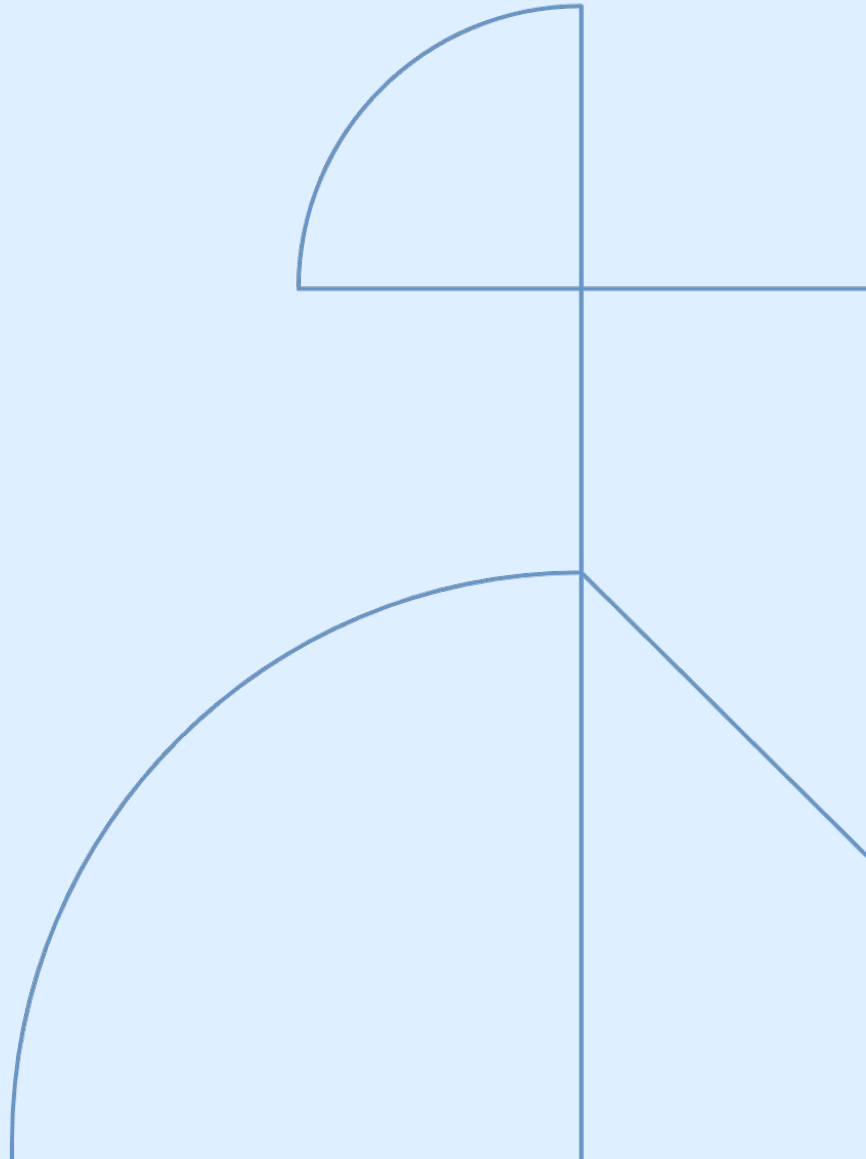
DD2419 Project Course in Robotics and Autonomous Systems

Lecture 2: TF, etc

TF



TF

What is a transform?

TF

What is a transform?

- Parent frame

TF

What is a transform?

- Parent frame
- Child frame

TF

What is a transform?

- Parent frame
- Child frame
- Translation

TF

What is a transform?

- Parent frame
- Child frame
- Translation
- Rotation

TF

What is a transform?

- Parent frame
- Child frame
- Translation
- Rotation
- ?

TF

What is a transform?

