo reference="camera">
   <sensor type="camera" name="camera1">
     <visualize>true</visualize>
     <update_rate>30.0</update_rate>
     <camera name="head">
       <horizontal_fov>1.3962634</horizontal_fov>
       <image>
         <width>512</width>
         <height>512</height>
         <format>R8G8B8</format>
       </image>
       <clip>
         <near>0.02</near>
         <far>500</far>
       </clip>
     </camera>
```

```
        <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
          <alwaysOn>true</alwaysOn>
          <updateRate>0.0</updateRate>
          <cameraName>/camera</cameraName>
          <imageTopicName>image_raw</imageTopicName>
          <cameraInfoTopicName>camera_info</cameraInfoTopicName>
          <frameName>camera_link</frameName>
          <hackBaseline>0.07</hackBaseline>
        </plugin>
      </sensor>
      <material>Gazebo/Blue</material>
    </gazebo>

</robot>
```

Create a launch folder
Create a launch file rviz.launch.py

```
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    package_dir = '/home/asha/ros2_ws/src/lab6/urdf'
    urdf = os.path.join(package_dir,'car.urdf')

    return LaunchDescription([
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            arguments=[urdf]),

        Node(
            package='joint_state_publisher_gui',
            executable='joint_state_publisher_gui',
            name='joint_state_publisher_gui',
            arguments=[urdf]),

        Node(
            package='rviz2',
            executable='rviz2',
            name='rviz2',
            output='screen'),
        ])
```

Create a launch folder
Create a launch file gazebo.launch.py inside the launch folder

```
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, ExecuteProcess
from launch.substitutions import LaunchConfiguration
from launch_ros.actions import Node
from launch.launch_description_sources import PythonLaunchDescriptionSource

def generate_launch_description():
  urdf = '/home/asha/ros2_ws/src/lab6/urdf/car.urdf'
  return LaunchDescription([
    #   publishes TF for links of the robot without joints
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            arguments=[urdf]),
       # publish TF for Joints only links
        Node(
            package='joint_state_publisher',
            executable='joint_state_publisher',
            name='joint_state_publisher',
            output='screen',
            ),
        # open gazebo
        ExecuteProcess(
            cmd=['gazebo', '--verbose', '-s', 'libgazebo_ros_factory.so'],
            output='screen'),
        Node(
            package='gazebo_ros',
            executable='spawn_entity.py',
            name='urdf_spawner',
            output='screen',
            arguments=["-topic", "/robot_description", "-entity", "lab6"])
  ])
```

- **Height and width**: These are the dimensions in camera pixels. In this case, it's 512 **x 512**.
- **Encoding**: How these pixels are encoded. This means what each value in the data array will mean. In this case, it's **rgb8**. This means that the data values will be a color value represented as red/green/blue in 8-bit integers.
- **Data**: The image data.

Terminal 1:
ros2 launch lab6 rviz.launch.py
ros2 launch lab6 gazebo.launch.py

ctrl + c to kill the rviz and gazebo window

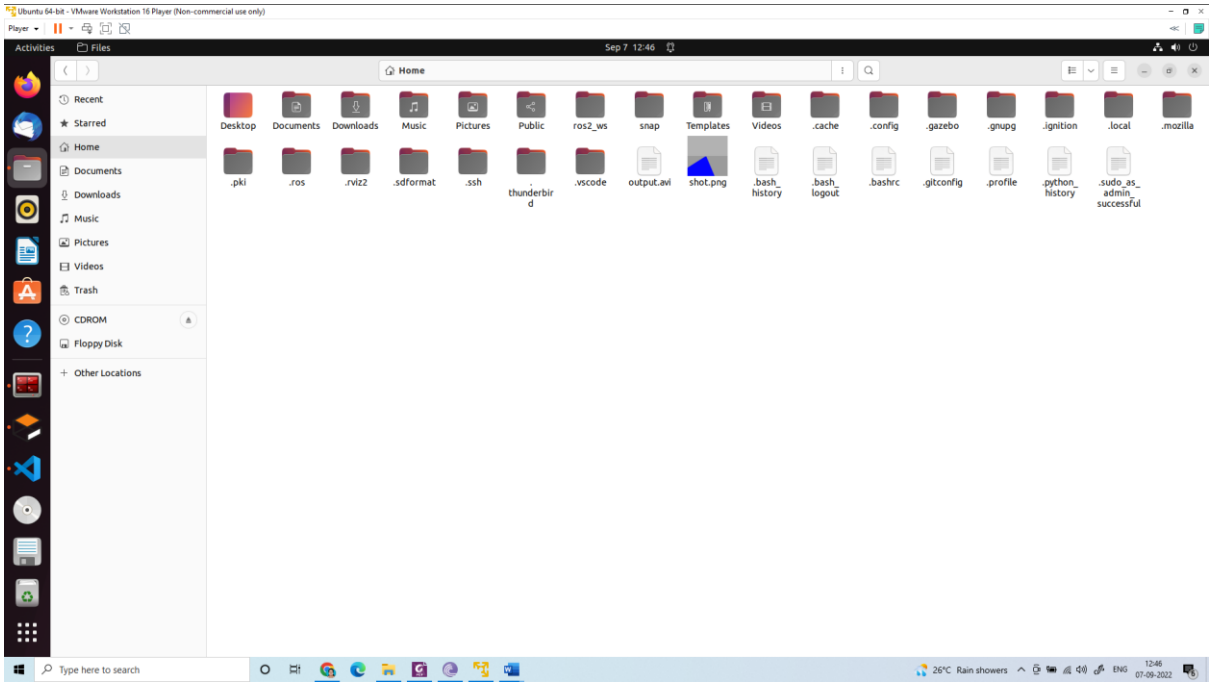
create a file capture_image.py inside the lab 6 folder

```
import rclpy
import cv2
from rclpy.node import Node
from cv_bridge import CvBridge
from sensor_msgs.msg import Image

class Capture(Node):
  def __init__(self):
    super().__init__('video_subscriber')
    self.subscriber =
self.create_subscription(Image,'/camera1/image_raw',self.process_data,10)
    self.out =
#cv2.VideoWriter('/home/asha/output.avi',cv2.VideoWriter_fourcc('M','J','P','G'), 10,
(512,512))
    self.bridge = CvBridge()

  def process_data(self, data):

    frame = self.bridge.imgmsg_to_cv2(data)
    self.out.write(frame)
    self.img = cv2.imwrite('/home/asha/shot.png', frame)
    cv2.imshow("output", frame)
    cv2.waitKey()
    cv2.destroyAllWindows()


def main(args=None):
  rclpy.init(args=args)
  node = Capture()
  rclpy.spin(node)
  rclpy.shutdown()

if __name__ == '__main__':
  main()
```

Terminal 1:
ros2 launch lab6 gazebo.launch.py
draw a line and place the robot on the line (insert→yellow line)

Terminal 2:
ros2 run lab6 capture

Press ctrl + c to capture the image of the road

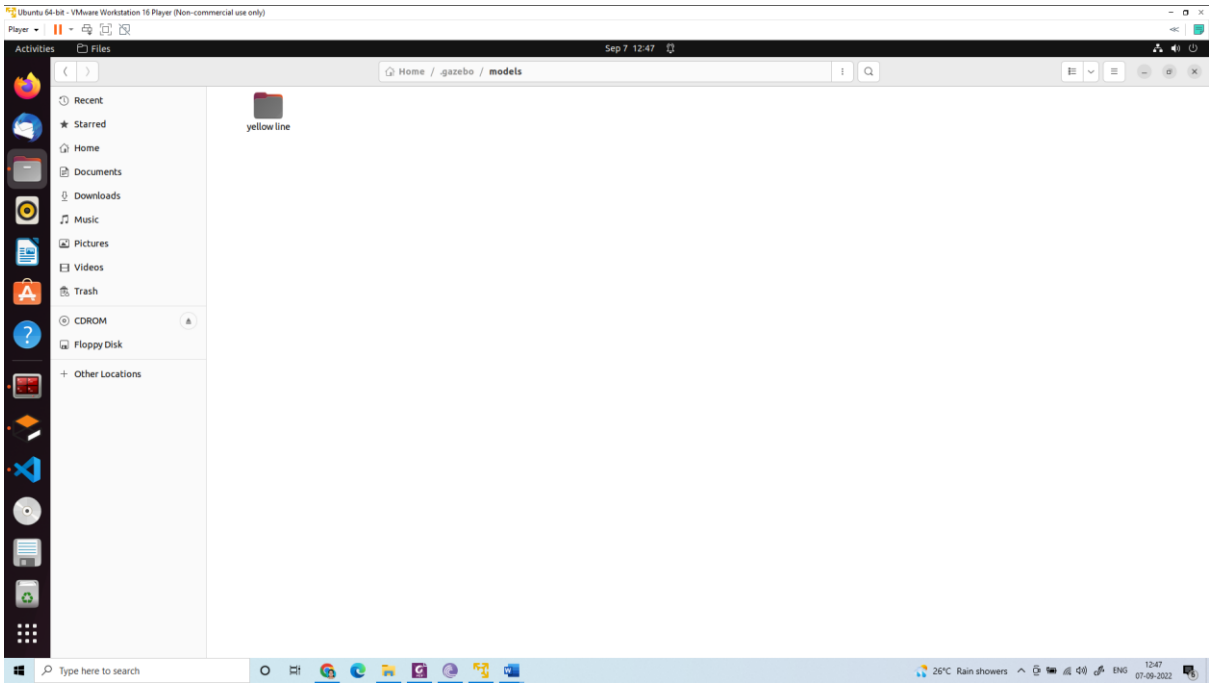


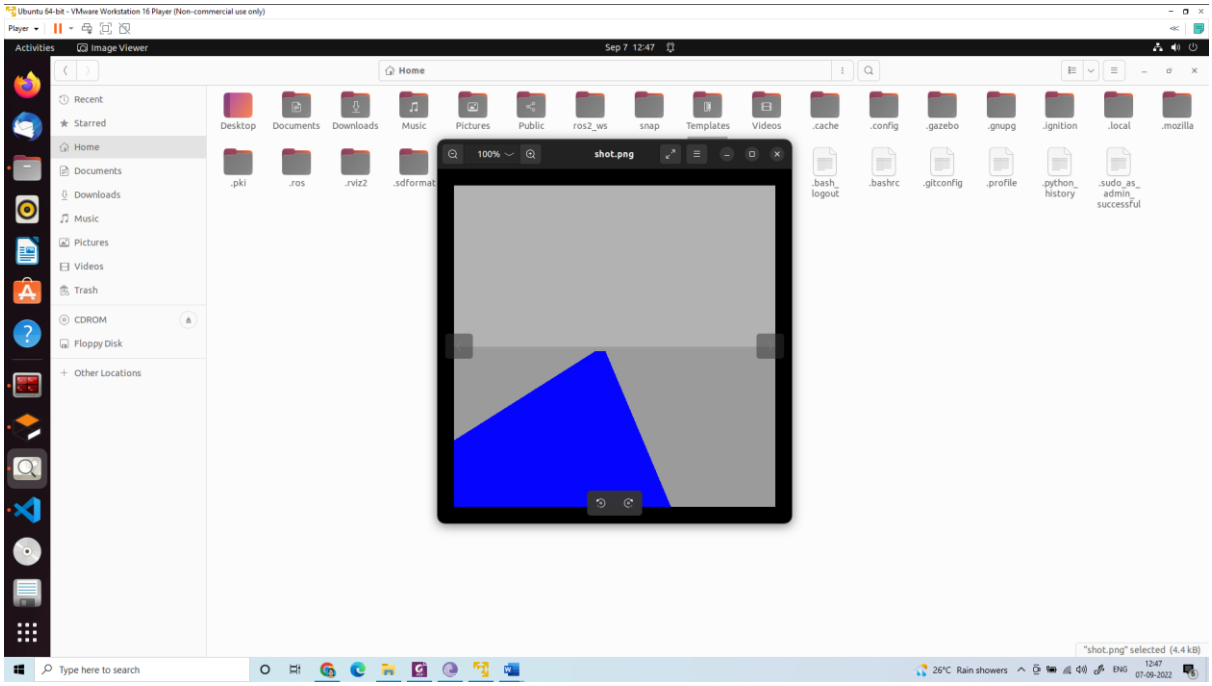
create a file extract_road.py inside the lab 6 folder

```
import cv2
import numpy

image = cv2.imread('/home/asha/shot.png')

def mouse(event,x,y,flags,param):
        if event==cv2.EVENT_LBUTTONDOWN:
                h=image[y,x,0]
                s=image[y,x,1]
                v=image[y,x,2]
                print("H:",h)
                print("S:",s)
                print("V:",v)

cv2.namedWindow('mouse')
cv2.setMouseCallback('mouse',mouse)


cv2.imshow("original image", image)
cv2.imshow("mouse", image)
cv2.waitKey(0)
cv2.destroyAllWindows()


light_line = numpy.array([250,0,0])
dark_line = numpy.array([255,10,10])
mask = cv2.inRange(image, light_line,dark_line)
cv2.imshow('mask', mask)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()

canny= cv2.Canny(mask,30,5)
cv2.imshow('edge', canny)
cv2.waitKey(0)
cv2.destroyAllWindows()
print(canny.shape)

r1=200;c1=0
img = canny[r1:r1+200,c1:c1+512]
cv2.imshow('crop', img)
cv2.waitKey(0)
cv2.destroyAllWindows()

edge=[]
row =150

for i in range (512):
    if(img[row,i]==255):
        edge.append(i)
print(edge)


if(len(edge)==4):
    left_edge=edge[0]
    right_edge=edge[2]
    print(edge)
if(len(edge)==3):
    if(edge[1]-edge[0] > 5):
        left_edge=edge[0]
        right_edge=edge[1]
    else:
        left_edge=edge[0]
        right_edge=edge[2]

road_width=(right_edge-left_edge)
frame_mid = left_edge + (road_width/2)
mid_point = 512/2
img[row,int(mid_point)]=255
print(mid_point)
error=mid_point-frame_mid

if(error < 0):
    action="Go Right"
else :
    action="Go Left"

print("error", error)


img[row,int(frame_mid)]=255
```

```
print("mid point of the frame", frame_mid)


f_image = cv2.putText(img, action, (50,50), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255,0,0), 1, cv2.LINE_AA)
cv2.imshow('final image',f_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





Create a line_follow.py inside the folder lab 6

```
#!/usr/bin/env python3

import sys
import cv2
import numpy
import rclpy
from rclpy.node import Node
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
from geometry_msgs.msg import Twist

class LineFollower(Node):
    def __init__(self):
        super().__init__('line_follower')
        self.bridge = CvBridge()
        self.subscriber =
self.create_subscription(Image,'/camera1/image_raw',self.process_data, 10)
        self.publisher = self.create_publisher(Twist, '/cmd_vel', 40)
        timer_period = 0.2
        self.timer = self.create_timer(timer_period, self.send_cmd_vel)
        self.velocity=Twist()
        self.empty = False
        self.error = 0
        self.action=""
        self.get_logger().info("Node Started!")

    def send_cmd_vel(self):

        if(self.empty):
            self.velocity.linear.x=0.0
            self.velocity.angular.z= 0.0
            self.action="Stop"

        else:
            if(self.error > 0):
                self.velocity.linear.x=0.1
                self.velocity.angular.z=0.1
                self.action="Go Left"
            elif(self.error < 0):
                self.velocity.linear.x=0.1
                self.velocity.angular.z=-0.1
                self.action="Go Right"
            elif(self.error==0):
                self.velocity.linear.x=0.1
                self.velocity.angular.z= 0.0
                self.action="Go Straight"
```

