Multi-Robot Experiments with TurtleBots & MATLAB

4DM70 – ANALYSIS AND DESIGN OF NETWORKED DYNAMICAL SYSTEMS

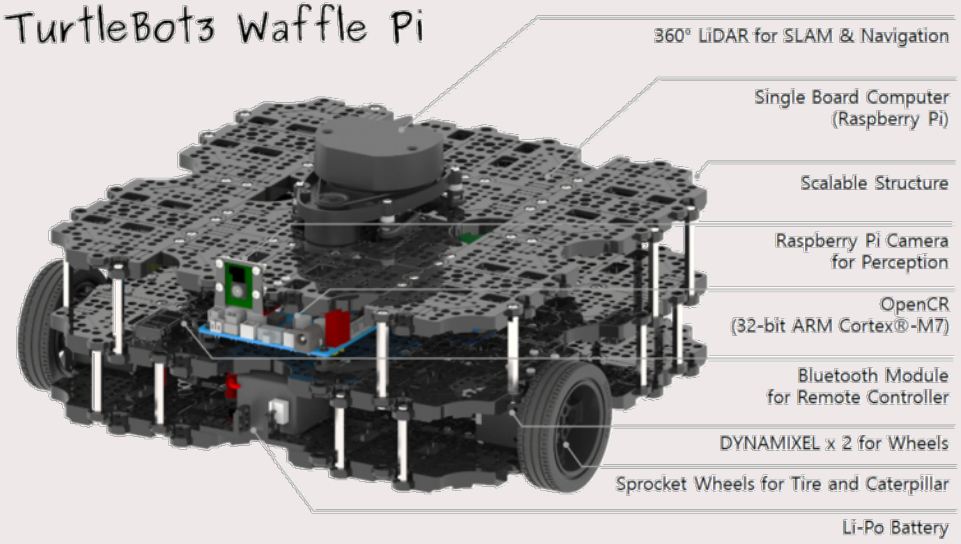
Ömür Arslan, Assistant Professor of Robotics, E-mail: o.arslan@tue.nl

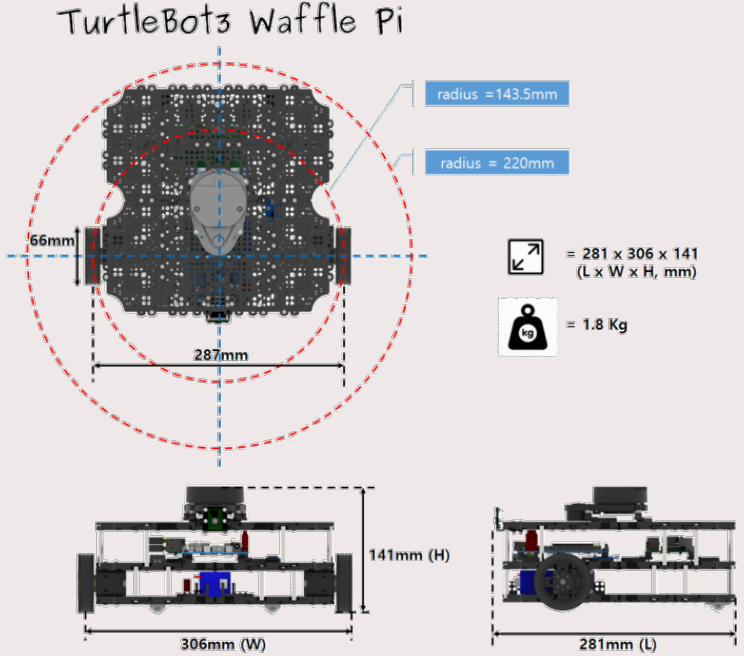
Department of Mechanical Engineering | Robotics (RBT)

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Part I – Experimental Setup

Robot Platform





Maximum Translational Velocity	**0.26 m/s**
Maximum Rotational Velocity	1.82 rad/s (104.27 deg/s)

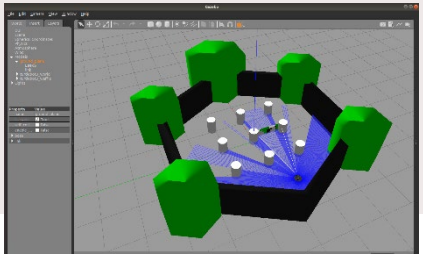
ROS: Robot Operating System
VRPN: Virtual Reality Peripheral Network