- Parent frame
- Child frame
- Translation
- Rotation
- Timestamp

TF Buffer

- Stores known frames and transforms between them
- If we take a look at the constructor:
  - tf2_ros.buffer.Buffer(cache_time=None)
    - cache_time is how long transforms should be stored in the buffer. Default is 10 seconds
- Note that the buffer needs some time before it has received the transforms
  - If you call buffer.can_transform(...) right after you have constructed the buffer or transform it will not be able to retrieve it

TF Buffer Different Methods

- can_transform(target_frame: str, source_frame: str, time: Time, timeout: Duration = Duration())
  - Check if it is possible to transform between two frames
- lookup_transform(target_frame: str, source_frame: str, time: Time, timeout: Duration = Duration())
  - Get a transform between two frames
- do_transform_point(point: PointStamped, transform: TransformStamped)
  - Apply a transform to a point
- do_transform_vector3(vector3: Vector3Stamped, transform: TransformStamped)
  - Apply a transform to a Vector3
- do_transform_pose(pose: Pose, transform: TransformStamped)
  - Apply a transform to a Pose
- do_transform_pose_stamped(pose: PoseStamped, transform: TransformStamped)
  - Apply a transform to a Pose

- https://github.com/ros2/geometry2/blob/humble/tf2_ros_py/tf2_ros/buffer.py
- https://github.com/ros2/geometry2/blob/humble/tf2_geometry_msgs/src/tf2_geometry_msgs/tf2_geometry_msgs.py

Timestamp

```
stamp = self.get_clock().now()

buffer.can_transform("camera_link", "map", stamp)
```

Timestamp

stamp = self.get_clock().now()

buffer.can_transform(“camera_link”, “map”, stamp)

- If the transform between camera_link and map is static then this is fine

Timestamp

stamp = self.get_clock().now()

buffer.can_transform("camera_link", "map", stamp)

- If the transform between camera_link and map is static then this is fine
- It not, this will fail most of the times
  - Since you are checking if the transform exists at the current time
  - For this to succeed, either
    1. the broadcasting of the dynamic transform has to happen between the two lines
    2. the broadcasting of the dynamic transform had to have a future timestamp

Timestamp

```
stamp = self.get_clock().now()

timeout = rclpy.duration.Duration(seconds=0.5)

buffer.can_transform("camera_link", "map", stamp, timeout)
```

- If the transform between camera_link and map is static then this is fine
- It not, this will fail most of the times
  - Since you are checking if the transform exists at the current time
  - For this to succeed, either
    1. the broadcasting of the dynamic transform has to happen between the two lines
    2. the broadcasting of the dynamic transform had to have a future timestamp

Timestamp

```
stamp = self.get_clock().now()

timeout = rclpy.duration.Duration(seconds=0.5)

buffer.can_transform("camera_link", "map", stamp, timeout)
```

- If the transform between camera_link and map is static then this is fine
- It not, this will fail most of the times
  - Since you are checking if the transform exists at the current time
  - For this to succeed, either
    1. the broadcasting of the dynamic transform has to happen between the two lines
    2. the broadcasting of the dynamic transform had to have a future timestamp
- Now it will wait up to timeout number of seconds for the transform to be available at time stamp