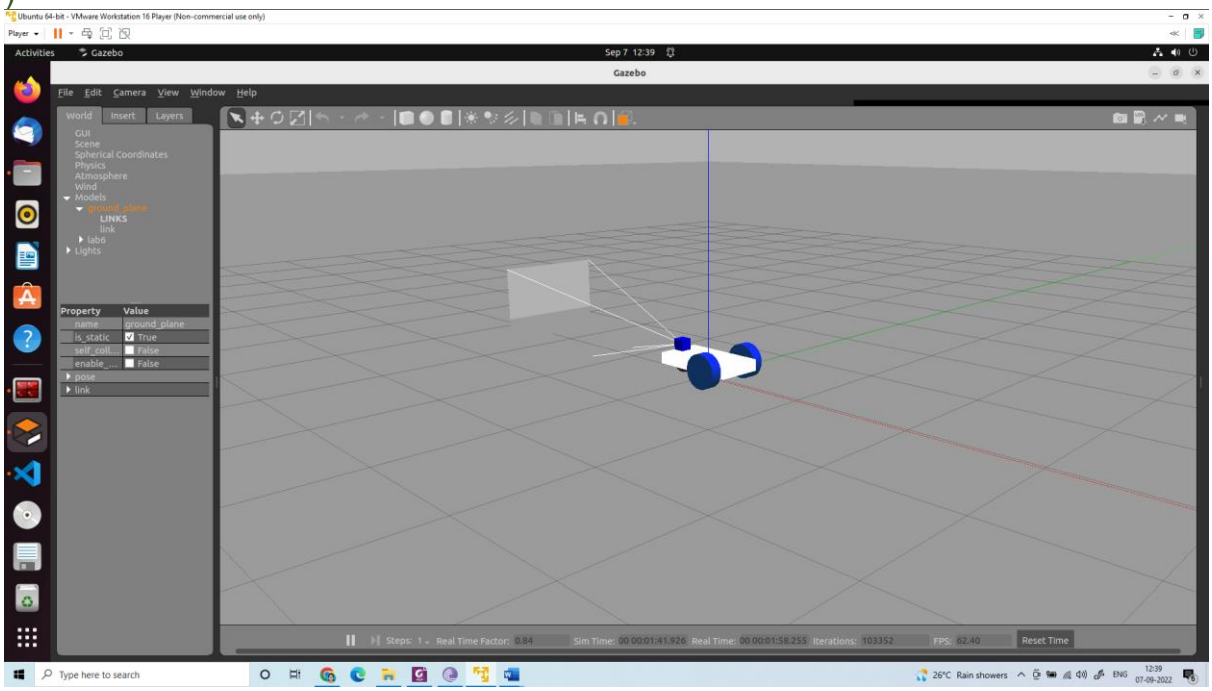
```
    self.publisher.publish(self.velocity)


## Subscriber Call Back

def process_data(self, data):
    self.get_logger().info("Image Received!")
    frame = self.bridge.imgmsg_to_cv2(data)
    light_line = numpy.array([250,0,0])
    dark_line = numpy.array([255,10,10])
    mask = cv2.inRange(frame, light_line,dark_line)
    cv2.imshow('mask', mask)


    canny= cv2.Canny(mask,30,5)
    cv2.imshow('edge', canny)


    r1=200;c1=0
    img = canny[r1:r1+200,c1:c1+512]
    cv2.imshow('crop', img)


    edge=[]
    row =150

    for i in range(512):
        if(img[row,i]==255):
            edge.append(i)
    print(edge)

    if(len(edge)==0):
        left_edge=512//2
        right_edge=512//2
        self.empty = True

    if(len(edge)==1):
        if edge[0]>512//2:
            left_edge=0
            right_edge=edge[0]
            self.empty = False
        else:
            left_edge=edge[0]
            right_edge=512
            self.empty = False


    if(len(edge)==2):
        left_edge=edge[0]
```

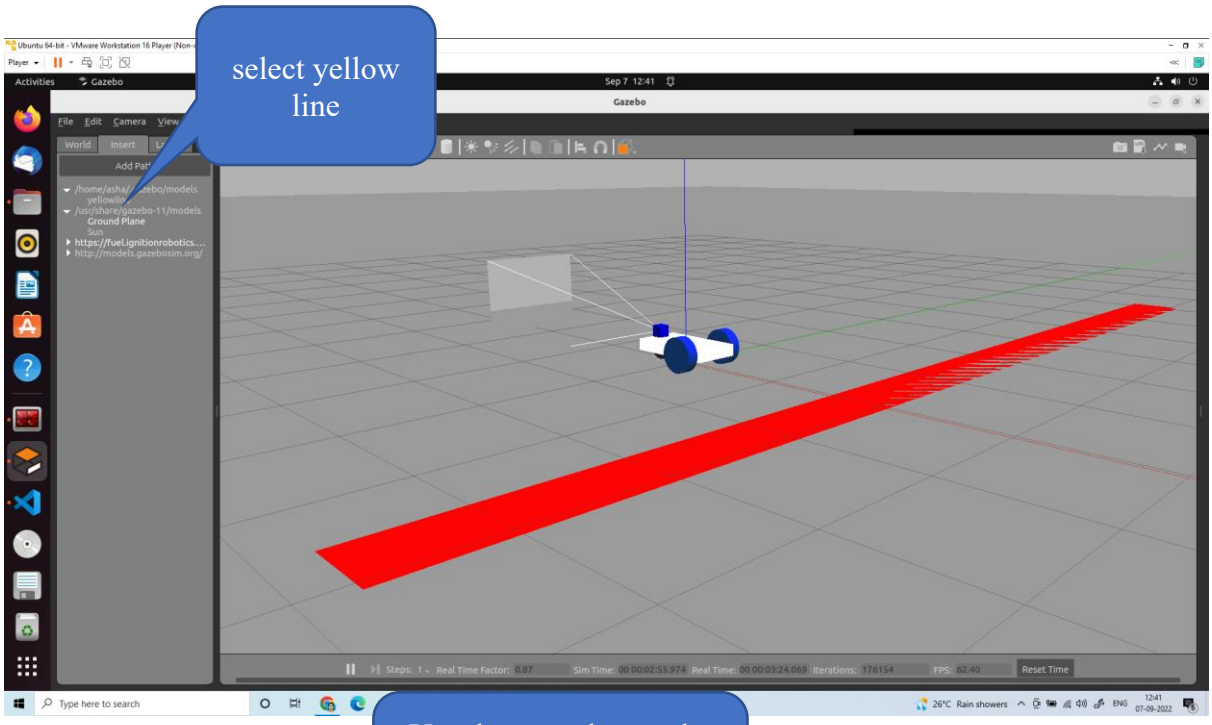
```
            right_edge=edge[1]
            self.empty = False

        if(len(edge)==3):
            if(edge[1]-edge[0]>5):
                left_edge=edge[0]
                right_edge=edge[1]
                self.empty = False
            else:
                left_edge=edge[0]
                right_edge=edge[2]
                self.empty = False

        if(len(edge)==4):
            left_edge=edge[0]
            right_edge=edge[2]
            self.empty = False

        if(len(edge)>=5):
            left_edge=edge[0]
            right_edge=edge[len(edge)-1]
            self.empty = False


        road_width=(right_edge-left_edge)
        frame_mid = left_edge + (road_width/2)
        mid_point = 512/2
        img[row,int(mid_point)]=255
        print(mid_point)
        self.error=mid_point-frame_mid
        img[row,int(frame_mid)]=255
        print(self.action)
        f_image = cv2.putText(img, self.action, (100,100), cv2.FONT_HERSHEY_SIMPLEX,
1, (255,0,), 2, cv2.LINE_AA)

def main(args=None):
  rclpy.init(args=args)
  node = LineFollower()
  rclpy.spin(node)
  rclpy.shutdown()

if __name__ == '__main__':
  main()
```

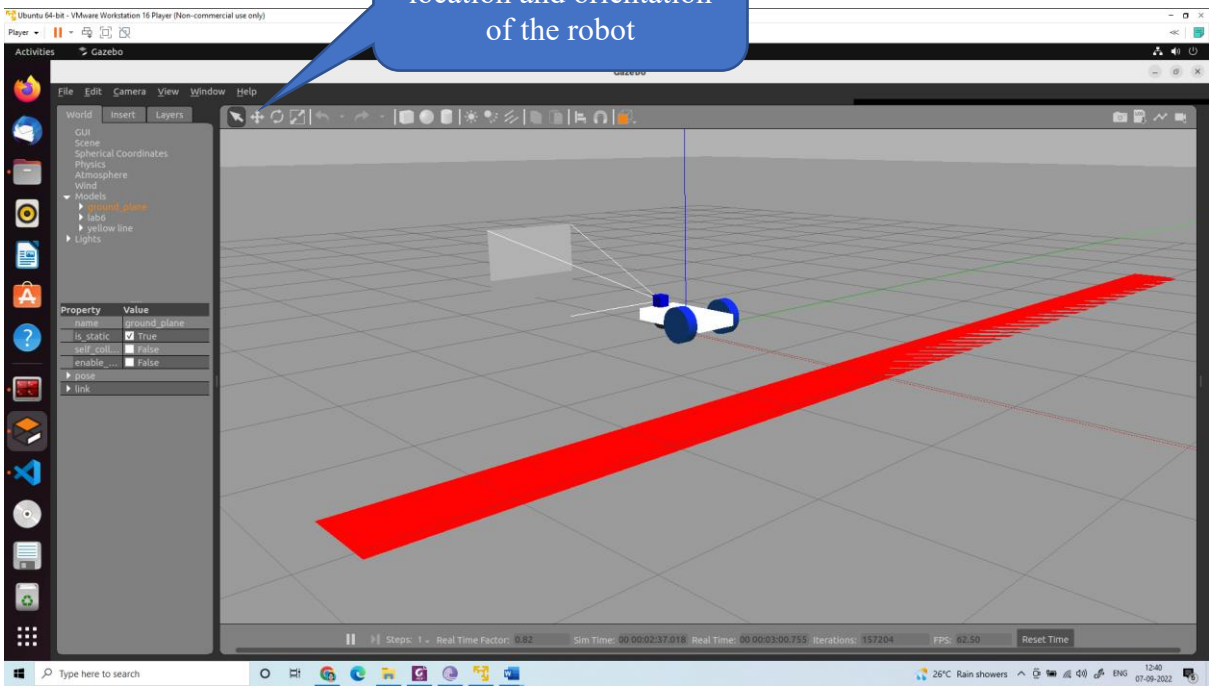
edit setup.py file

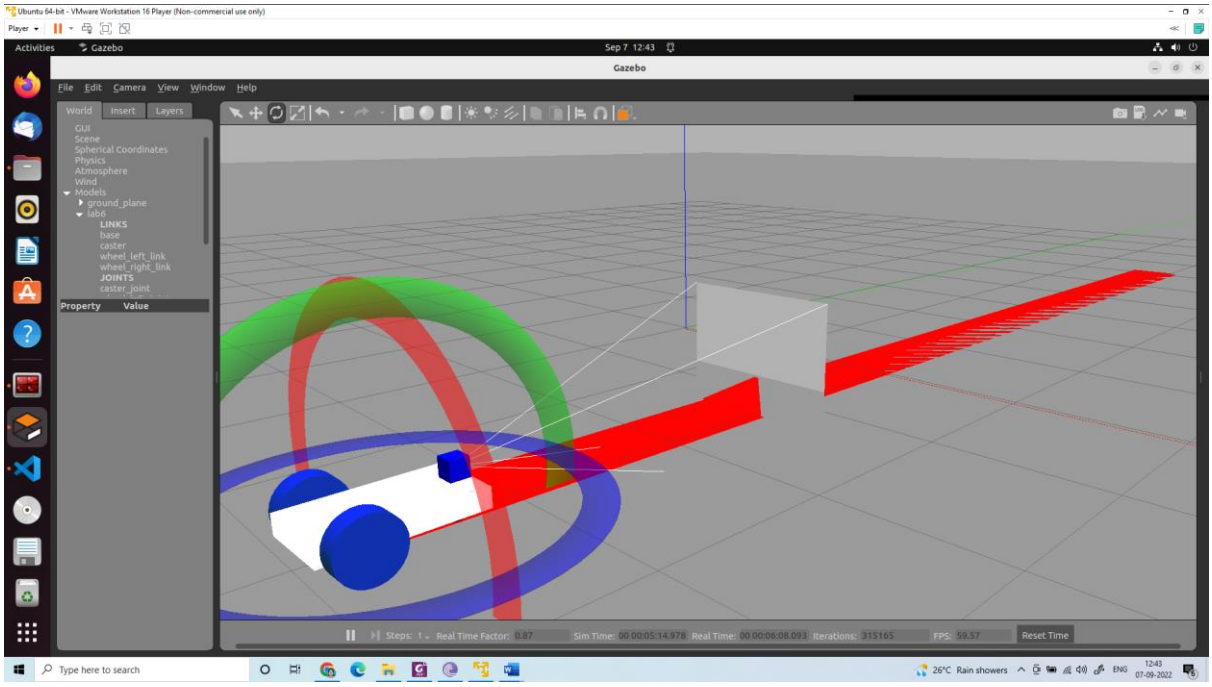
```
from setuptools import setup
import os
```

```
from glob import glob

package_name = 'lab6'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launch/*')),
        (os.path.join('share', package_name), glob('urdf/*'))
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='asha',
    maintainer_email='asha.cs12@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'capture = lab6.capture_image:main',
            'line = lab6.line_follow:main'
        ],
    },
)
```