Experimental Setup

External Computer
(Windows, MATLAB ROS Toolbox)

ROS

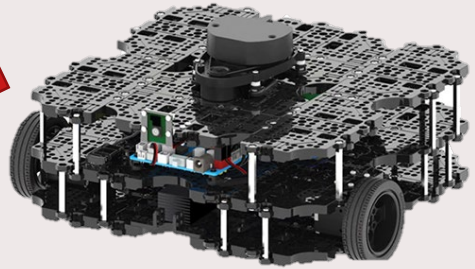
Central/Simulation Computer
(Ubuntu, ROS Master, Gazebo)



VRPN

Motion Capture
(Windows, Motive)

ROS



TurtleBot3
(Ubuntu, ROS)

TU/e

Network Connection

- Connect to the appropriate available CORE WiFi network (ask a lab instructor)
  - "**CoreMobileWiFi**" Network (Password: **core_mobile-123!**)
  - "**CoreTurtlebotWiFi**" network (Password: **core_turtlebot-123!**)
  - "**CoreMinicarWiFi**" network (Password: **core_minicar-123!**)
- Check your IP address (192.168.6.???), e.g., `ipconfig`
- Check your connection with a simulation computer
  - Simulation Computer **#i**: `ping 192.168.4.10i`
  - "**CoreMobileWiFi**" Router: `ping 192.168.0.1`
  - "**CoreTurtlebotWiFi**" Router: `ping 192.168.2.1`
  - "**CoreMinicarWiFi**" Router: `ping 192.168.1.1`

Start Gazebo on Central/Simulation Computer

- Open a terminal, e.g., **Ctrl + Alt + T**
- Run the following ROS command in the terminal

```
ros2 launch core_tue4dm70_turtlebot3_simulate team1_threeturtlebot3_gazebo_arena.launch.py
```

- To stop simulation, use **Ctrl + C** in the terminal

List of Example Launch Files

- Three TurtleBots for Team 1: team1_threeturtlebot3_gazebo_arena.launch.py
- Three TurtleBots for Team 2: team2_threeturtlebot3_gazebo_arena.launch.py
- Three TurtleBots for Team 3: team3_threeturtlebot3_gazebo_arena.launch.py
- One Team of Three TurtleBots: oneteam_threeturtlebot3_gazebo_arena.launch.py
- Two Teams of Three TurtleBots: twoteam_threeturtlebot3_gazebo_arena.launch.py
- Three Teams of Three TurtleBots: threeteam_threeturtlebot3_gazebo_arena.launch.py
- One TurtleBot & One Goal: demo_turtlebot3_goal_gazebo_empty.launch.py
- Three TurtleBots & One Goal: demo_threeturtlebot3_goal_gazebo_empty.launch.py

```
$ ros2 launch core_tue4dm70_turtlebot3_simulate LAUNCH_FILE_NAME
```

Network: **CoreMobileWiFi**
Password: **core_mobile-123!**

Network: **CoreTurtlebotWiFi**
Password: **core_turtlebot-123!**

Network: **CoreMinicarWiFi**
Password: **core_minicar-123!**

TurtleBot3 MATLAB Software

- Download sample MATLAB code **on your (external) computer**
  https://gitlab.tue.nl/core_robotics/courses/tue4dm70/core_tue4dm70_turtlebot3_matlab_humble

DOWNLOAD

CORE Multi-Robot Simulation Software, built on Ubuntu 22.04 + ROS Humble + Gazebo Fortress, is available at https://gitlab.tue.nl/core_robotics/courses/tue4dm70/core_tue4dm70_humble

Start MATLAB

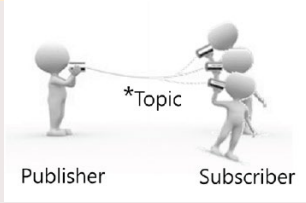
- Open the “core_tue4dm70_turtlebot3_matlab_humble” folder in MATLAB
- Run **`startup.m`**

Network: **CoreMobileWiFi**
Password: **core_mobile-123!**

Network: **CoreTurtlebotWiFi**
Password: **core_turtlebot-123!**

Network: **CoreMinicarWiFi**
Password: **core_minicar-123!**

Check ROS Topic List in MATLAB



- Run “**help demo_ros2_topic_list**”

```
%%% PLEASE MAKE SURE YOUR ROS SETTINGS ARE CORRECT! %%%

Example:
  ROS_DOMAIN_ID = "0";
  setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID);
  ros2 topic list
```