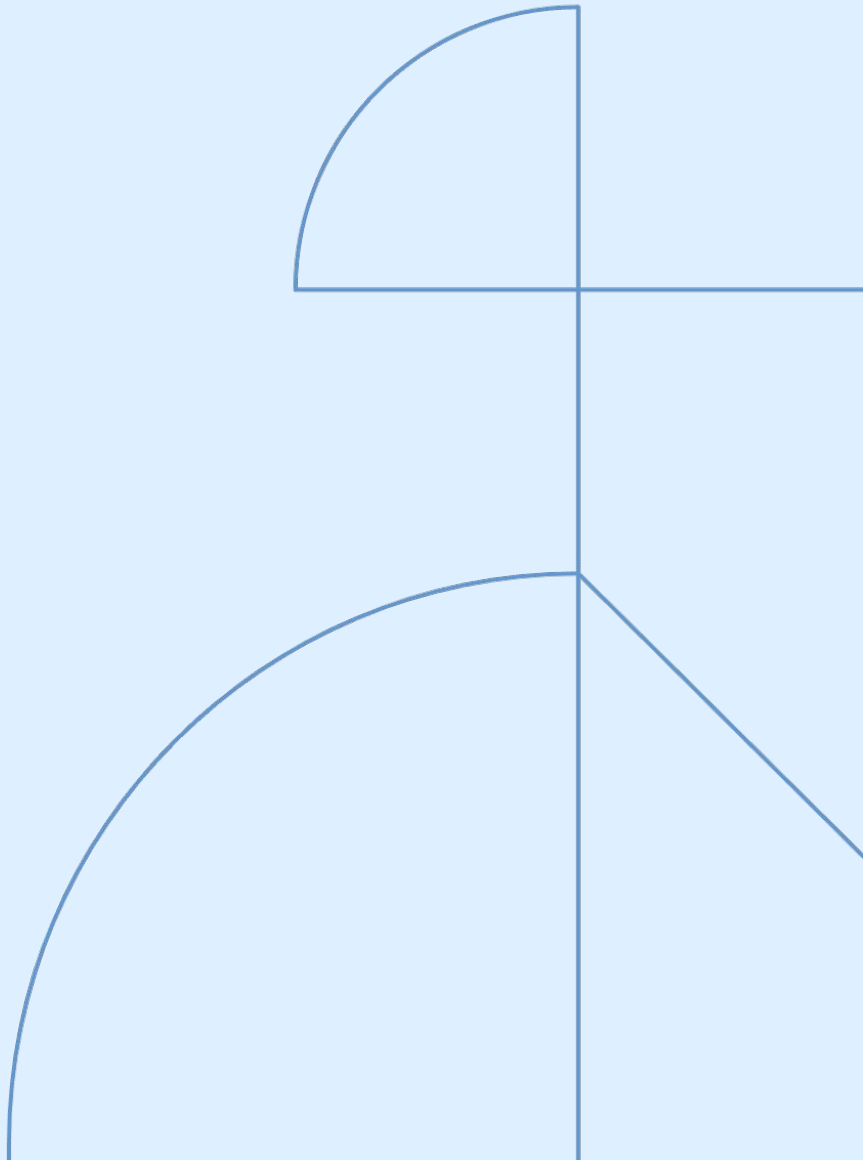
Always Include a Timestamp!

- If you do not include a timestamp when you broadcast a dynamic transform it is impossible to know at which time the transform is valid
  - It can be seen as a static transform in this case, since the timestamp is 0 if you do not set it
- This will cause a lot of problems for you after a while

Different Times

- self.get_clock().now()
  - Current time
- rclpy.time.Time() == rclpy.time.Time(seconds=0)
  - Input this into transform functions to get latest **available** transform
- *message*.header.stamp
  - Use the stamp of another message if it is important that you get the transform at that specific time

ROS Environment



ROS Environment

Read https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Configuring-ROS2-Environment.html