select yellow line

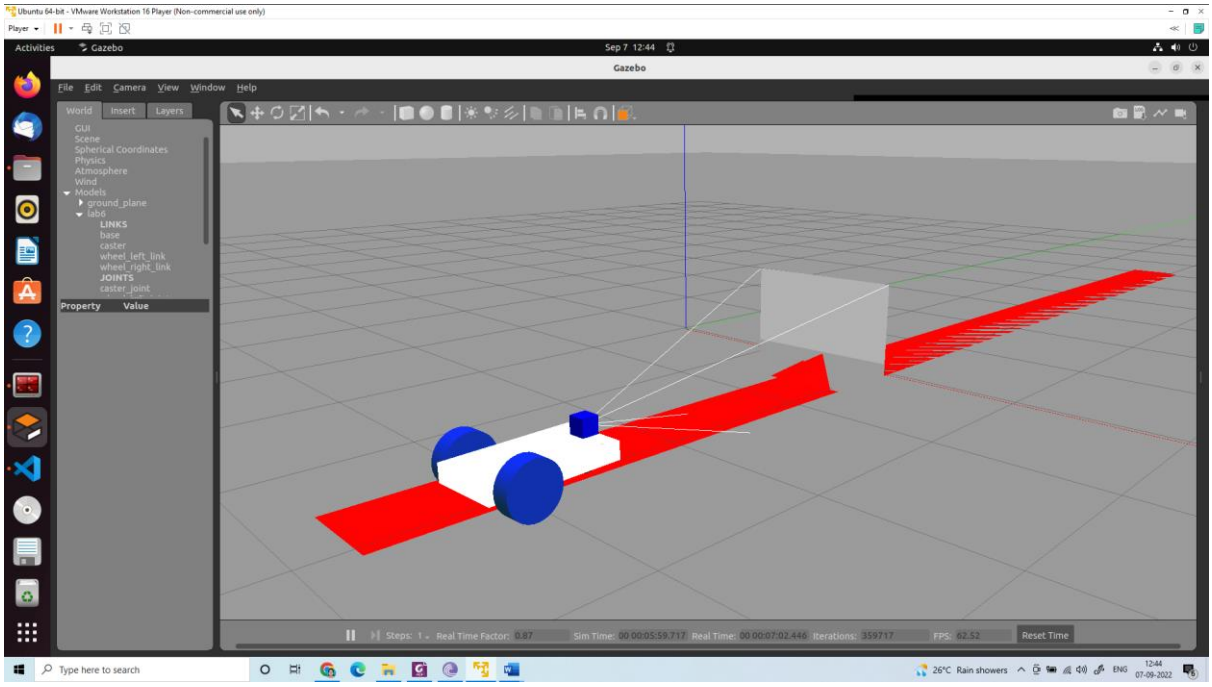

Use these to change the location and orientation of the robot

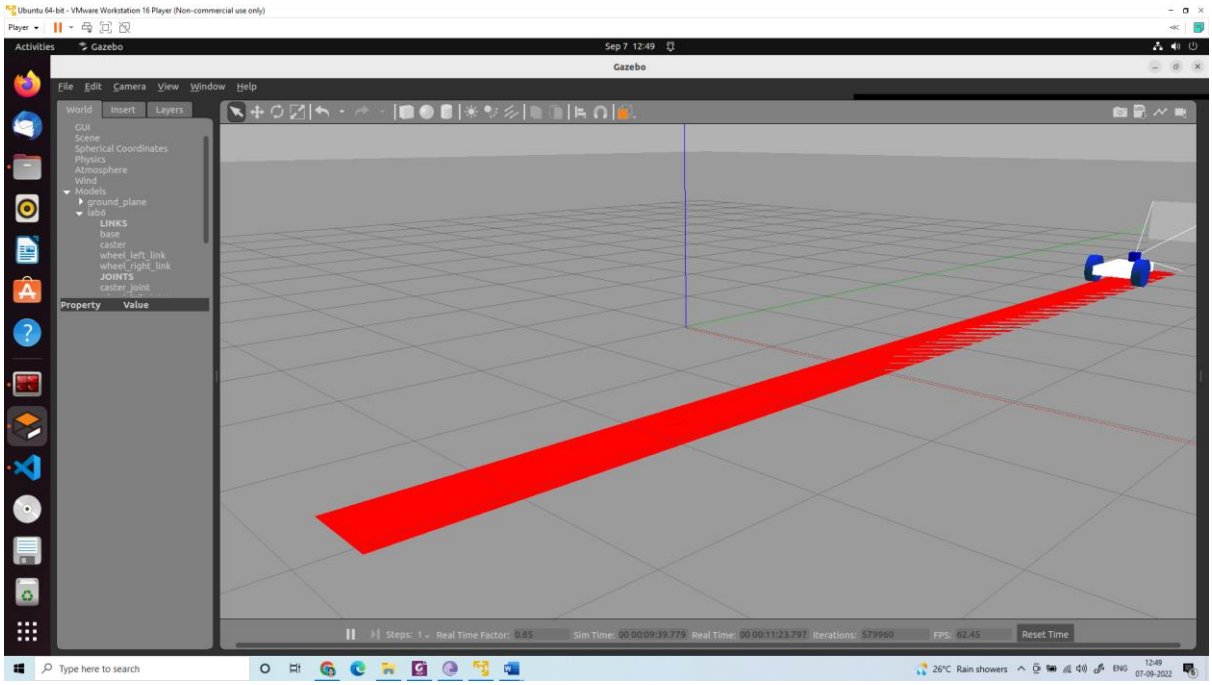


Terminal 1:
ros2 launch lab6 gazebo.launch.py

Terminal 2:
ros2 topic list

Terminal 3:
ros2 run lab6 line





Create a red line

Create a world folder
Create a yellow line folder
Supplementary material:

Create a model.config

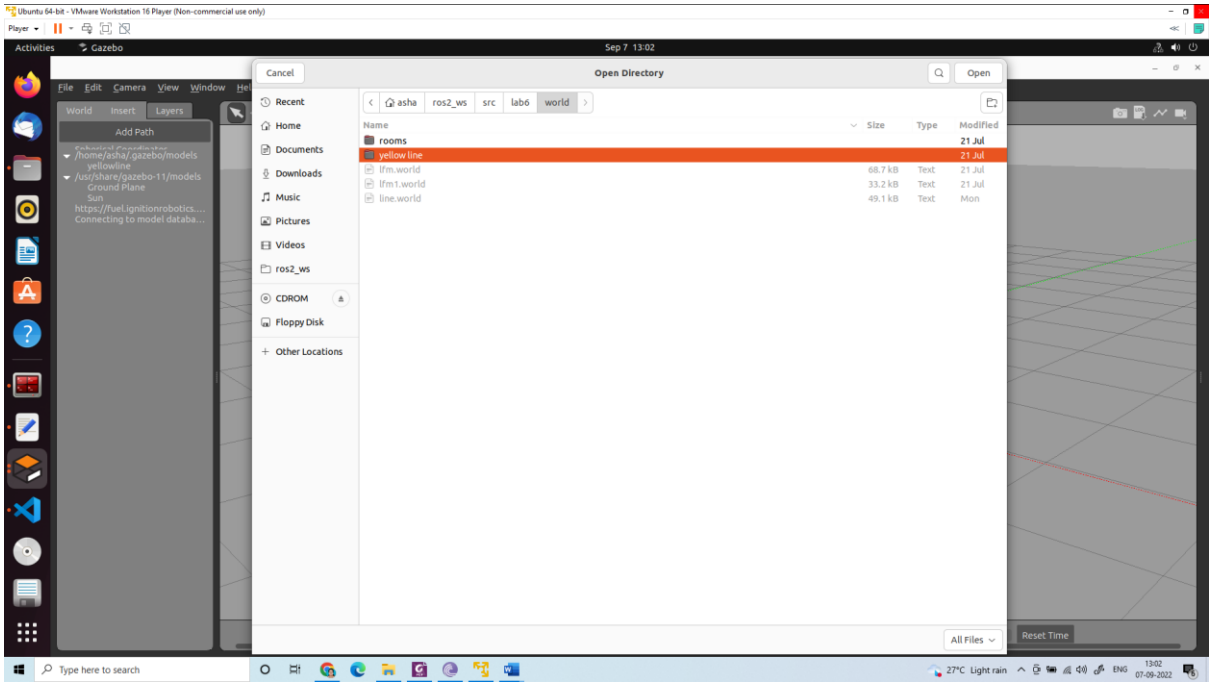
```
<?xml version="1.0" ?>
<model>
   <name>yellowline</name>
   <version>1.0</version>
   <sdf version="1.6">model.sdf</sdf>
   <author>
      <name></name>
      <email></email>
   </author>
   <description></description>
</model>
```

Create a model.sdf

```
<?xml version="1.0"?>
<sdf version="1.6">
<model name="yellow line">
  <static>true</static>
    <link name="link_ground">
      <collision name="collision">
```

```
      <geometry>
        <plane>
          <normal>0 0 1</normal>
          <size>0.1 3.2</size>
        </plane>
      </geometry>

      <surface>
        <friction>
          <ode>
            <mu>100</mu>
            <mu2>50</mu2>
          </ode>
        </friction>
      </surface>
    </collision>
    <visual name="visual_ground">
      <cast_shadows>false</cast_shadows>
      <geometry>
        <plane>
          <normal>0 0 1</normal>
          <size>0.5 10</size>
        </plane>
      </geometry>
     <material>
        <script>
          <uri>file://media/materials/scripts/gazebo.material</uri>
          <name>Gazebo/Red</name>
        </script>
      </material>
    </visual>
  </link>
 </model>
</sdf>
```

Copy the folder into /home/asha/.gazebo/models/
In the gazebo window (gazebo)
Insert→/home/asha/.gazebo/models/→yellow line→all files
close gazebo (killall 9 gzserver) and open again (gazebo)



Insert→/home/asha/.gazebo/models/→yellow line

Exercise: Write a code to track the red ball in the gazebo simulation.

ROS2 control for Turtlebot 2

sudo apt-get install ros-foxy-teleop-twist-keyboard
sudo apt-get install ros-foxy-joint-state-publisher
sudo apt-get install ros-foxy-xacro
sudo apt-get install ros-foxy-kobuki-*

ros2 launch turtlebot_interface interface.launch.py
ros2 run teleop_twist_keyboard teleop_twist_keyboard
sudo apt-get update && sudo apt-get upgrade -y && sudo apt-get dist-upgrade -y

Open Terminal

hostname -I
or ifconfig

note down the IP address

From a remote computer you can connect to turtlebot Laptop

Using PuTTY (in windows)

Reminna (in Ubuntu)

ssh username@ipaddress (in terminal)

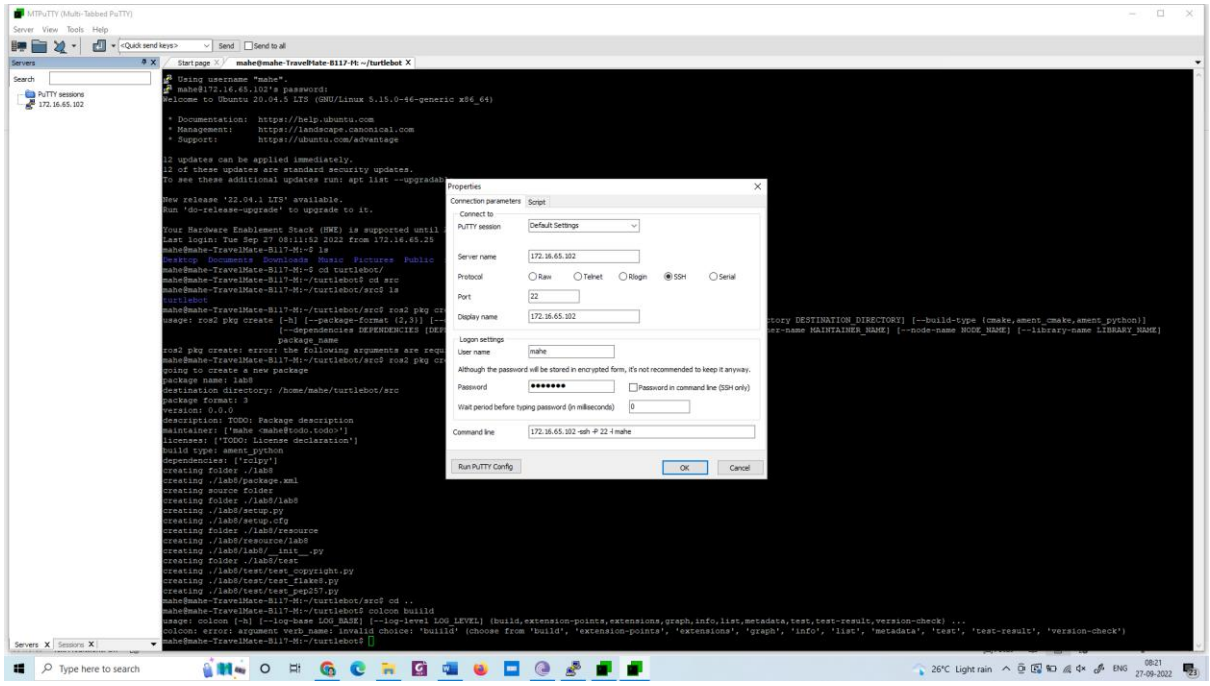
Connect camera USB and Kubuki USB and switch on Kubuki.
Switch on the netbook.
Use Remmina to connect to turtlebot from Ubuntu/mtputty from windows
See that the turtlebot laptop and remote laptop are connected to same WiFi
Find the address of netbook placed on turtlebot
Open terminal and type
$ifconfig
Copy the address inet addr: ----(For example 172.16.65.109)
Open Remmina
Establish SSH connection
Type Name (for example turtlebot)
Server Name: 172.16.65.109
User name: mahe or robolab
Password:robolab

Open the terminal and type

$ ros2 launch turtlebot_interface interface.launch.py

Open another terminal and type

$ ros2 run teleop_twist_keyboard teleop_twist_keyboard



cd turtlebot
cd src
ros2 pkg create lab8 –build-type ament_python –dependencies rclpy
cd ..
colcon build
Open lab8 using visual studio
Create a python file move_robot.py inside lab8 folder

```
#!/usr/bin/env python
import rclpy
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
from rclpy.node import Node
import sys



class MoveRobot(Node):
    def __init__(self):
        super().__init__("move_robot")
```

```
        self.lin_vel = 0.1
        self.ang_vel = 0.0
        self.distance = 1.0
        self.publisher = self.create_publisher(Twist, "/cmd_vel", 10)
        self.subscriber = self.create_subscription(Odometry, "odom", self.control_loop,
10)

    def control_loop(self, msg):
        X=msg.pose.pose.position.x
        print("position", X)
        vel = Twist()
        if abs(X) < self.distance:
            vel.linear.x = self.lin_vel
            vel.angular.z = 0.0
        else:
            vel.linear.x = 0.0
            vel.angular.z = 0.0
        print('speed : {}'.format(vel))
        self.publisher.publish(vel)

def main(args=None):
    rclpy.init(args=args)
    node = MoveRobot()
    rclpy.spin(node)
    rclpy.shutdown()


if __name__ == "__main__":
          main()
```

Edit setup.py as follows

```
from setuptools import setup

package_name = 'gotogoal'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
```

```
    zip_safe=True,
    maintainer='mahe',
    maintainer_email='mahe@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'move = lab8.move_robot:main'
        ],
    },
)
```

```
cd ~/turtlebot/
colcon build
```

Terminal 1:

```
ros2 launch turtlebot_interface interface.launch.py
```

Terminal 2:

```
ros2 topic list
```

Terminal 2:

```
ros2 run lab8 move
```

Example 2:

```
#!/usr/bin/env python
import rclpy
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
from rclpy.node import Node
import math
import time
from std_srvs.srv import Empty
import sys



class MoveRobot(Node):
    def __init__(self):
        super().__init__("move_robot")
        self.lin_vel = 0.1
        self.ang_vel = 0.0
        self.distance = 1.0
        self.publisher = self.create_publisher(Twist, "/cmd_vel", 10)
        self.move(0.1,5,True)
        time.sleep(2)
```

```
        self.rotate(30,125,False)
        self.stop()

    def move(self, speed, time, is_forward):
        t0= self.get_clock().now()
         self.velocity = Twist()
         if(is_forward):
              self.velocity.linear.x = abs(speed)
              self.get_logger().info("Turtlebot moving forward")

         else:
             self.velocity.linear.x =-abs(speed)
             self.get_logger().info("Turtlebot moving backward")
             t1= self.get_clock().now()
             if (t1-t0)>time:
                  self.get_logger().info("Time closed")
                  self.get_logger().warn("Stopping the robot")
                  self.velocity.linear.x =0

         self.publisher.publish(self.velocity)

def stop(self):
        self.velocity = Twist()
         self.velocity.linear.x=0
         self.publisher.publish(self.velocity)

def rotate(self, ang_speed_deg,relative,speed_deg,clockwise):
self.velocity = Twist()
         self.velocity.linear.x=0
         ang_speed=math.radians(abs(ang_speed_deg))
         if(clockwise):
                self.velocity.angular.z=-abs(ang_speed)
         else:
                self.velocity.angular.z=abs(ang_speed)
        angle_moved = 0
        t0= self.get_clock().now()
        while(True):
                self.publisher.publish(self.velocity)
                self.get_logger().info("Turtlebot ratates")
                t1= self.get_clock().now()
                current_ang = (t1-t0)*ang_speed_degree
                if(current_ang > relative_speed_deg):
                        self.get_logger().info("Reached")
                        break
         self.velocity.angular.z=0
         self.publisher.publish(self.velocity)
```

```
def main(args=None):
    rclpy.init(args=args)
    node = MoveRobot()
    rclpy.spin(node)
    rclpy.shutdown()

if __name__ == "__main__":
        main()
```