- Update **ROS_DOMAIN_ID**

```
>> setenv("ROS_DOMAIN_ID", ???); %  SIMULATION COMPUTER ID
```

- Run **demo_rostopic_list()**

```
>> ros2 topic list
/clock
/mocap/turtlebot1/pose
/mocap/turtlebot2/pose
/mocap/turtlebot3/pose
/mocap/turtlebot4/pose
```

Your ROS_DOMAIN_ID should match the ROS_DOMAIN_ID of the Central Simulation Computer
ROS_DOMAIN_ID = Computer_Number

Check your connection with the Simulation Computer using its IP address
`ping 192.168.?.10?`

Part II – TurtleBot Teleoperation

Network: **CoreMobileWiFi**
Password: **core_mobile-123!**

Network: **CoreTurtlebotWiFi**
Password: **core_turtlebot-123!**

Network: **CoreMinicarWiFi**
Password: **core_minicar-123!**

TurtleBot Teleoperation in MATLAB

- Run **help demo_turtlebot_teleop**

```
>> help demo_turtlebot_teleop

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLEASE MAKE SURE YOUR ROS SETTINGS ARE CORRECT! %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID)
   ctrlTopic = '/turtlebot1/cmd_vel_ctrl';
   demo_turtlebot_teleop(ctrlTopic);
```

- Set **ROS_DOMAIN_ID**

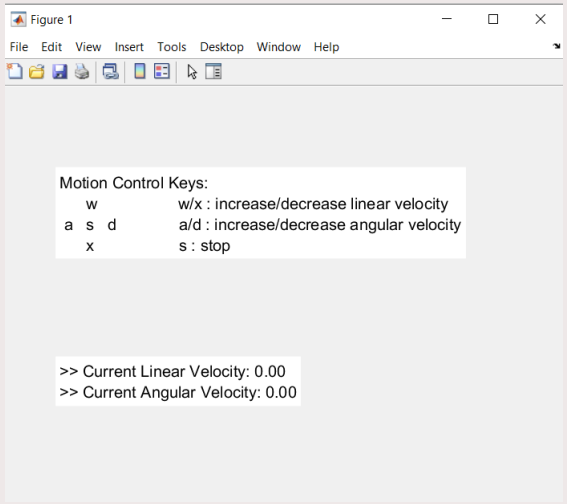
```
>> setenv("ROS_DOMAIN_ID", ???); %  SIMULATION COMPUTER ID
```

- Set TurtleBot control topic name

```
>> ctrlTopic = '/turtlebot1/cmd_vel_ctrl';; % TURTLEBOT ID
```

- Run **demo_turtlebot_teleop**

```
>> demo_turtlebot_teleop(ctrlTopic);
```



- To stop, close the figure or use **Ctrl-C**

TU/e

TurtleBot Teleoperation: Code Structure

Initialization

Interface

Control Loop

```
function demo_turtlebot_teleop(ctrlTopic)
% Authors: Omur Arslan, o.arslan@tue.nl ...

%% ROS Node Settings
% Register a turtlebot teleoperation node at the ROS master
nodeName = ['/matlab_turtlebot_teleop_' int2str(1000000000*rand(1))];
teleopNode = ros2node(nodeName);
% Create and register a turtlebot control publisher at the ROS master
ctrl_pub = ros2publisher(teleopNode, ctrlTopic, "geometry_msgs/Twist");
ctrl_msg = ros2message("geometry_msgs/Twist");
ctrl_msg_handle = core_ros2tools.ROS2MessageHandle(ctrl_msg);

%% Visualization for turtlebot control interface
h.figure = figure('KeyPressFcn', @(src, data) KeyPressFcn(src, data, ctrl_msg_handle),...
                  'DeleteFcn', @(src, event) DeleteFcn (src, event, ctrl_msg_handle));
h.axes = axes('Position', [0.0, 0.0, 1.0, 1.0], 'XLim', [0, 1], 'YLim', [0,1], 'Visible', 'off');
inputText = [sprintf('Motion Control Keys:\n')...
             sprintf('      w                 w/x : increase/decrease linear velocity\n')...
             sprintf(' a    s    d           a/d : increase/decrease angular velocity\n')...
             sprintf('      x                 s : stop')];
h.keytext = text(h.axes, 0.1, 0.7, inputText, 'FontSize', 12, 'BackgroundColor', 'w');
fcmdtext = @(cmd) sprintf('>> Current Linear Velocity: %.2f \n>> Current Angular Velocity: %.2f',...
                          cmd.linear.x, cmd.angular.z);
h.cmdtext = text(h.axes, 0.1, 0.3, fcmdtext(ctrl_msg_handle.msg), 'FontSize', 12, 'BackgroundColor', 'w');

cleanupObj = onCleanup(@() demo_turtlebot_teleop_cleanup(h.figure));

%% Start teleoperation loop
while (true)
    % Terminate if the figure is closed
    if not(ishandle(h.figure))
        break;
    end
    % Send turtlebot control message
    ctrl_pub.send(ctrl_msg_handle.msg);
    % Update turtlebot control info
    set(h.cmdtext, 'String',  fcmdtext(ctrl_msg_handle.msg));
    % Pause for an update rate of at most 10Hz
    pause(0.1);
end
end
```

