TF2

Applications: compute inv and fwd kinematics, obstacle avoidance, convey robot location, convert sensor data from one ref to another, analyze multiple robot data in a global frame

Coordinate Frames: Gives spatial relationship between objects and axes

Static Transforms: don't change wrt time. Eg: camera wrt base does not change transform
Dynamic Transforms: moves wrt time. Eg: a manipulator arm on a mobile robot can move

KEVIN WOOD AND OFFICIAL DOCS:-

ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py: launches 2 turtles in turtlesim
ros2 run turtlesim turtle_teleop_key

When u move turtle1, u will notice turtle2 also moves wrt to it. This is because of 3 frames: world, turtle1 and turtle2. tf2_broadcaster publishes coordinates and tf2_listener calculates dist and moves turtle2 towards turtle1

view_frames: creates a diagram of the frames being broadcast by tf2

tf2_echo: reports the transform between any two frames

ros2 run tf2_ros tf2_echo [source_frame] [target_frame]
Eg*: ros2 run tf2_ros tf2_echo turtle2 turtle1*

Transform changes as relativity between the turtles changes.

To visualize in rviz,
ros2 run rviz2 rviz2 -d $(ros2 pkg prefix --share turtle_tf2_py)/rviz/turtle_rviz.rviz

Writing a static tf2 broadcaster
The code in the docs wont work as there is nothing to keep the node working as the transform is static and only publishes at startup

An easier way is this command:

Rotation in radians:
ros2 run tf2_ros static_transform_publisher --x 0 --y 0 --z 1 --yaw 0 --pitch 0 --roll 0 --frame-id world --child-frame-id mystaticturtle

Rotation in quaternions:
ros2 run tf2_ros static_transform_publisher --x 0 --y 0 --z 1 --qx 0 --qy 0 --qz 0 --qw 1 --frame-id world --child-frame-id mystaticturtle

You can also include this in a launch file

No loops allowed in a tf2 tree. Hence only one parent but can have multiple children

Now when multiple frames are present and u want to change target frame:
ros2 launch tf2_package adding_fixed_frame.launch.py target_frame:=carrot1

else:

```
 demo_nodes = IncludeLaunchDescription(
        ...,
        launch_arguments={'target_frame': 'carrot1'}.items(),
        )
```
in launch file

How to debug tf2 files

Firstly, we need to find out what exactly we are asking tf2 to do. Therefore, we go into the part of the code that is using tf2. Then try the code, tf2_echo and view_frames. And finally you can use tf2_monitor like
ros2 run tf2_ros tf2_monitor turtle2 turtle1

Important Functions:

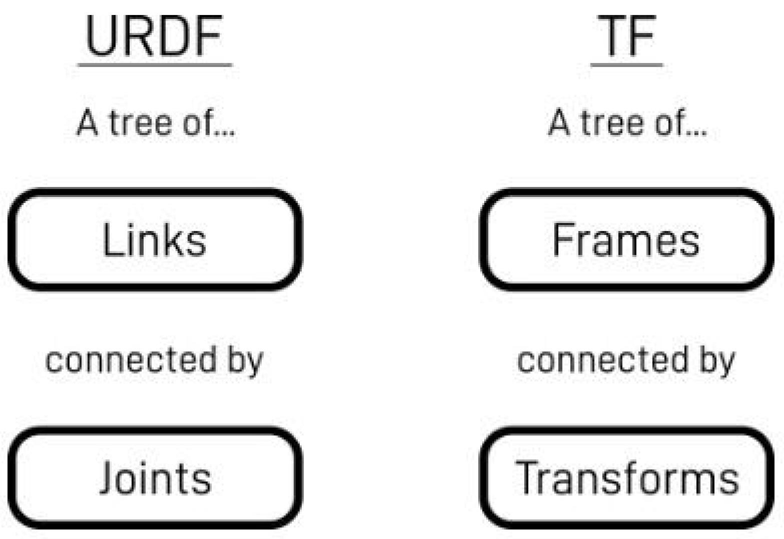
StaticTransformBroadcaster(self) → Sends static coordinate frame transforms to TF2 system
TransformStamped() → Defines a single transformation between two coordinate frames at a specific time
TransformBroadcaster(self) →
TransformListener(self.tf_buffer,self) → Subscribes to transform data and populates a buffer so other components can query transforms