UR5 control using ROS2

sudo apt-get update && sudo apt-get upgrade -y && sudo apt-get dist-upgrade -y

sudo apt-get install python3-colcon-ros

sudo apt-get install python3-colcon-bash

sudo apt update && sudo apt install curl gnupg2 lsb-release

sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(source /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

sudo apt update

sudo apt-get upgrade

sudo apt install ros-foxy-desktop
sudo apt install ros-foxy-moveit
sudo apt install ros-foxy-ros2-control
sudo apt install ros-foxy-rqt-service-caller
sudo apt install ros-foxy-warehouse-ros-mongo

mkdir ur5
cd ur5
mkdir ros2
cd ros2
mkdir src
cd src

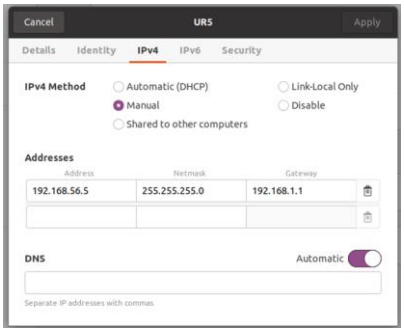
git clone --recurse-submodules -j8 https://github.com/ashacs2/ur5_interface.git -b ros2
git clone --recurse-submodules -j8 https://github.com/ashacs2/ur5_ros2_interface.git -b ros2
gedit ~/.bashrc
source ~/ur5/ros2/install/setup.bash

cd ur5/ros2/

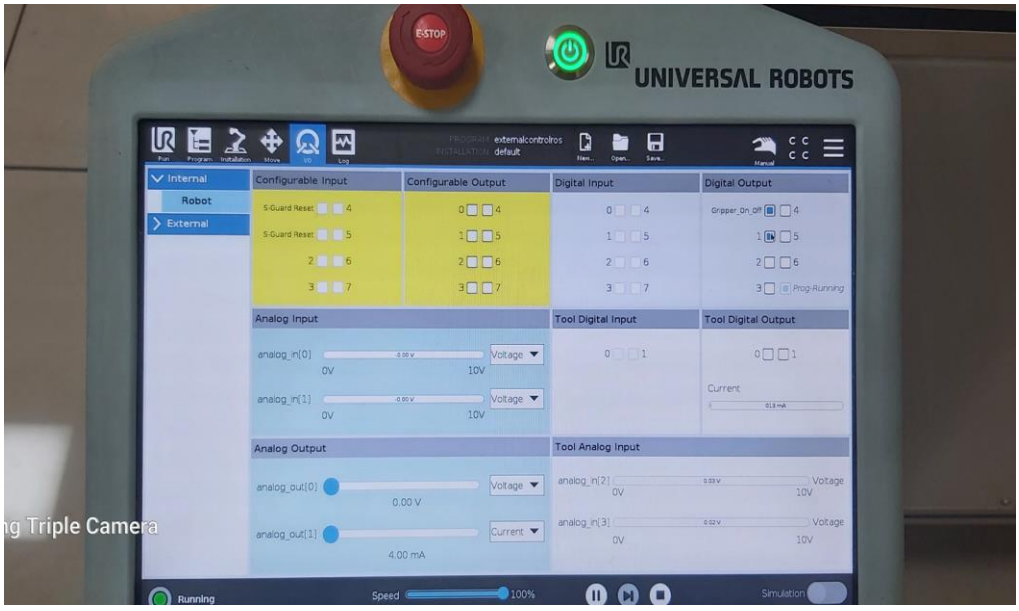
rosdep install --from-paths src --ignore-src -r -y

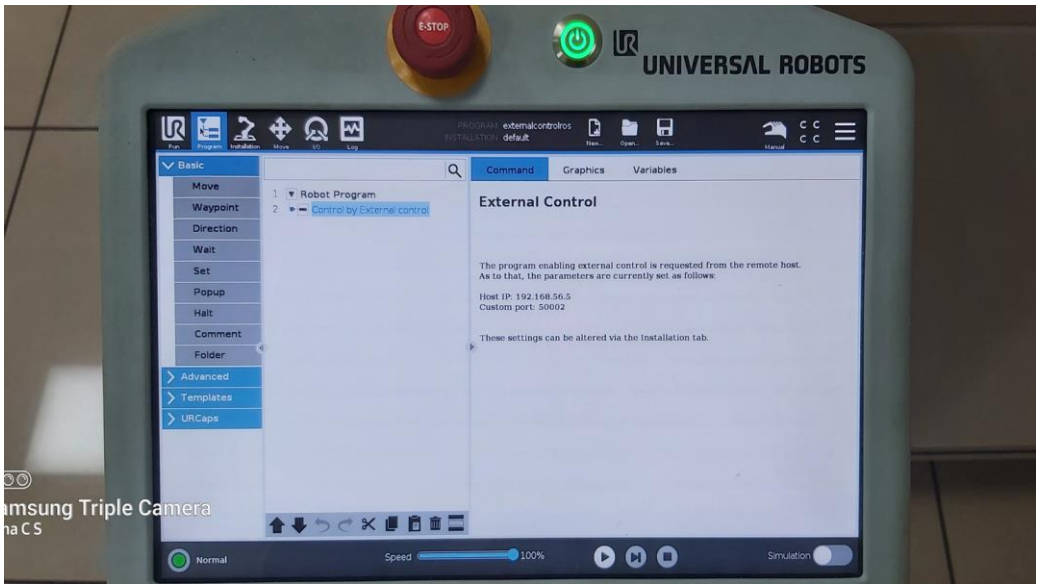
cd ..
colcon build --symlink-install --cmake-args -
DCMAKE_EXPORT_COMPILE_COMMANDS=1

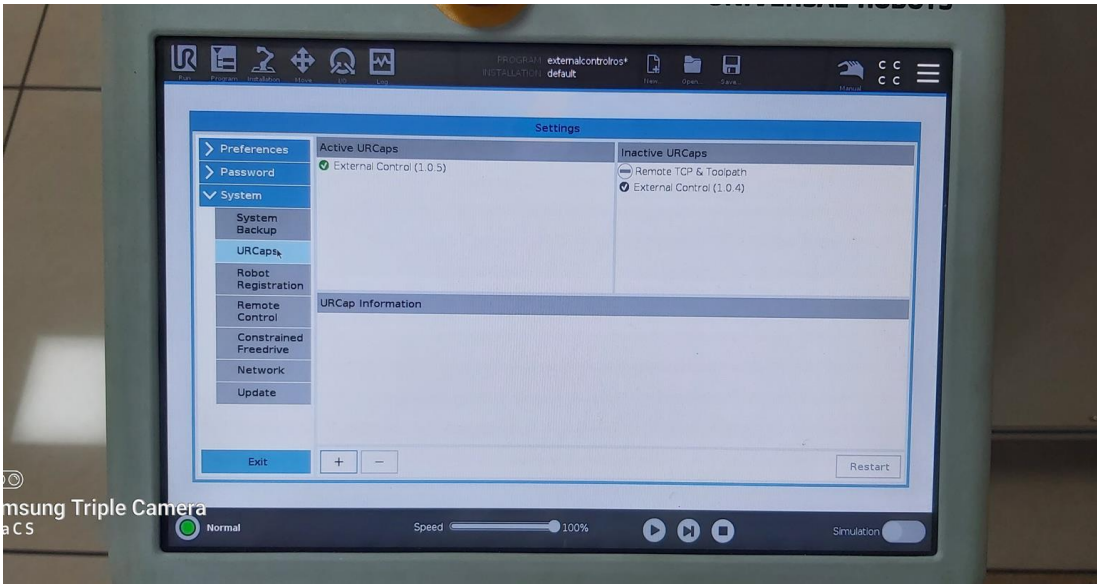
Connect an ethernet cable from UR5 to PC.

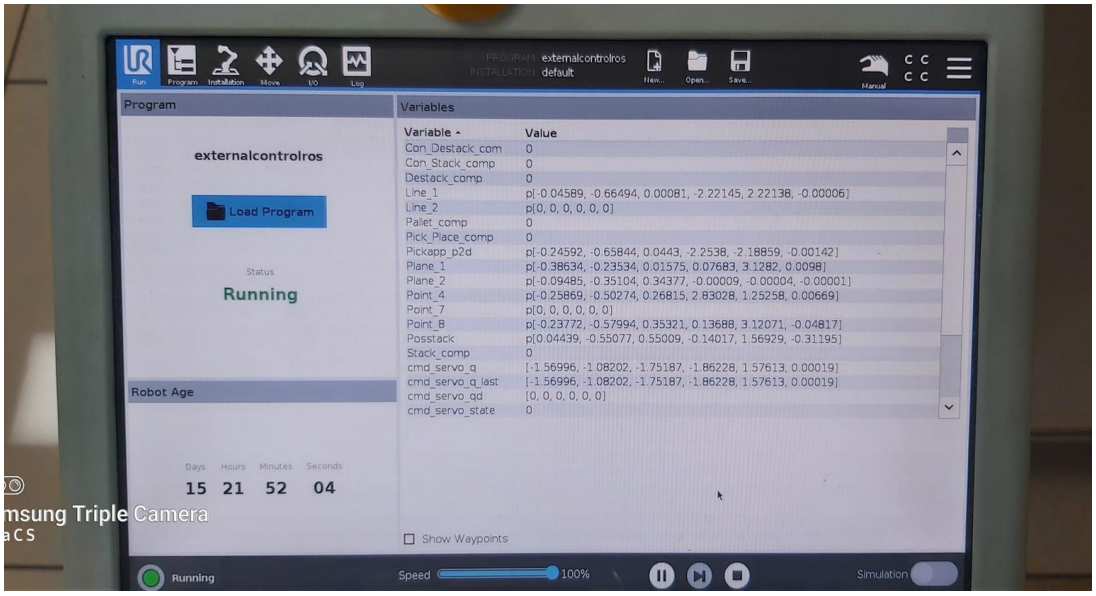


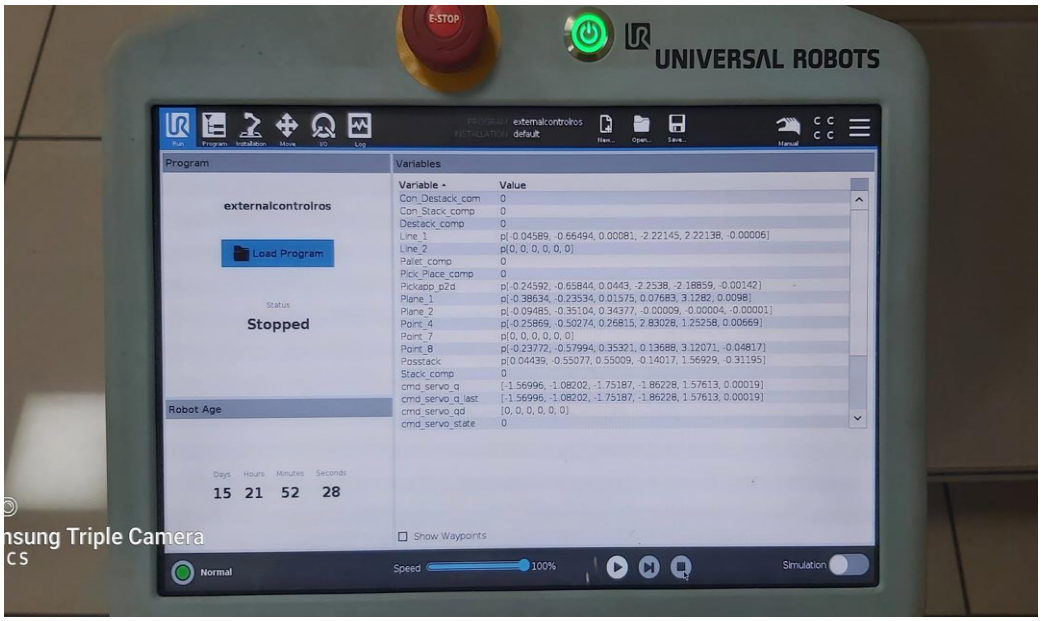
Settings in Teach Pendent:

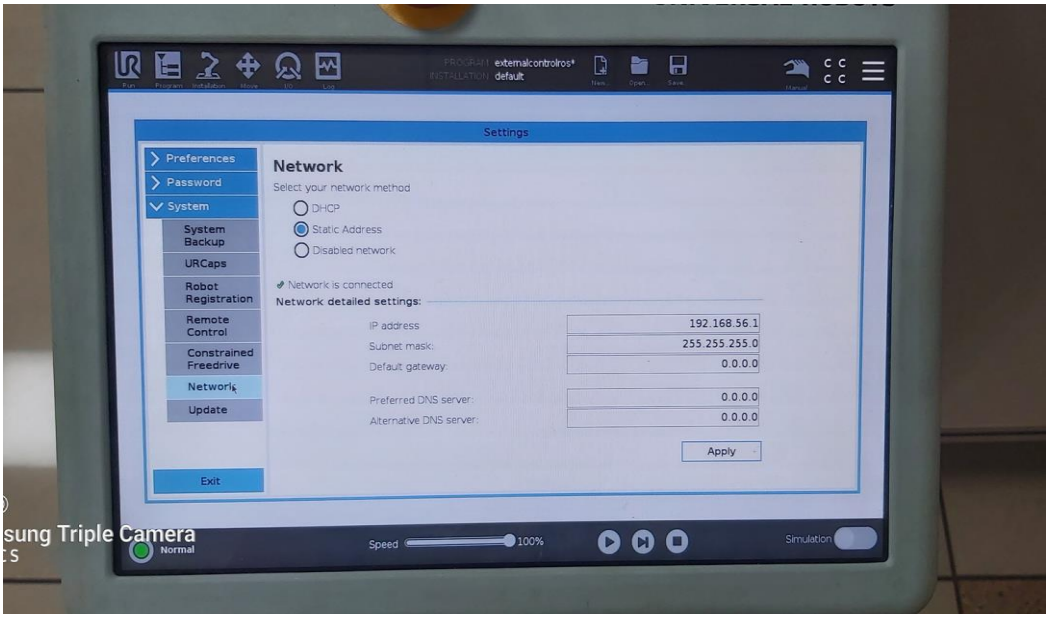


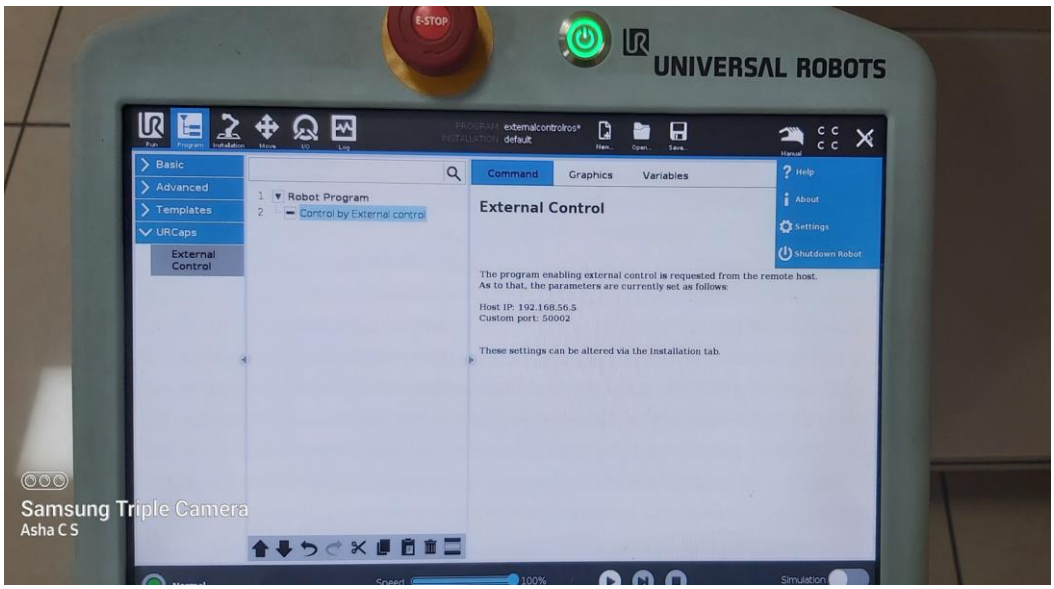

E-STOP
UNIVERSAL ROBOTS
Basic
Move
Waypoint
Direction
Wait
Set
Popup
Halt
Comment
Folder
Advanced
Templates
URCaps
Command
Graphics
Variables
1 Robot Program
2 Control by External control
External Control
The program enabling external control is requested from the remote host.
As to that, the parameters are currently set as follows:
Host IP: 192.168.56.5
Custom port: 50002
These settings can be altered via the Installation tab.
Normal
Speed
100%
Simulation