Otherwise you **will** have networks problems during the project

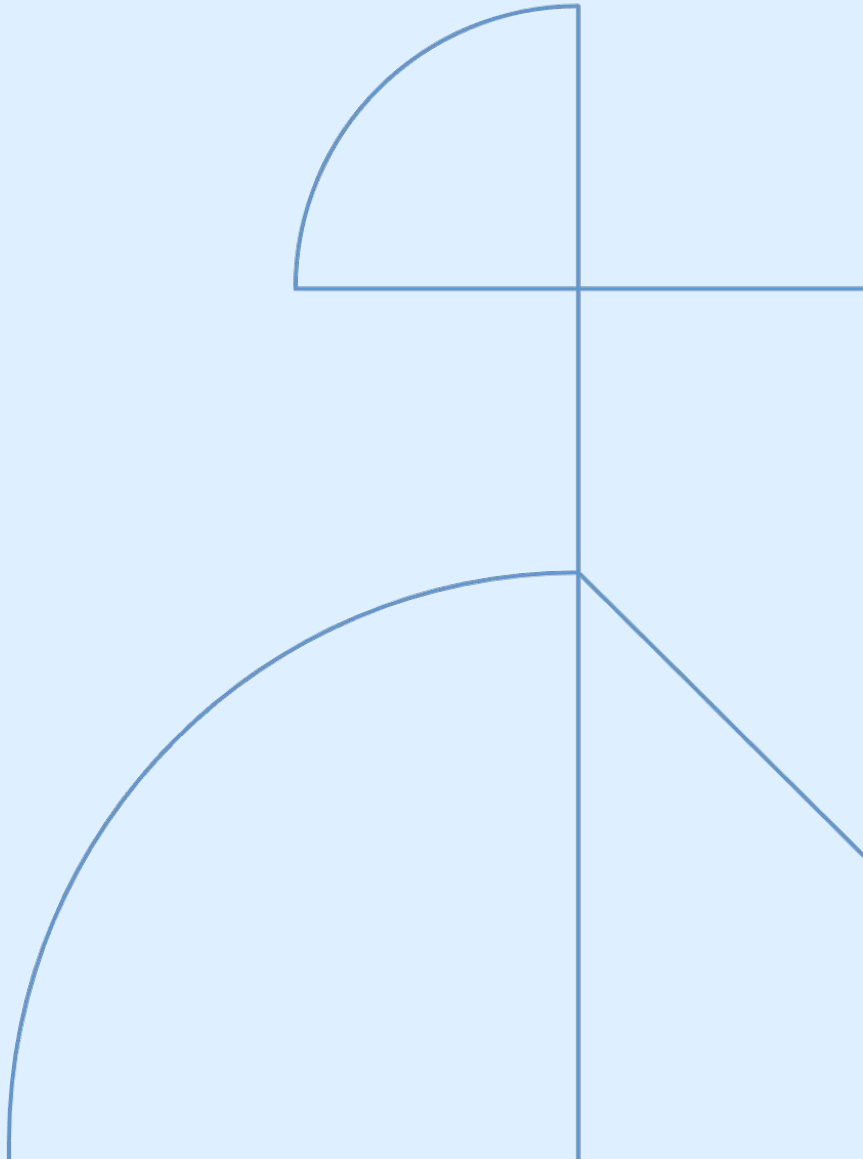
ROS Environment

Read https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Configuring-ROS2-Environment.html

Otherwise you **will** have networks problems during the project

Probably during boot camp as well if you are in A:235

Etc



Fun

We are going to write a function that adds two numbers, **a** and **b**, and then add **2** to that and returns the total sum

Fun

We are going to write a function that adds two numbers, **a** and **b**, and then add **2** to that and returns the total sum

1.
```
def add_plus_2(a, b):
    return a + b + 3
```

Fun

We are going to write a function that adds two numbers, **a** and **b**, and then adds **2** to that and returns the total sum

1.
```
def add_plus_2(a, b):
    return a + b + 3
```
2.
```
def add_plus_2(a, b):
    return a * b + 3
```

Fun

We are going to write a function that adds two numbers, **a** and **b**, and then add **2** to that and returns the total sum

1.
```
def add_plus_2(a, b):
    return a + b + 3
```
2.
```
def add_plus_2(a, b):
    return a * b + 3
```
3.
```
def add_plus_2(a, b):
    return a / b + 3
```

Fun

We are going to write a function that adds two numbers, **a** and **b**, and then add **2** to that and returns the total sum

1.
```
def add_plus_2(a, b):
    return a + b + 3
```
2.
```
def add_plus_2(a, b):
    return a * b + 3
```
3.
```
def add_plus_2(a, b):
    return a / b + 3
```

Test cases:

- add_plus_2(1, 1) == 4
- add_plus_2(2, 1) == 5
- add_plus_2(3, 1) == 6