KeyPress Event

```
function KeyPressFcn(src, data, ctrl_msg_handle)
    % Callback function for the figure-keypressed event to set up turtlebot control inputs

    % Turtlebot control limits
    maxLinSpeed = 0.26;
    maxAngSpeed = 1.82;

    % Adjust turtlebot control inputs
    switch data.Key
        case 'w'
            ctrl_msg_handle.msg.linear.x = ctrl_msg_handle.msg.linear.x + 0.1*maxLinSpeed;
            ctrl_msg_handle.msg.linear.x = min(ctrl_msg_handle.msg.linear.x, maxLinSpeed);
        case 'x'
            ctrl_msg_handle.msg.linear.x = ctrl_msg_handle.msg.linear.x - 0.1*maxLinSpeed;
            ctrl_msg_handle.msg.linear.x = max(ctrl_msg_handle.msg.linear.x, -maxLinSpeed);
        case 'a'
            ctrl_msg_handle.msg.angular.z = ctrl_msg_handle.msg.angular.z + 0.1 * maxAngSpeed;
            ctrl_msg_handle.msg.angular.z = min(ctrl_msg_handle.msg.angular.z, maxAngSpeed);
        case 'd'
            ctrl_msg_handle.msg.angular.z = ctrl_msg_handle.msg.angular.z - 0.1 * maxAngSpeed;
            ctrl_msg_handle.msg.angular.z = max(ctrl_msg_handle.msg.angular.z, -maxAngSpeed);
        case 's'
            ctrl_msg_handle.msg.linear.x = 0;
            ctrl_msg_handle.msg.angular.z = 0;
    end
end
```

Figure Delete Event

```
function DeleteFcn(src, event, ctrl_msg_handle)
    % Callback function for the figure-closed event
    % Stop turtlebot by setting control inputs to zero
    ctrl_msg_handle.msg.linear.x = 0;
    ctrl_msg_handle.msg.angular.z = 0;
end
```

Cleanup Event

```
function demo_turtlebot_teleop_cleanup(figure_handle)
    %Cleanup tasks upon function completion
    delete(figure_handle);
end
```


TU/e

TurtleBot Teleoperation: Initialization

```
19      %% ROS Node Settings
20      % Register a turtlebot teleoperation node at the ROS master
21      nodeName = ['/matlab_turtlebot_teleop_' int2str(1000000000*rand(1))];
22      teleopNode = ros2node(nodeName);
23      % Create and register a turtlebot control publisher at the ROS master
24      ctrl_pub = ros2publisher(teleopNode, ctrlTopic, "geometry_msgs/Twist");
25      ctrl_msg = ros2message("geometry_msgs/Twist");
26      ctrl_msg_handle = core_ros2tools.ROS2MessageHandle(ctrl_msg);
```

TurtleBot Teleoperation: Interface

```
28      %% Visualization for turtlebot control interface
29      h.figure = figure('KeyPressFcn', @(src, data) KeyPressFcn(src, data, ctrl_msg_handle),...
30                        'DeleteFcn', @(src, event) DeleteFcn (src, event, ctrl_msg_handle));
31      h.axes = axes('Position', [0.0, 0.0, 1.0, 1.0], 'XLim', [0, 1], 'YLim', [0,1], 'Visible', 'off');
32      inputText = [sprintf('Motion Control Keys:\n')...
33                   sprintf('      w                  w/x : increase/decrease linear velocity\n')...
34                   sprintf(' a    s    d              a/d : increase/decrease angular velocity\n')...
35                   sprintf('      x                   s : stop')];
36      h.keytext = text(h.axes, 0.1, 0.7, inputText, 'FontSize', 12, 'BackgroundColor', 'w');
37      fcmdtext = @(cmd) sprintf('>> Current Linear Velocity: %.2f \n>> Current Angular Velocity: %.2f',...
38                                cmd.linear.x, cmd.angular.z);
39      h.cmdtext = text(h.axes, 0.1, 0.3, fcmdtext(ctrl_msg_handle.msg), 'FontSize', 12, 'BackgroundColor', 'w');
40
41      cleanupObj = onCleanup(@() demo_turtlebot_teleop_cleanup(h.figure));
```