Settings
Preferences
Password
System
System Backup
URCaps
Robot Registration
Remote Control
Constrained Freedrive
Network
Update
Active URCaps
External Control (1.0.5)
Inactive URCaps
Remote TCP & Toolpath
External Control (1.0.4)
URCap Information
Exit
Restart
Normal
Speed
100%
Simulation

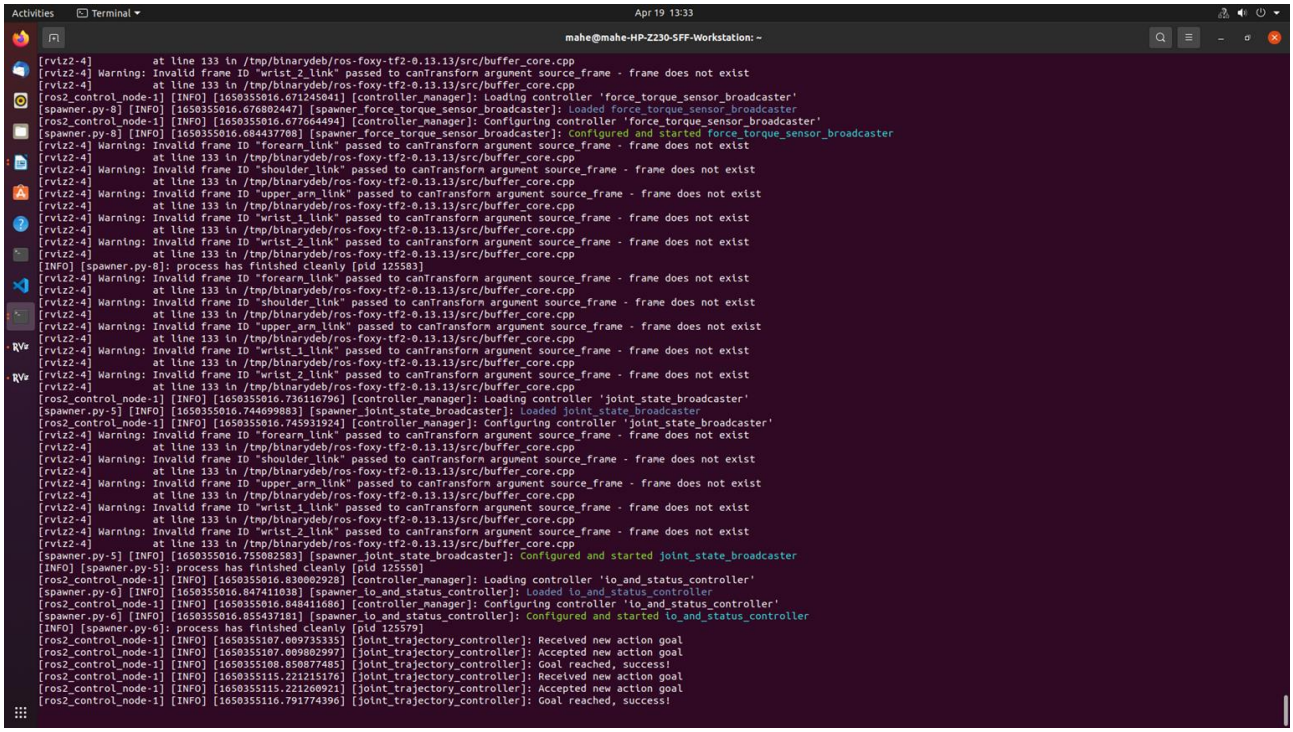

Run
Program
Installation
Move
I/O
Log
PROGRAM externalcontrolros
INSTALLATION default
New...
Open...
Save...
Manual
Program
externalcontrolros
Load Program
Status
Running
Robot Age
Days Hours Minutes Seconds
15 21 52 04
Variables
Variable
Value
Con_Destack_com 0
Con_Stack_comp 0
Destack_comp 0
Line_1 p[-0.04589, -0.66494, 0.00081, -2.22145, 2.22138, -0.00006]
Line_2 p[0, 0, 0, 0, 0, 0]
Pallet_comp 0
Pick_Place_comp 0
Pickapp_p2d p[-0.24592, -0.65844, 0.0443, -2.2538, -2.18859, -0.00142]
Plane_1 p[-0.38634, -0.23534, 0.01575, 0.07683, 3.1282, 0.0098]
Plane_2 p[-0.09485, -0.35104, 0.34377, -0.00009, -0.00004, -0.00001]
Point_4 p[-0.25869, -0.50274, 0.26815, 2.83028, 1.25258, 0.00669]
Point_7 p[0, 0, 0, 0, 0, 0]
Point_8 p[-0.23772, -0.57994, 0.35321, 0.13688, 3.12071, -0.04817]
Posstack p[0.04439, -0.55077, 0.55009, -0.14017, 1.56929, -0.31195]
Stack_comp 0
cmd_servo_q [-1.56996, -1.08202, -1.75187, -1.86228, 1.57613, 0.00019]
cmd_servo_q_last [-1.56996, -1.08202, -1.75187, -1.86228, 1.57613, 0.00019]
cmd_servo_qd [0, 0, 0, 0, 0, 0]
cmd_servo_state 0
Show Waypoints
Running
Speed 100%
Simulation

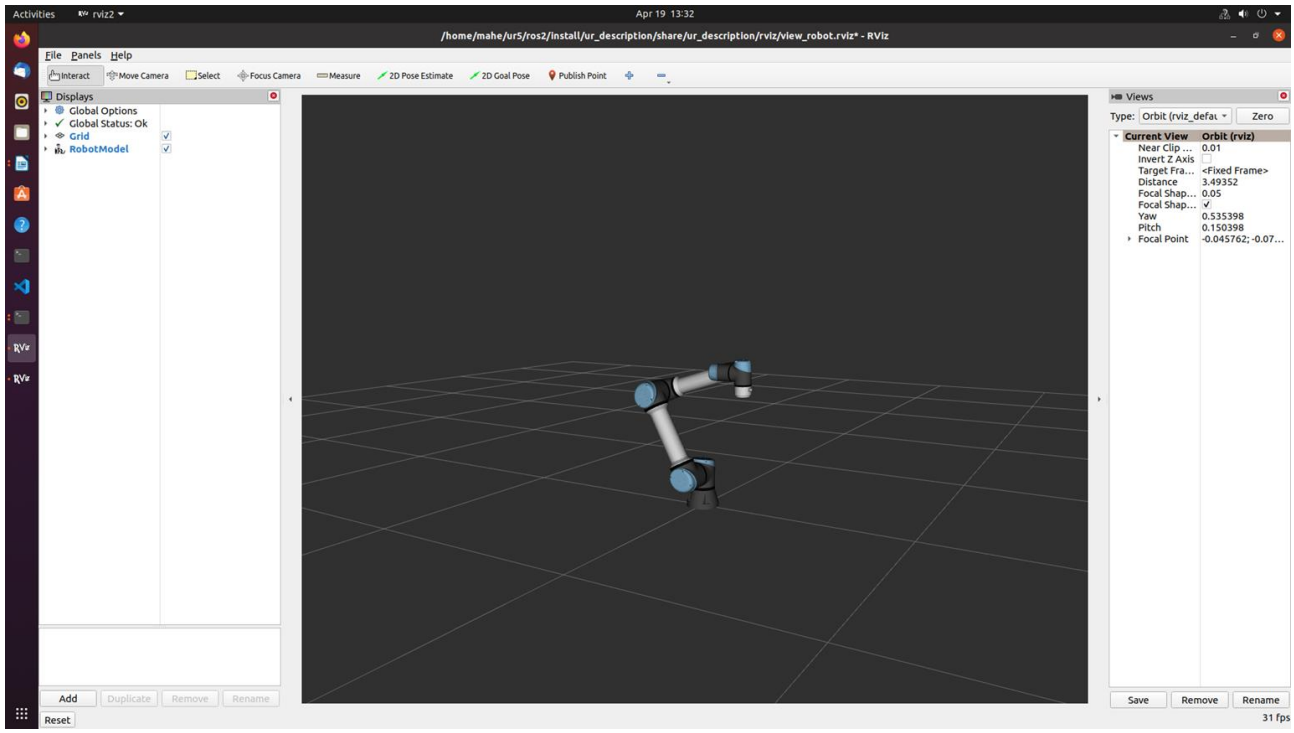

E-STOP
UNIVERSAL ROBOTS
Run
Program
Installation
Move
I/O
Log
PROGRAM externalcontrolros
INSTALLATION default
New...
Open...
Save...
Manual
Program
externalcontrolros
Load Program
Status
Stopped
Robot Age
Days Hours Minutes Seconds
15 21 52 28
Variables
Variable
Value
Con_Destack_com 0
Con_Stack_comp 0
Destack_comp 0
Line_1 p[-0.04589, -0.66494, 0.00081, -2.22145, 2.22138, -0.00006]
Line_2 p[0, 0, 0, 0, 0, 0]
Pallet_comp 0
Pick_Place_comp 0
Pickapp_p2d p[-0.24592, -0.65844, 0.0443, -2.2538, -2.18859, -0.00142]
Plane_1 p[-0.38634, -0.23534, 0.01575, 0.07683, 3.1282, 0.0098]
Plane_2 p[-0.09485, -0.35104, 0.34377, -0.00009, -0.00004, -0.00001]
Point_4 p[-0.25869, -0.50274, 0.26815, 2.83028, 1.25258, 0.00669]
Point_7 p[0, 0, 0, 0, 0, 0]
Point_8 p[-0.23772, -0.57994, 0.35321, 0.13688, 3.12071, -0.04817]
Posstack p[0.04439, -0.55077, 0.55009, -0.14017, 1.56929, -0.31195]
Stack_comp 0
cmd_servo_q [-1.56996, -1.08202, -1.75187, -1.86228, 1.57613, 0.00019]
cmd_servo_q_last [-1.56996, -1.08202, -1.75187, -1.86228, 1.57613, 0.00019]
cmd_servo_qd [0, 0, 0, 0, 0, 0]
cmd_servo_state 0
Show Waypoints
Normal
Speed 100%
Simulation


PROGRAM externalcontrolros*
INSTALLATION default
Settings
Preferences
Password
System
System Backup
URCaps
Robot Registration
Remote Control
Constrained Freedrive
Network
Update
Exit
Network
Select your network method
DHCP
Static Address
Disabled network
Network is connected
Network detailed settings:
IP address: 192.168.56.1
Subnet mask: 255.255.255.0
Default gateway: 0.0.0.0
Preferred DNS server: 0.0.0.0
Alternative DNS server: 0.0.0.0
Apply
Normal
Speed 100%
Simulation



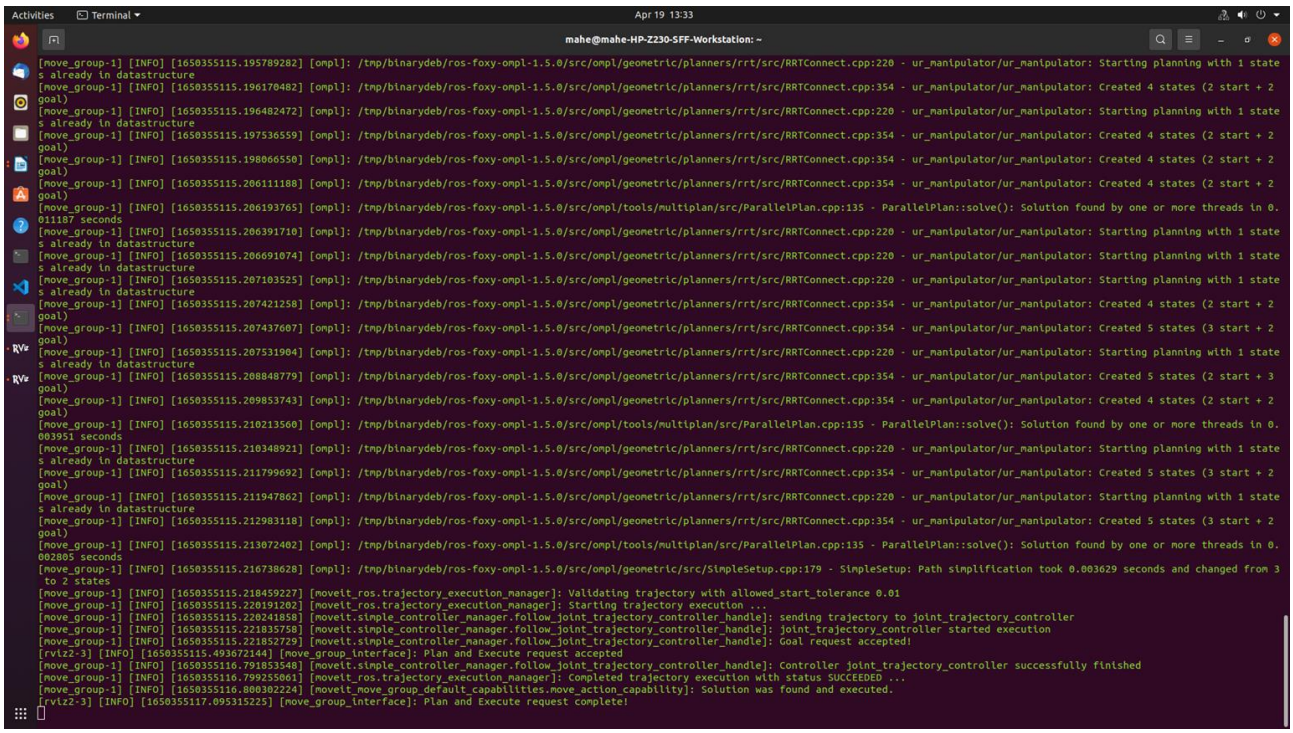
ros2 launch ur_bringup ur5e.launch.py robot_ip:=192.168.56.1