Quaternion: a 4-tuple representation of orientation. are very efficient for analyzing situations where rotations in three dimensions are involved.
They are tough to analyze so first think in terms of euler angles and then convert. To apply the rotation of one quaternion to a pose, simply multiply the previous quaternion of the pose by the quaternion representing the desired rotation. An easy way to invert a quaternion is to negate the w-component

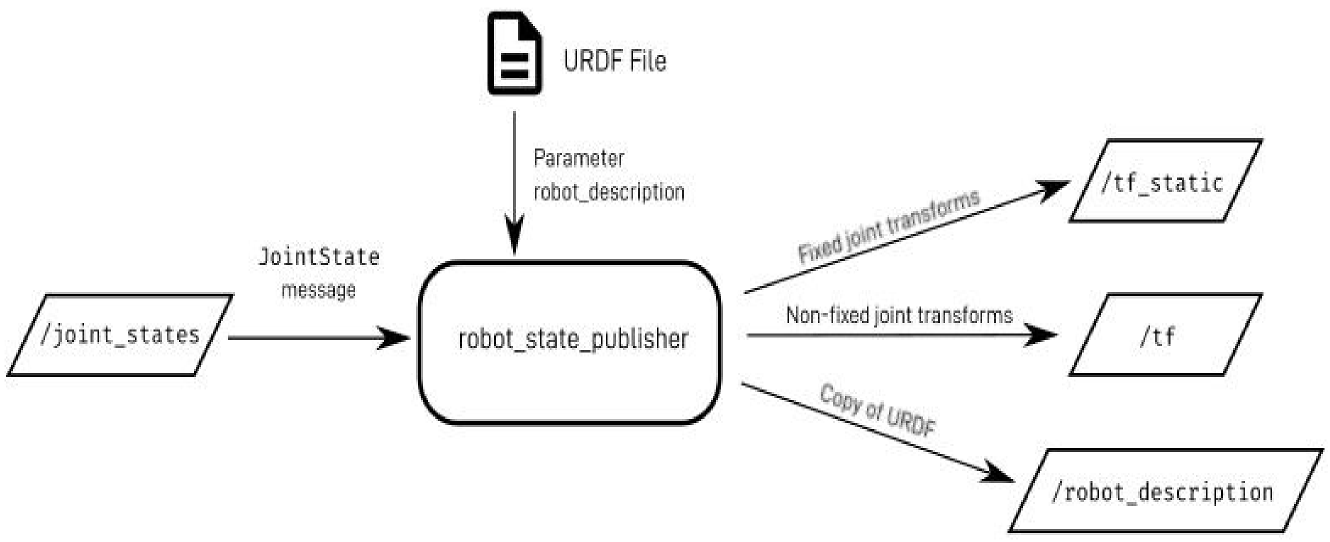
ARTICULATED ROBOTICS:-
files are robot_state_publisher.launch.py, example.urdf,

Underneath, the TF2 libraries are still using topics /*tf* and /*tf_static* to handle all this communication, but because we don't actually publish to the topic ourselves, we call it broadcasting and listening instead of publishing and subscribing.

ros2 run tf2_ros static_transform_publisher x y z yaw pitch roll parent_frame child_frame



as they are similar, there is a ROS node called robot_state_publisher which can take in a URDF file and automatically broadcast all the transforms from it. It will also publish the full contents of the URDF file to the topic /robot_description so that any other nodes that need it can all use the same file.



For static transform:

ros2 run tf2_ros static_transform_publisher 2 1 0 0.785 0 0 world robot_1

ros2 run tf2_ros static_transform_publisher 1 0 0 0 0 0 robot_1 robot_2

Then open rviz2, add TF, choose world frame as world or robot_1 and robot_2. In rviz, arrows go from child to parent

For dynamic transform:

ros2 launch tf2_package robot_state_publisher.launch.py

ros2 run joint_state_publisher_gui joint_state_publisher_gui visualize in rviz. U can also see robot using add robotmodel

ALEXANDER HABER:

files are model.urdf, test1, test2, display.launch.py

ros2 launch tf2_package display.launch.py

go to rviz, add robot model and frame to base_link

ros2 run tf2_package test1.py

or ros2 run tf2_package test2.py