TU/e

TurtleBot Teleoperation: Control Loop

```
43      %% Start teleoperation loop
44      while (true)
45          % Terminate if the figure is closed
46          if not(ishandle(h.figure))
47              break;
48          end
49          % Send turtlebot control message
50          ctrl_pub.send(ctrl_msg_handle.msg);
51          % Update turtlebot control info
52          set(h.cmdtext, 'String',  fcmdtext(ctrl_msg_handle.msg));
53          % Pause for an update rate of at most 10Hz
54          pause(0.1);
55      end
```

TurtleBot Teleoperation: KeyPress Event

```
function KeyPressFcn(src, data, ctrl_msg_handle)
    % Callback function for the figure-keypressed event to set up turtlebot control inputs

    % Turtlebot control limits
    maxLinSpeed = 0.26;
    maxAngSpeed = 1.82;

    % Adjust turtlebot control inputs
    switch data.Key
        case 'w'
            ctrl_msg_handle.msg.linear.x = ctrl_msg_handle.msg.linear.x + 0.1*maxLinSpeed;
            ctrl_msg_handle.msg.linear.x = min(ctrl_msg_handle.msg.linear.x, maxLinSpeed);
        case 'x'
            ctrl_msg_handle.msg.linear.x = ctrl_msg_handle.msg.linear.x - 0.1*maxLinSpeed;
            ctrl_msg_handle.msg.linear.x = max(ctrl_msg_handle.msg.linear.x, -maxLinSpeed);
        case 'a'
            ctrl_msg_handle.msg.angular.z = ctrl_msg_handle.msg.angular.z + 0.1 * maxAngSpeed;
            ctrl_msg_handle.msg.angular.z = min(ctrl_msg_handle.msg.angular.z, maxAngSpeed);
        case 'd'
            ctrl_msg_handle.msg.angular.z = ctrl_msg_handle.msg.angular.z - 0.1 * maxAngSpeed;
            ctrl_msg_handle.msg.angular.z = max(ctrl_msg_handle.msg.angular.z, -maxAngSpeed);
        case 's'
            ctrl_msg_handle.msg.linear.x = 0;
            ctrl_msg_handle.msg.angular.z = 0;
    end
end
```

TurtleBot Teleoperation: Delete & Cleanup Events

```
85  function DeleteFcn(src, event, ctrl_msg_handle)
86      % Callback function for the figure-closed event
87      % Stop turtlebot by setting control inputs to zero
88      ctrl_msg_handle.msg.linear.x = 0;
89      ctrl_msg_handle.msg.angular.z = 0;
90  end
```

```
92  function demo_turtlebot_teleop_cleanup(figure_handle)
93      %Cleanup tasks upon function completion
94      delete(figure_handle);
95  end
```

Use **demo_ros2_topic_list()** to find the correct TurtleBot control topic

Make sure Gazebo running on the simulation computer:
$ ros2 launch core_tue4dm70_turtlebot3_simulate team1_threeturtlebot3goal_gazebo_arena.launch

Network: **CoreMobileWiFi**
Password: **core_mobile-123!**

Network: **CoreTurtlebotWiFi**
Password: **core_turtlebot-123!**

Network: **CoreMinicarWiFi**
Password: **core_minicar-123!**

TurtleBot Teleoperation in MATLAB

- Run **help demo_turtlebot_teleop**

```
>> help demo_turtlebot_teleop

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLEASE MAKE SURE YOUR ROS SETTINGS ARE CORRECT! %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID)
   ctrlTopic = '/turtlebot1/cmd_vel_ctrl';
   demo_turtlebot_teleop(ctrlTopic);
```