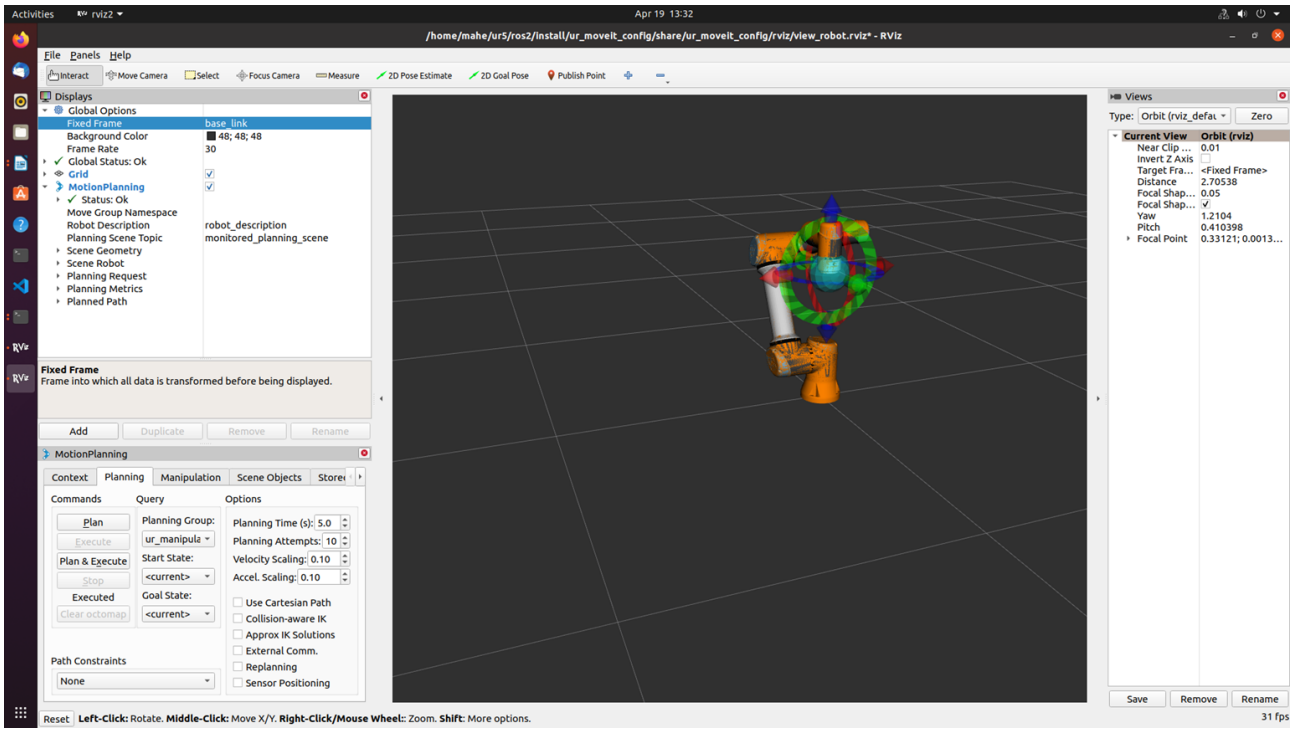




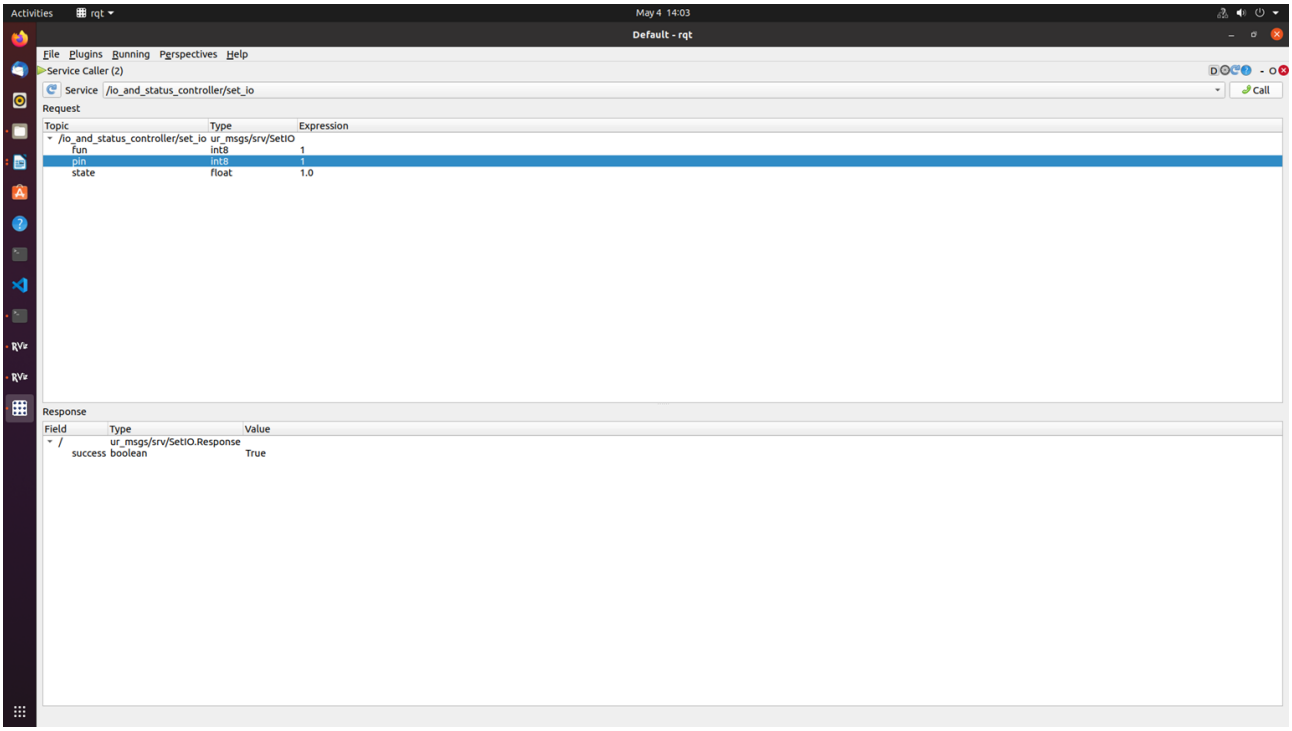
Set in manual mode (1234)
Open externalcontrolros program
Run the program

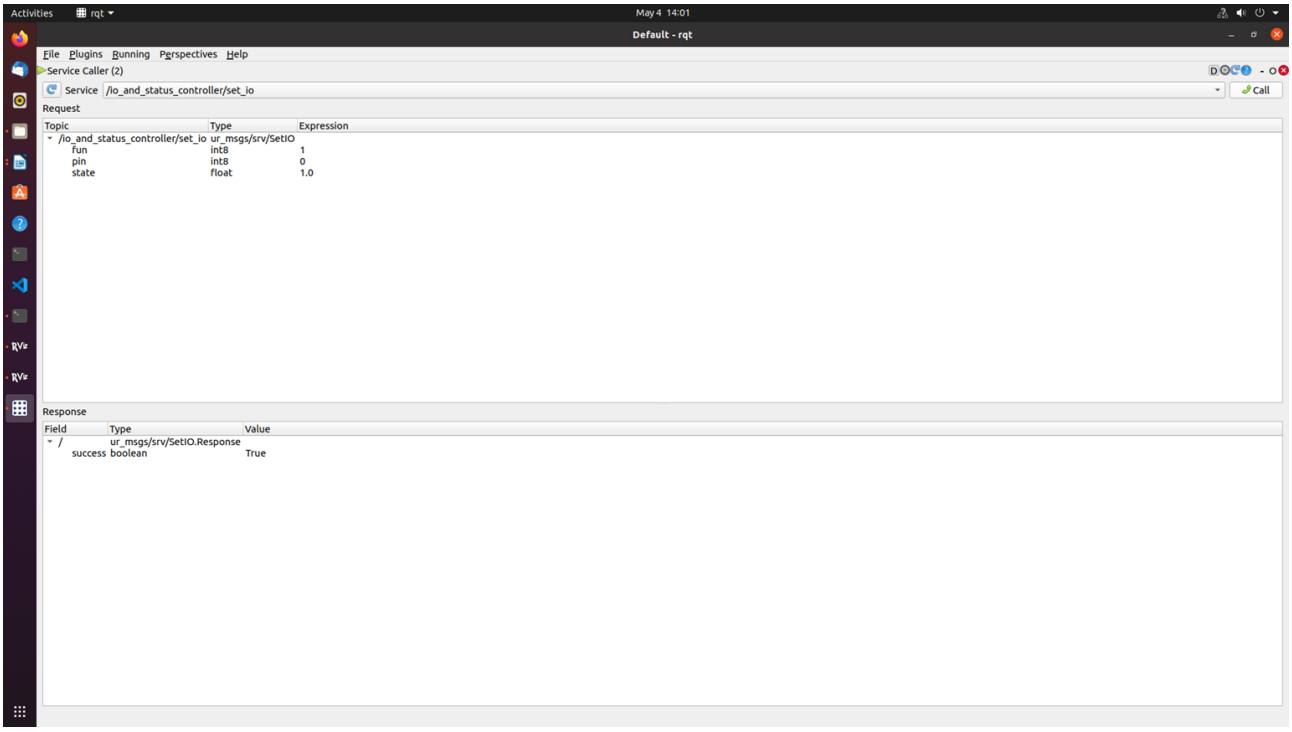
ros2 launch ur_bringup ur_moveit.launch.py ur_type:=ur5e robot_ip:=192.168.56.1

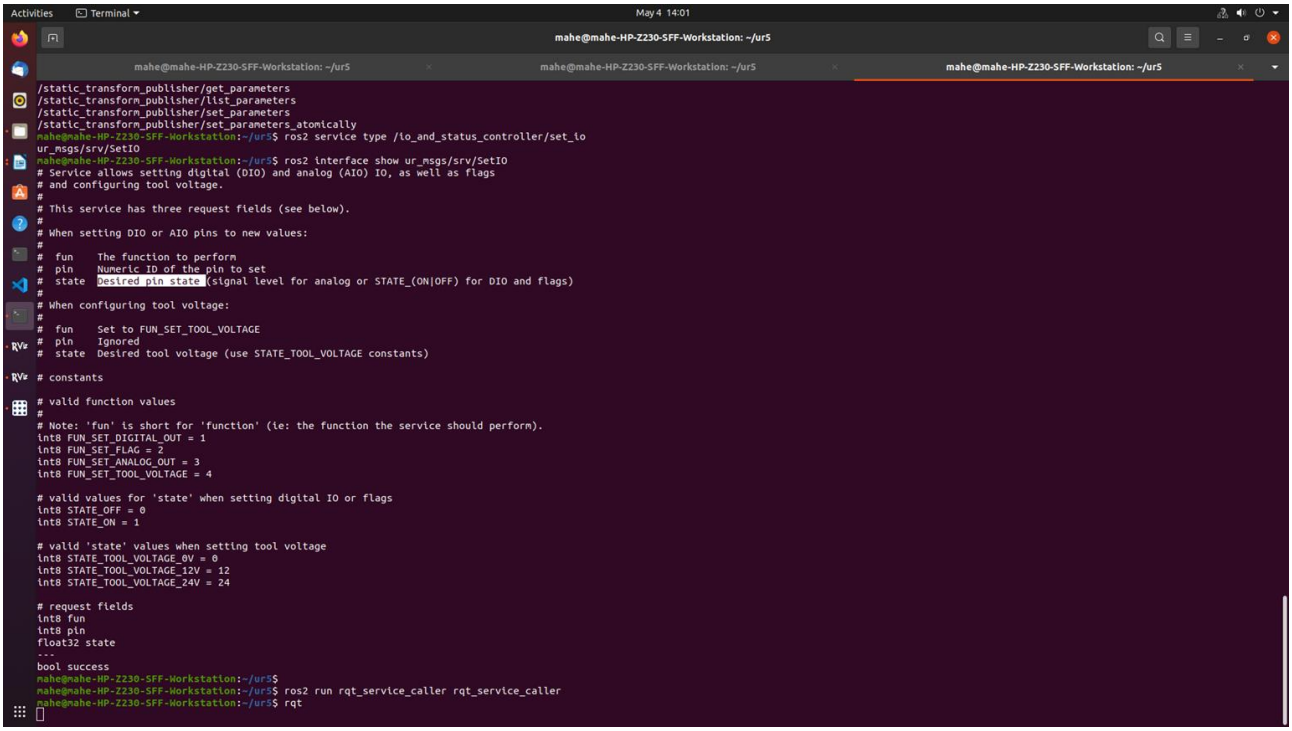




type >>
rqt
plugins→service caller
select io_and_status_controller/set_io
set the pin as 0 for gripper
state as 1.0 to set the gripper off
state as 0.0 to set the gripper off


Activities
rqt
May 4 14:03
Default - rqt
File Plugins Running Perspectives Help
Service Caller (2)
Service /io_and_status_controller/set_io
Call
Request
Topic Type Expression
/io_and_status_controller/set_io ur_msgs/srv/SetIO
fun int8 1
pin int8 1
state float 1.0
Response
Field Type Value
/ ur_msgs/srv/SetIO.Response
success boolean True


Activities
rqt
May 4 14:01
Default - rqt
File Plugins Running Perspectives Help
Service Caller (2)
Service /io_and_status_controller/set_io
Call
Request
Topic Type Expression
/io_and_status_controller/set_io ur_msgs/srv/SetIO
fun int8 1
pin int8 0
state float 1.0
Response
Field Type Value
/ ur_msgs/srv/SetIO.Response
success boolean True



Edit test_goal_publishers_config file as

```
publisher_scaled_joint_trajectory_controller:
  ros__parameters:

    controller_name: "scaled_joint_trajectory_controller"
    wait_sec_between_publish: 6

    goal_names: ["pos1", "pos2", "pos3", "pos4"]
    pos1: [-1.76, -2.129, -1.117, -1.483, 1.57, 0.1745]
    pos2: [-1.76, -2.3911, -1.448, -0.8726, 1.57, 0.17]
    pos3: [-1.76, -2.129, -1.117, -1.483, 1.57, 0.1745]
    pos4: [-1.76, -2.3911, -1.448, -0.8726, 1.57, 0.17]

    joints:
      - shoulder_pan_joint
      - shoulder_lift_joint
      - elbow_joint
      - wrist_1_joint
      - wrist_2_joint
      - wrist_3_joint
```

```
    check_starting_point: true
    starting_point_limits:
      shoulder_pan_joint: [-3.14,3.14]
      shoulder_lift_joint: [-3.14,3.14]
      elbow_joint: [-3.14,3.14]
      wrist_1_joint: [-3.14,3.14]
      wrist_2_joint: [-3.14,3.14]
      wrist_3_joint: [-3.14,3.14]

publisher_joint_trajectory_controller:
  ros__parameters:

    controller_name: "joint_trajectory_controller"
    wait_sec_between_publish: 6

    goal_names: ["pos1", "pos2", "pos3", "pos4"]
    pos1: [-1.76, -2.129, -1.117, -1.483, 1.57, 0.1745]
    pos2: [-1.76, -2.3911, -1.448, -0.8726, 1.57, 0.17]
    pos3: [-1.76, -2.129, -1.117, -1.483, 1.57, 0.1745]
    pos4: [-1.76, -2.3911, -1.448, -0.8726, 1.57, 0.17]

    joints:
      - shoulder_pan_joint
      - shoulder_lift_joint
      - elbow_joint
      - wrist_1_joint
      - wrist_2_joint
      - wrist_3_joint

    check_starting_point: true
    starting_point_limits:
      shoulder_pan_joint: [-3.14,3.14]
      shoulder_lift_joint: [-3.14,3.14]
      elbow_joint: [-3.14,3.14]
      wrist_1_joint: [-3.14,3.14]
      wrist_2_joint: [-3.14,3.14]
      wrist_3_joint: [-3.14,3.14]


cd ur5
colcon build –packages-select ur_bringup
ros2 launch ur_bringup test_joint_trajectory_controller.launch.py
```

Self-Study Material: Robotic Safety Industry Standards

Aim:

- To know safety, risk assessment, ethical, quality management, moral issues in the industry environment.
- To know various industry standards applicable to robotic engineer

Source: {https://blog.isa.org/new-industrial-standards-robotic-safety}

Books:

- Functional safety: safety instrumented systems for the process industry sector
- Safety Instrumented Systems: Design, Analysis, and Justification,

There have been significant changes in industrial safety standards for robotics. Safety managers, plant managers, and others need to keep pace with the latest codes and regulations. According to the U.S. Occupational Safety and Health Administration (OSHA), “machine guarding” that pertains to machines’ general requirements and general industry (29 CFR 1910.212) consistently falls in the top ten most frequently cited OSHA standards violated in any given year.

When combined with newly introduced safety regulations, it is easy to understand why this commonly misunderstood topic is more confusing than ever. The multitude of robotic applications and the growth of robot use and automation in all industries only worsens the problem.

Conducting a thorough risk assessment is the best way to maintain a safe work environment, especially when adding new automated processes. Thanks to the Robotic Industries Association (RIA) R15.06-2013 standard, proper risk assessments are no longer just a best practice; they are mandatory.

RIA 15.06-2013 harmonizes international and U.S. standards

This standard references ISO 10218-1 & 2, which addresses robots, robot systems, and integration. The RIA 15.06-2013 was written to be compliant with international standards already in place in Europe, making life easier for manufacturers and end users. This standard requires better hazard identification related not only to robotic motion, but also to the task being performed. Additionally,

it requires validation and verification of the safety systems employed and requires designs that incorporate protective measures for the robot cell and the operator.

Some of the biggest changes in the RIA 15.06 industrial robot standard have to do with safety- rated motion and allowing advanced programmable safety devices to be used. What this means is software will now be allowed “safety-rated” control of various aspects of the robot’s function, limiting the area in which the robot operates and the speed of robot motion. This is a departure from older standards in which programmable safety controls were not allowed. In addition, as part of this standard, risk assessments are now required. Many professionals responsible for plant safety have been conducting risk assessments to increase safety as a matter of practice. These regulations mandate risk assessments be conducted.

The basics of machine guarding risk assessment

Understanding and assessing these risks—and ensuring compliance—is not a simple task. The first step for facility/safety professionals is to identify and understand all applicable codes and regulations for their facility and operation. Next, they should examine the prevailing machine guarding choices for those applications to validate their safety system and its components. Although many guarding methods and products are available, not all can be applied universally. Every machine guarding application has a set of unique challenges and associated risk.

The choices a facility manager makes for one application might not be the same, or appropriate, for the next. In most cases, safety-conscious managers would not guard an industrial robot the same way they would guard other equipment, because the risk associated with each differs greatly. Risk may even vary between similar operations, depending upon employee exposure and other factors.

Determining risk

When performing a proper risk assessment, point-of-operation guarding is the most involved aspect. It is easy to place perimeter guarding around the entire process. However, in most situations a machine operator needs to interact with the process by loading or unloading materials (such as metals to be welded) and “running” the machine. This point-of-operation is where things get tricky. Many details must be considered when it comes to this area, including the layout or design of the process and the limits of the system.

Also, facilities must properly identify all associated hazards and devise methods for hazard elimination and risk reduction. Once the severity of the potential hazard has been determined, the

frequency or duration of exposure and the possibility of eliminating or limiting exposure can help safety managers choose the proper machine guarding device. Also, using the distance formula identified in OSHA guidelines can help in this selection. Per this formula, the safeguarding device has a prescribed location based on a number of factors, including secondary hazards that might harm a machine operator.

Limiting hazard exposure

Light curtains, laser scanners, and other presence-sensing devices are a commonly used and widely accepted method of machine guarding in manufacturing facilities from Tier 1 automotive to small machine shops and fabrication facilities. With presence-sensing, the automated process ceases once the safety device's infrared beam is tripped. In many instances these devices provide acceptable safety.

However, they are not always the best choice in all applications, especially after a risk assessment is performed. Curtains may be the right choice in some applications. However, fast-acting automated barrier doors or roll-up curtains may be better choices because they can eliminate exposure to both the dangerous movement of the machine and the secondary hazards produced by the process, such as smoke, flash, splash, mist, and flying debris. This further diminishes the potential risk and the severity of exposure.

Coupled with safety interlocks (up to PLe per EN ISO 13849-1 when integrated properly), automated barrier doors and roll-up curtains offer an increased level of protection for point-of- operation guarding. They restrict access to the process and contain secondary hazards of automated welding operations by placing a barrier between machine operators and machine movement. These types of guards are an ideal alternative to light curtains and other presence-sensing devices in many situations.

A fast-acting automated barrier door or roll-up curtain eliminates exposure to dangerous movement machines and hazards produced by the process, such as smoke, flash, splash, mist, and flying debris.

From EN 954-1 to ISO 13849-1 and EN 62061

One of the biggest regulatory paradigm shifts occurred with the move from EN 954-1 to ISO 13849-1 and EN 62061. Although approval of this harmonized standard was a hotly contested fight, it is now here to stay. Fortunately for those in charge of safety, best practices and market- ready solutions already exist. ISO 13849-1, when broken down to the basics, provides a clearly defined set of rules to follow when designing the safety system as applied to industrial machine control systems. Officially defined as “safety of machinery, safety-related parts of control systems, general principles for design,” this regulatory shift was made necessary by increasingly complex manufacturing processes using robotic and automated technology.

Safety control systems and methodologies were forced to keep pace. The ISO 13849-1 standard is more quantitative than EN 954-1. It applies common sense and forces facility managers to validate their safety systems, whereas EN 954-1 was conceptual and only required facilities to apply safety devices (controls) properly, specifying non-programmable, out-of-date technology. Let’s face it, our increasingly complex manufacturing processes require more complex systems to monitor their safe operation and keep machine operators safe.

Automated processes, robotics, and even time-tested processes all require considerable attention to ensure those processes can proceed both efficiently and safely. EN ISO 13849-1 is ultimately making a much safer manufacturing environment, because it accounts for the regulatory gaps in the older standards. Because every robotic system is different and has its own set of guidelines, it is important to realize what they are before implementation.

Know specifications such as space and cycles. Integrators of new robotic systems will be required to perform these risk assessments in an attempt to identify potential dangers and ways to limit and eliminate them. RIA 15.06 is similar to ISO 13849-1 in that it has a quantitative approach to hazard identification. A functional safety requirement of D (performance level “d”) will be required of all robotic systems, as well as structure category 3 (dual channel), unless a risk assessment determines otherwise. PL safety and category ratings will offer a much more measurably reliant way to gauge safety.

The new standards in safety

For regulations such as RIA 15.06 and EN ISO 13849-1, it is important to keep up with the latest and greatest safety technologies available to match the right product to the right process. Consider not only potential machine hazards, but also the task being performed. Advances in design and

available technology make automated barrier doors ideal for guarding the machine and protecting operators, ultimately increasing productivity and the level of safety for years to come.

Creating a smaller manufacturing cell

Due to the nature of a properly interlocked automated barrier door, certain aspects of OSHA's safety distance formula become moot, because there is no depth penetration factor. The safeguard can be placed much closer to the hazardous area, and there is less space dedicated to a "safety zone," which reduces the manufacturing cell's footprint. This space savings is a huge benefit in most facilities.

The smaller safety zone may also help to increase productivity and create a better ergonomic situation for the machine operator by limiting required motion. Eliminating accidental entry into the cell is another benefit of interlocked automated barrier doors. Because their safeguarding can be seen (unlike the invisible infrared beams of presence-sensing devices), they greatly reduce the opportunity for accidental work stoppage. The physical separation is a clear visual indicator that the machine operator needs to be on task.

Do not fall out of compliance

Regardless of the safety device selected for machine guarding, facility managers must remember to perform a proper risk assessment. Although it can be tricky, the process will ultimately make a facility safer for workers and keep it in compliance with RIA R15.06.

Source:

{https://blog.isa.org/are-autonomous-vehicles-the-answer-to-road-safety}

How Autonomous Vehicles Work

Of course, it all starts with understanding how autonomous vehicles work. The industry itself is extremely fast-paced, developing all the time on a number of different fronts. According to a report on connected vehicles by Verizon Connect, the technology has three distinct forms: V2V, V2I, and V2X. The differences between these types are as follows:

- V2V stands for vehicle-to-vehicle, and refers to short-range communication technology that will allow cars to communicate with one another. This tech is designed to reduce road incidents by sharing information such as the speed of the surrounding vehicles.

- V2I stands for vehicle-to-infrastructure, where the infrastructure in question refers to overhead RFID readers, streetlights, traffic lights, and signage. These data collection points will gather detail on traffic conditions, weather, and potential roadblocks in order to transmit these to vehicles on the road.
- V2X stands for vehicle-to-everything, and essentially encompasses V2V and V2I systems. According to the Verizon report, creating V2X systems can also help facilitate toll payments. With autonomous vehicle technology paving the way in the automotive industry, it may only be a matter of time before this tech is branching into motorcycles and bikes.

The Public Safety Benefits of Autonomous Vehicles

One day, autonomous vehicles may decrease instances of human error on the road, which still accounts for the majority of car accidents. While autonomous vehicles still don't excuse reckless driving behaviors such as driving while intoxicated or speeding, they can help the everyday driver make smarter decisions when on the road.

By relying on V2I-connected infrastructure to monitor traffic conditions, local government agencies can also learn whether existing traffic procedures are working. Autonomous vehicles will likely play an important role in the smart city phenomenon. Indeed, there's a huge ripple effect that autonomous vehicles can have on our cities. Understanding traffic congestion through smart technology can inform decisions like how bus routes are formed and where pedestrian lanes are put.

Autonomous vehicles may also have a big impact on sustainability efforts in the future. Manufacturers like Ford and Volkswagen are racing to get more electric cars with self-driving features on the road. A recent study in Nature Energy investigated the trade-offs in weight, computing load, and sensor load between autonomous vehicle functionality and electrification. "We're getting to a point where we won't need to choose between autonomous driving and electric cars," Venkat Viswanathan, an author of the study, told Bloomberg.

The same study also found that autonomous vehicles can see an energy savings of up to 10\%. By driving smarter and reducing traffic congestion, autonomous vehicles can end up reducing a car's power usage as well as its overall fuel consumption.

Autonomous Vehicles as Part of a System-Wide Solution

Innovations in autonomous vehicles aren't just a concern for private companies. Researchers from the University of Texas at Austin are working with their city government to see how smart

technology can be implemented in and around Austin. Furthermore, Vox's article on the future of self-driving cars makes the argument that regulations and standards will have to be set to ensure that autonomous cars are safe enough for the road.

With these coming safety regulations in mind, the design and manufacture of autonomous vehicles must take into account the diverse environments in which they'll operate. They must be safe in all conditions.

In all honesty, we're likely years away from autonomous vehicle technology becoming commonplace and being fully incorporated into smart cities. It will be even longer before fully autonomous, completely driverless vehicles take over the roads. That said, the groundwork that's already being done promises great advancements for road safety and overall smarter driving.

Source: {https://blog.isa.org/functional-safety-culture-what-is-a-business-leaders-role}

You're on the business side of the enterprise, not on the engineering side—how could functional safety possibly be your responsibility?

"Safety is everyone's responsibility," you likely just thought to yourself, dutifully. While that's certainly true, do you really have a deep understanding of your specific role and responsibility? Do you have defined goals, KPIs, and timelines? Does your boss understand your contribution and support you with time and budget?

If any of those questions make you a little nervous, you're in the right place. (If none of those questions make you nervous, you're still in the right place—we'd love to bring you into the conversation and learn more about how you've tackled these challenges in your facility.) Consider this: the head of Transocean testified that while he wished his crew had done more to prevent the infamous Deepwater Horizon disaster, the company's investigation found no failure of management.

How is that possible? The only explanation, outside of dishonesty, is the disconnect between a company's business leaders and its real, on-the-ground, actual safety posture. "Our facilities have been accident-free for 495 days," you just thought, rather smugly. But what constitutes an accident in your internal reporting practices? What incentives will managers lose for reporting minor or repeated problems with equipment or people? Are audits being conducted,

or are you relying on proactive incident reporting to uncover issues? Do you monitor leading indicators regularly?

Now that you're sufficiently concerned, it's time to explore the role and responsibility that you have for creating and sustaining a safety culture as a business leader in a process facility.

Quick Tips

Creating a Safety Culture: A Leader's Role

According to an article written by Scott Stricoff, president of the consulting firm BST, there are four organizational elements critical to a safety culture:

- Anticipation: recognizing and acting on the weak signals that indicate potential for events
- Inquiry: ensuring the right questions are asked and the right analyses are done
- Execution: using systems consistently and reliably
- Resilience: enabling workers to have the knowledge and willingness to intervene on small issues and prevent them from becoming big issues

As a business leader in a process facility, your role includes the following: Understand

Develop a working knowledge of the IEC/ISA 61511 functional safety standards, which are considered "good engineering practice" by OSHA, have been adopted as national standards by every country in the European Union, and are now referenced in the Canadian Electrical Code Determine the levels of expertise needed at each level of personnel, and

At regular intervals, take time to sit down with peers and subordinates and listen to their challenges Communicate

Develop, and articulate, a comprehensive strategy for achieving functional safety

Include safety strategy as a regular topic on the agenda for meetings with your bosses and with your subordinates

Delegate

Assign responsibility for safety-related tasks and decisions; set clear objectives and measures; monitor process and progress

Oversee contractor practices; make sure your managers have documented requirements and processes for contracted labor resources, especially related to safety-critical functions

Anticipate

Focus on leading indicators and proactive asset management

Remember that leading indicators aren't just physical or technical in nature; often, complacency of employees or contractors is your biggest leading indicator for safety problems

Document

Encourage, and even reward, accurate reporting and documentation of incidents, including small ones that might hint at larger issues

Review documentation thoroughly and make sure management is following up on recommendations and resolutions

Remember that documentation isn't an indictment of your facility's failures—it's evidence of committed and capable employees, strong management, and well-defined procedures to resolve small issues before they become big ones. Documentation is the only way we can turn individual observations into actionable improvements, because otherwise, you'll never see the trends that are emerging in your operations

Prioritize

Decide that the long game is more important, or at least equally important. Process safety is a marathon, not a sprint. Budgets and profitability, when viewed in the short term, could be seen as barriers to a safety culture—but when you look at profitability as a long-term objective, you'll see that well trained, careful, measure-twice-cut-once employees will be safer AND more productive As renowned inventor Dean Kamen likes to say, "we get what we celebrate." What are you incentivizing through your company's bonus, performance evaluation, or promotion practices? You might be saying, "safety is important" with your words, but are you saying it where it counts? Or, are you inadvertently disincentivizing reporting of smaller incidents, prioritizing uptime over important proactive maintenance, and without meaning to, rewarding a "sweep it under the rug" mentality?

Source: {https://blog.isa.org/the-role-of-standards-in-functional-safety}

Internationally recognized functional safety standards have been developed and adopted to increase equipment and process safety. The primary goal of these standards is to develop a continuous improvement approach to safety system management and enable end users to understand the safety status of their assets.

IEC 61508 and IEC/ISA 61511

The International Electrotechnical Commission (IEC) published IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems, as a general standard applicable to many different industries. IEC 61508 provides the core requirements for safe system design of hardware and software, and it is the framework for sector-specific standards, such as IEC 61511 (process industries), IEC 61513 (nuclear applications), and IEC 62061 (discrete manufacturing and machineries).

ANSI/ISA-84.00.01-2004, Functional Safety: Safety Instrumented Systems for the Process Industry Sector, was first issued in 1996. The series of standards have been harmonized with IEC 61511.