- Set **ROS_DOMAIN_ID**

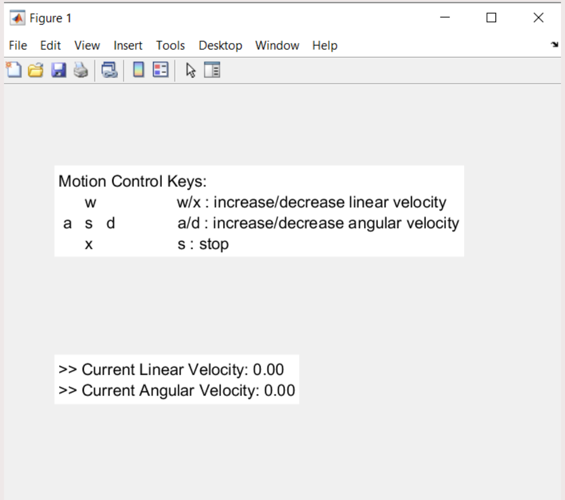
```
>> setenv("ROS_DOMAIN_ID", ???); %  SIMULATION COMPUTER ID
```

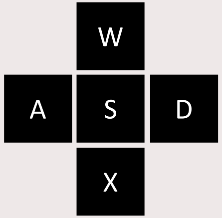
- Set TurtleBot control topic name

```
>> ctrlTopic = '/turtlebot1/cmd_vel_ctrl';; % TURTLEBOT ID
```

- Run **demo_turtlebot_teleop**

```
>> demo_turtlebot_teleop(ctrlTopic);
```





- To stop, close the figure or use **Ctrl-C**

Part III – TurtleBot Leader-Follower Coordination

Use **demo_ros2_topic_list()** to find the correct pose and control topics

Make sure Gazebo running on the simulation computer:
$ ros2 launch core_tue4dm70_turtlebot3_simulate oneteam_threeturtlebot3goal_gazebo_empty.launch

Network: **CoreMobileWiFi**
Password: **core_mobile-123!**

Network: **CoreTurtlebotWiFi**
Password: **core_turtlebot-123!**

Network: **CoreMinicarWiFi**
Password: **core_minicar-123!**

TurtleBot Leader-Follower Coordination

- Run **help demo_turtlebot_leaderfollower**

```
>> help demo_turtlebot_leaderfollower

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLEASE MAKE SURE YOUR ROS SETTINGS ARE CORRECT! %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID)
   leaderPoseTopic = '/mocap/turtlebot1/pose';
   followerPoseTopic = '/mocap/turtlebot2/pose';
   followerCtrlTopic = '/turtlebot2/cmd_vel_ctrl';
   demo_turtlebot_leaderfollower(leaderPoseTopic, followerPoseTopic, followerCtrlTopic);
```

- Set pose & control topic names

```
>> leaderPoseTopic = '/mocap/turtlebot1/pose';
>> followerPoseTopic = '/mocap/turtlebot2/pose';
>> followerCtrlTopic = '/turtlebot2/cmd_vel_ctrl';
```

- Set **ROS_DOMAIN_ID**

```
>> setenv("ROS_DOMAIN_ID", ???); % SIMULATION COMPUTER ID
```

- Stop by **Ctrl-C**

- Run **demo_turtlebot_leaderfollower**

```
>> demo_turtlebot_leaderfollower(leaderPoseTopic, followerPoseTopic, followerCtrlTopic);
Leader-Follower Control is running...
Follower pose is not received!
Leader pose is not received!
Leader Pose: (-4.00, -0.17, 0.94), Follower Pose(-4.12, -0.60, 1.05)
Leader Pose: (-4.00, -0.17, 0.94), Follower Pose(-4.12, -0.60, 1.05)
```