Regulatory Requirements

In 2000, the U.S. regulatory body OSHA issued a letter identifying the ISA 84 standard as “good engineering practice” for safety instrumented system design. Reaffirmed by OSHA in 2005, the guidance effectively makes the ISA 84 standard part of process safety management (PSM) requirements. Paragraph (d)(3)(ii) of the OSHA PSM standard specifies: "The employer shall document that equipment complies with recognized and generally accepted good engineering practices.”

The European standards body, CENELEC, has adopted the standard as EN 61511. Each member state in the European Union has subsequently published the standard as a national standard. As recently as a few months ago, IEC 61511 was adopted into the Canadian Electrical Code as CSA- C22.2 NO.61511:17.

Widespread adoption, however, doesn’t guarantee a safer environment. Compliance with the standard requires a focused and continuous approach to functional safety.

Source: {https://blog.isa.org/what-is-functional-safety}

Functional safety focuses on the detection of a potentially dangerous condition and depends on automatic protection or correction to prevent an unwanted consequence or reduce its severity. The automatic protection system is designed to respond appropriately to errors, hardware failures, and operational stressors.

When every specified safety function is carried out and meets the set level of performance, functional safety is achieved. This requires a process that includes:

Identifying the required safety functions, usually through HAZIDs, HAZOPs, reviews of accidents, and process reviews

Assessing the risk reduction required; setting a safety integrity level (SIL), which applies to the safety instrumented function (SIF) intended to prevent the hazardous event

Ensuring that the safety function performs under various conditions, including failure modes and operator error; personnel must be qualified and competent to test these functions against IEC/ISA 61511

Verifying that the system and software meets the assigned SIL by determining the probability of failure, checking minimum levels of redundancy, and reviewing systematic capability; these three metrics are often called “the three barriers”

Conducting regular safety audits to make sure that the appropriate safety lifecycle management techniques have been applied throughout the life of a product or system

The functional safety lifecycle provides an end-to-end approach, beginning at the concept design phase and ending at decommissioning. It’s a closed loop model that identifies and assesses risks, creates a design, and then implements, verifies, and maintains that design.

Source: {https://blog.isa.org/how-to-achieve-career-success-in-process-automation}

Leadership, the ability to motivate it, and the path to results it provides is important in any human venture, especially those requiring the broad diversity of skills that many of our projects do. With that in mind, it is worth reflecting for a moment on some key concepts about the you in your work. Know yourself, your preferences, strengths, weaknesses, desires, interests, and ambitions that are part of who you are. No matter how great you might presently be, you can be better. Seek self- improvement. Strive to expand and improve your skills, attitude, and dedication—be all you can be. When I was in high school, the Harlem Globetrotters played at our school. In a discussion with students, one of the players at the time talked about how to become great. He said that when he was in high school, his coach asked him what basketball skills he was going to improve over the summer. He was shocked by the comment, and responded that he led the league in scoring now, and played the game as well as anyone he had played with or against. The coach said, "Yes, but you could be better." That triggered a thought process for the high schooler—there was a long shot

in that championship game, there was the one-step defensive move that stopped a drive, and so on. He walked out on the court and took a long shot and watched the ball arc toward the basket and thought, "Yes! I could do this better—faster and from farther away." It became part of his life to think about not only success, but also improvement. Be all you can be, but keep trying to be better! Care about your people, and care about your mission. Put you into it. The mission is your assignment and purpose, and it cannot happen without your people—your team. Keep that firmly in mind. Communication with your team is an important means of conveying information and paying attention to their needs, concerns, and expectations. It also helps the team focus.

n that championship game, there was the one-step defensive move that stopped a drive, and so on. He walked out on the court and took a long shot and watched the ball arc toward the basket and thought, "Yes! I could do this better—faster and from farther away." It became part of his life to think about not only success, but also improvement. Be all you can be, but keep trying to be better! Care about your people, and care about your mission. Put you into it. The mission is your assignment and purpose, and it cannot happen without your people—your team. Keep that firmly in mind. Communication with your team is an important means of conveying information and paying attention to their needs, concerns, and expectations. It also helps the team focus.

Stand for your values and duties. Your team will respect consistent and logical application of principles. They will admire a leader who commits to and spares no effort in doing what is ethical, proper, and required.

Set the example in all things. People love to see success, and tend to shun and reject failure or things that look like it. Exemplify what you want your team to be, and how you want them to act. A leader is endowed with privilege and authority, but that only works when you do the right things the right way. Become the best example for everything you advocate.

Finally, one important concept seems to distract many upcoming people. There is a pertinent and important quote that suggests, "If you can dream it, you can make it so." That is absolutely a true statement. The modern perversion of it, though, removes the you and substitutes some amorphous god-like societal purpose, claiming this force will somehow make your life what you dream it will be. Dreaming will not make it so. You will.

Once upon a time, I was reviewing the program for training new soldiers associated with a force expansion. The senior soldier leader involved in the project observed that success was possible— hundreds of soldiers had proven that—but achieving success required desire. He went on to point out we could teach them, develop them, provide opportunities for experience, and fine-tune results, but individual success depended on each of them. They had to want it.

Dreams help create vision, and vision helps translate desire into reality. This translation into reality involves the focused manifestation of your available and necessary desire, ideas, and other fundamentals. In my experience, there is no shortcut for that path. The psychiatrist Carl Jung observed, "Who looks outside, dreams; who looks inside, awakes." The you on the path to making a dream "so" is a very important part.

Mini Project (20 Marks)

Mini Project Guidelines

Group Report: Date 18-08-2022

Submission Date	30-11-2021
Marks Weighting	20 Marks
Submission Method	Report Submission through MS team Assignment Presentation and contribution by team members Synopsis– 25-08-2022 Component Selection- 30-08-2022 Project Plan (Use Excel Chart) – Implementation Plan Report Writing, Presentation- 25-11-2022 The topic should be relevant to your core course only. The topics lists are shown in https://github.com/ashacs2. A detailed report should be in submitted in the Latex Template/- Word Template. It should be covered within 40 pages, including references. The sample conference format is to be prepared as per IEEE guidelines. The number of pages is limited to 6. (This is applicable if the work is novel in terms of design or application, Repetition of the existing work is encouraged for publishing) https://www.ieee.org/ conferences/publishing/templates.html **Overleaf template for the report writing** https://www.overleaf.com/read/spzdpgmdktxb Upload the reference papers, videos, report, ppt, and workspace as a zip folder in the assignment section.
Max members	4

Group Work Objectives	Explain the learning benefits of working in diverse groups. Expect all students to perform all roles and show excellency in their strengths (writing, presentation, planning, implementation, etc.). Encourage students to set ground rules such as -What is acceptable -Separate role for everyone -Clarity about task and individual contributions -Remind groups that practical arrangements should work for all. -Everyone will be expected to participate. -Improve presentation skills before the final presentation. -Engage in tasks considering religious commitments, gender issues, etc. Design Task is better done in groups than individually. involves all group members. makes use of participants' skills and experiences. makes diverse skills and knowledge an asset.
Group Form Criteria	Achieving Diverse Groups: allocation based on technical diversity, regional and gender diversity, language diversity Start with writing on a synopsis to encourage the group to start working together. Try to break initial tasks into specific briefs for each student so everyone can contribute. (Mention the task assigned in the Gantt chart)
Group Work Assessment	Marking criteria considers the reward of working effectively in a diverse group. Encourages students to show presentation skills in the assessment. The option is to learn in groups, but the assessment is done individually. For disabled students: Extra written work instead of an oral Presentation if this is difficult due to a disability. Will recognise different contributions of group members such as planning, coordinating, research, IT, writing, and presentation skills.
Mandatory Coursera Course	**Collaborative Robot Safety: Design & Deployment – Coursera Robotic System, ROS & Applications in Public Safety – Udemy** Complete the course before Oct 30th and submit the certificate to your mentor and course instructor One-page report on what you learnt and how this course benefitted your learning

Mini Project Learning Outcomes

- Identify key research areas in automation, aerial robotics, swarm robots, manipulators.
- Select the requirements of resources based on an application task; formulate the objec- tives.
- Implement design solutions to real-world problems using modern hardware and software tools.
- Manage the work plan with team members to effectively.
- Present the design solution through of report and oral presentation.

Requirements

You are required

- to design, implement, and evaluate algorithms for your application.
- to use the template as mentioned for report writing
- to propose, implement, and perform the detailed result analysis.
- Each team member to make an equal contribution to this assignment, although it is up to you how you would like to divide the project into work packages and allocate resources

When writing, plagiarism will be checked for the report and codes used.

Group Report Template with Marking Criteria

No	Name of group members	Contribution to Project	Contribution to report
1			
2			
3			
4			

Report Guidelines:

1. **Introduction** (5%)

Introduce the background of the project. Illustrate any assumptions made. Clearly

show the aim and objectives of this project and discuss the challenges.

2. **Related works** (10%)

Conduct a short literature review on methods relevant to your area. For example, if algorithms proposed in prior works on similar areas seem to be applicable to your area, you may also include a critical review of these. In later sections, you may use this literature review to assist with the justification of your methodology and discuss its capabilities and limitations. You will be assessed on the breadth and depth of the review. You are suggested to consider recent five years of research papers in the automation and robotics.

3. **Methodology** (30%)

Show technical breadth and depth. Justify the use of specific methods. Use flowcharts/block diagrams to illustrate the process. Youmust explain the design task and programming parameters in detail with proper justification.

4. Experiment and implementation (15%)

Demonstrate that you can implement the proposed approaches using ROS. Describe the software/hardware, core python packages, and how you selected parameters/optimisation of key algorithms if applicable. Use flowcharts, diagrams and pseudo code where applicable.

5. Results and Evaluation (15%)

Present results; evaluate the proposed approaches (quantitatively and qualitatively) using appropriate metrics and interpret findings. Make sure you explain how results were obtained and what they mean. Use graphical ad mathematical tools appropriately.

6. Conclusions and Future works (5%)

Conclude the project. Identify challenges relevant to your application that have not been fully resolved within this project's scope. Propose future works to deal with these challenges.

7. Application (Quality of the work) (10%)

The work is assessed based on the application, including health, safety, ethics, risk

assement in the robotics area.

8. Report Writing and Presentation (10%)

The remaining 10% of the mark is allocated to report presentation, including logical structure and clarity, quality of writing, spelling, grammar, diagrams, figures and tables, clarity of expression and use of English, and accuracy, consistency and completeness of citations and references.

The marking scheme is given as below. You should structure the paper as follows: a) Abstract.

b) Introduction and focus of the paper. c) Review of the literature related to the topic's key theme(s). d) Discussion of the fundamental issues relating to the topic. e) Conclusions, including recommendations for future research.

Overleaf Templates for viewing and downloading

https://www.overleaf.com/read/spzdpgmdktxb

The rubrics include the work, report and presentation of individual members of the group

	1-25	25-50	50-75	75-100
Abstract (5%)	The abstract does not accurately summarise the work	summarises some of the key aspects of the work, but some points unclear	Well written, but does not fully capture the scope	An excellent summary of the paper presented concisely and accurately
Introduction (5%)	Does not clearly set the background of the work	The scope of the paper is clear but does not clarify the focus	The scope and focus of the paper is well written but more consideration to rationale needed	An incredibly thoughtful and well-considered introduction which presents a clear rationale and focuses of the paper
Literature review (40%)	Little literature included	A limited background literature review or not fully understood	Some sound academic content, but not challenging	Exceptional breadth and depth of content understood and implemented
Discussion (25%)	Very rudimentary and superficial	Demonstrates application of theory through critical analysis but discussion lacking depth and scope	Clear application of theory through critical analysis for some of the discussion	Very analytical and clear discussion well-grounded in theory and literature showing synthesis of concepts and ideas
Conclusions (25%)	Very limited and superficial	Discussion of some issues, but overall, the conclusions lack insight	A broad range of issues discussed and follow on from the previous sections	An excellent, well considered conclusion which shows understanding and insight

Selection of topics for mini projects:

1. ROS2 and deep learning for autonomous driving.
2. ROS2 for manipulation. (pick and place).
3. ROS2 for Navigation (Autonomous Navigation).
4. Moveit2 for Cobot application.
5. Design of Robot in Solidworks and Programming.
6. Teleoperation of a robot in the real world using ROS2 (Webots).
7. Quadrouter programming using ROS2.
8. Blender-based world creation and use in simulation.
9. Chatbot design and programming.
10. ROS2 for Industrial Robots. (UR, KUKA, ...)

Component List Available for Mini Project

Lab In-charge (for components): Dr Pooja Nag/Mrs Vedavathi (Contact her for components and free lab slots)

Rules for components usage:

- You can borrow the components with prior permission from the lab in charge
- You need to enter the details of components in the log book.
- Use the components carefully and return to the in charge before leaving the lab.
- You are not allowed to take the components out of the lab.

List the components required for the mini project based on the list below.

Sl. No	Components available for mini project	Number	Specifications
1.	Small Metal Chassis	5	
2.	Motors (300 RPM)	6	
3.	3D printing material PLA/TPU	2 bundles	
4.	BLDC motor	10	
5.	ESC	10 sets	
6.	Propellers	5 sets	
7.	Arduino NANO	02	
8.	Arduino GEMMA	02	
9.	Raspberry pi 4	02	4GB RAM
10	Raspberry pi 3	03	3GB RAM
11	Arduino UNO	06	
12	Arduino YUN	02	
13	Arduino Mega	04	
14	Arduino Due	04	
15	Bread board power supply (YW Robot)	04	
16	Arduino Mini	05	
17	LED	RGBY	
18	Flow Sensor	04	
19	IR Sensor	10	
20	Ultrasonic Sensor	10	
21	Motor driver	06	L293D/L293N
22	Accelerometer	05	
23	Fire sensor	04	
24	Voltage Regulator	10	
25	LDR sensor	10	

Sl. No	Components available for mini project	Number	Specifications
26	Encoder	05	
27	FTD1232 USB to TTL serial connector adapter module	02	
28	Capacitive touch digital keypad	10	
29	LDR	10	
30	Relay 8 channel	03	
31	Relay 4 channel	05	
32	Relay 1 channel	01	
33	LCD	10	14*2
34	Stepper motor	05	25kg
34	Servo motor	05	
36	Solder gun	05	
37	Lead	05	
38	Flux	03	
39	Pulleys and gears	1 set	
40	Multimeter	02	
41	Wireless Nano USB connector	01	
42	IMU breakout	12	
43	Single strand and Multi strand wires	1 bundle	
44	Wi-Fi shield Uno and Mega Compatible	02	
45	Ethernet shield for Arduino	01	
46	Breadboard	14	
47	Beagle bone	02	
48	USB adapter for Raspberry pi	03	
49	Wire stripper	05	
50	Drill machine with drill bit	01	

Sl. No	Components available for mini project	Number	Specifications
51	Screwdriver set	02	
52	Allen keys	01 set	
53	Jumper wires	01 set	
54	Power connector to Arduino	10	
55	Crocodile clips	20	
56	Nuts and Screws and Washers	1 set	
57	Resistor sets	1 bunch	
58	Long USB connector	03	
59	ARM controller	04	
60	Lego connecting cable	04	
61	Scissor	01	
62	Potentiometer	05	
63	Power connector	04	
64	Lego kit	04	
65	Balance charger	04	12V
66	DC Regulated power supply	02	
67	DC Regulated power supply	03	302D
68	LED Acid battery	04	12V and 1.3AH
69	Real sense depth camera	03	D435
70	Logitech web camera	02	720P
71	RP Lidar	05	
72	Clear path Turtle Bot 2	02	
73	My Rio kit	01	
74	Elvis kit	01	