Leader-Follower: Code Structure

Initialization

Control Loop

```
function demo_turtlebot_leaderfollower(leaderPoseTopic, followerPoseTopic, followerCtrlTopic)
% Authors: Omur Arslan, o.arslan@tue.nl ...

    %% ROS Node Settings
    % Register the leader-follower node at the ROS master
    nodeName = ['/matlab_turtlebot_leaderfollower_' int2str(100000000*rand(1))]; % Node Name
    leaderfollowerNode = ros2node(nodeName);
    % Create and register the follower control publisher at the ROS master
    [followerCtrlPublisher, followerCtrlMessage] = ros2publisher(leaderfollowerNode, followerCtrlTopic, "geometry_msgs/Twist");

    % Create and register the leader and follower pose subcribers at the ROS master
    leaderPoseSubscriber = ros2subscriber(leaderfollowerNode, leaderPoseTopic, 'geometry_msgs/PoseStamped', 'History', 'keeplast', 'Depth', 1, 'Reliability','reliable');
    followerPoseSubscriber = ros2subscriber(leaderfollowerNode, followerPoseTopic, 'geometry_msgs/PoseStamped', 'History', 'keeplast', 'Depth', 1, 'Reliability','reliable');

    %% Leader-Follower Control Loop
    disp('Leader-Follower Control is running...');
    while (true)
        % Pause for an update rate of at most 10Hz
        pause(0.1);
        % Get the lastest information from the ROS network
        leaderPose = leaderPoseSubscriber.LatestMessage;
        followerPose = followerPoseSubscriber.LatestMessage;
        if isempty(followerPose)
            disp('Follower pose is not received!');
            followerPose = ros2message('geometry_msgs/PoseStamped');
        end
        if isempty(leaderPose)
            disp('Leader pose is not received!');
            leaderPose = followerPose;
        end

        % Get the leader and follower information
        leaderPosition = [leaderPose.pose.position.x, leaderPose.pose.position.y];
        leaderOrientation = quat2angle([leaderPose.pose.orientation.w,...
                                        leaderPose.pose.orientation.x,...
                                        leaderPose.pose.orientation.y,...
                                        leaderPose.pose.orientation.z]);
        followerPosition = [followerPose.pose.position.x, followerPose.pose.position.y];
        followerOrientation = quat2angle([followerPose.pose.orientation.w,...
                                          followerPose.pose.orientation.x,...
                                          followerPose.pose.orientation.y,...
                                          followerPose.pose.orientation.z]);
        fprintf("Leader Pose: (%.2f, %.2f, %.2f), Follower Pose(%.2f, %.2f, %.2f)\n", ...
            leaderPosition(1), leaderPosition(2), leaderOrientation, ...
            followerPosition(1), followerPosition(2), followerOrientation);

        % Computer the follower control input
        [linvel, angvel] = unicycle_control.unicycle_fwdctrl(followerPosition, followerOrientation, leaderPosition, ...
                           'Tol', 0.5, 'LinGain', 1, 'AngGain', 1);
        [linvel, angvel] = turtlebot_control.turtlebot_control_governor(linvel, angvel);

        % Publish the follower control message
        followerCtrlMessage.linear.x = linvel;
        followerCtrlMessage.angular.z = angvel;
        followerCtrlPublisher.send(followerCtrlMessage);
    end
end
```

Leader-Follower: Initialization

```
21      %% ROS Node Settings
22      % Register the leader-follower node at the ROS master
23      nodeName = ['/matlab_turtlebot_leaderfollower_' int2str(1000000000*rand(1))]; % Node Name
24      leaderfollowerNode = ros2node(nodeName);
25      % Create and register the follower control publisher at the ROS master
26      [followerCtrlPublisher, followerCtrlMessage] = ros2publisher(leaderfollowerNode, followerCtrlTopic, "geometry_msgs/Twist",...
27          'History', 'keeplast', 'Depth', 1, 'Reliability','reliable');
28
29      % Create and register the leader and follower pose subcribers at the ROS master
30      leaderPoseSubscriber = ros2subscriber(leaderfollowerNode, leaderPoseTopic, 'geometry_msgs/PoseStamped',...
31          'History', 'keeplast', 'Depth', 1, 'Reliability','reliable');
32      followerPoseSubscriber = ros2subscriber(leaderfollowerNode, followerPoseTopic, 'geometry_msgs/PoseStamped',...
33          'History', 'keeplast', 'Depth', 1, 'Reliability','reliable');
```

Leader-Follower: Control Loop

```
35      %% Leader-Follower Control Loop
36      disp('Leader-Follower Control is running...');
37      while (true)
38          % Pause for an update rate of at most 10Hz
39          pause(0.1);
40          % Get the lastest information from the ROS network
41          leaderPose = leaderPoseSubscriber.LatestMessage;
42          followerPose = followerPoseSubscriber.LatestMessage;
43          if isempty(followerPose)
44              disp('Follower pose is not received!');
45              followerPose = ros2message('geometry_msgs/PoseStamped');
46          end
47          if isempty(leaderPose)
48              disp('Leader pose is not received!');
49              leaderPose = followerPose;
50          end
51
52          % Get the leader and follower information
53          leaderPosition = [leaderPose.pose.position.x, leaderPose.pose.position.y];
54          leaderOrientation = quat2angle([leaderPose.pose.orientation.w,...
55                                          leaderPose.pose.orientation.x,...
56                                          leaderPose.pose.orientation.y,...
57                                          leaderPose.pose.orientation.z]);
58          followerPosition = [followerPose.pose.position.x, followerPose.pose.position.y];
59          followerOrientation = quat2angle([followerPose.pose.orientation.w,...
60                                            followerPose.pose.orientation.x,...
61                                            followerPose.pose.orientation.y,...
62                                            followerPose.pose.orientation.z]);
63          fprintf("Leader Pose: (%.2f, %.2f, %.2f), Follower Pose(%.2f, %.2f, %.2f)\n", ...
64              leaderPosition(1), leaderPosition(2), leaderOrientation, ...
65              followerPosition(1), followerPosition(2), followerOrientation);
66
67          % Computer the follower control input
68          [linvel, angvel] = unicycle_control.unicycle_fwdctrl(followerPosition, followerOrientation, leaderPosition, ...
69                      'Tol', 0.5, 'LinGain', 1, 'AngGain', 1);
70          [linvel, angvel] = turtlebot_control.turtlebot_control_governor(linvel, angvel);
71
72          % Publish the follower control message
73          followerCtrlMessage.linear.x = linvel;
74          followerCtrlMessage.angular.z = angvel;
75          followerCtrlPublisher.send(followerCtrlMessage);
76      end
```


TU/e

Use **demo_ros2_topic_list()** to find the correct pose and control topics

Make sure Gazebo running on the simulation computer:
$ ros2 launch core_tue4dm70_turtlebot3_simulate team1_threeturtlebot3goal_gazebo_empty.launch

Network: **CoreTurtlebotWiFi**
Password: **core_turtlebot-123!**

Network: **CoreMinicarWiFi**
Password: **core_minicar-123!**

TurtleBot Leader-Follower Coordination

- Run **help demo_turtlebot_leaderfollower**

```
>> help demo_turtlebot_leaderfollower

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLEASE MAKE SURE YOUR ROS SETTINGS ARE CORRECT! %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID)
   leaderPoseTopic = '/mocap/turtlebot1/pose';
   followerPoseTopic = '/mocap/turtlebot2/pose';
   followerCtrlTopic = '/turtlebot2/cmd_vel_ctrl';
   demo_turtlebot_leaderfollower(leaderPoseTopic, followerPoseTopic, followerCtrlTopic);
```

- Set pose & control topic names

```
>> leaderPoseTopic = '/mocap/turtlebot1/pose';
>> followerPoseTopic = '/mocap/turtlebot2/pose';
>> followerCtrlTopic = '/turtlebot2/cmd_vel_ctrl';
```

- Set **ROS_DOMAIN_ID**

```
>> setenv("ROS_DOMAIN_ID", ???); % SIMULATION COMPUTER ID
```

- Stop by **Ctrl-C**

- Run **demo_turtlebot_leaderfollower**

```
>> demo_turtlebot_leaderfollower(leaderPoseTopic, followerPoseTopic, followerCtrlTopic);
Leader-Follower Control is running...
Follower pose is not received!
Leader pose is not received!
Leader Pose: (-4.00, -0.17, 0.94), Follower Pose(-4.12, -0.60, 1.05)
Leader Pose: (-4.00, -0.17, 0.94), Follower Pose(-4.12, -0.60, 1.05)
```

Part III – Recording & Replaying ROS Topics

Record/Replay ROS Topics via Bag Files

- Run **help demo_ros2bag_record**

```
>> help demo_ros2bag_record

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLEASE MAKE SURE YOUR ROS SETTINGS ARE CORRECT! %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID)
   bagTopics = {"/mocap/turtlebot_red/pose", "/turtlebot_red/cmd_vel"};
   bagPath = fullfile(getenv("HOME"), "Documents", "tmp", "demo_ros2bag");
   demo_ros2bag_record(bagPath, bagTopics)
```

- Run **help demo_ros2bag_replay**

```
>> help demo_ros2bag_replay

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLEASE MAKE SURE YOUR ROS SETTINGS ARE CORRECT! %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID)
   bagTopics = {"/mocap/turtlebot_red/pose", "/turtlebot_red/cmd_vel"};
   bagPath = fullfile(getenv("HOME"), "Documents", "tmp", "demo_ros2bag");
   demo_ros2bag_replay(bagPath, bagTopics)
```

Part IV – MATLAB-ROS TurtleBot Simulator

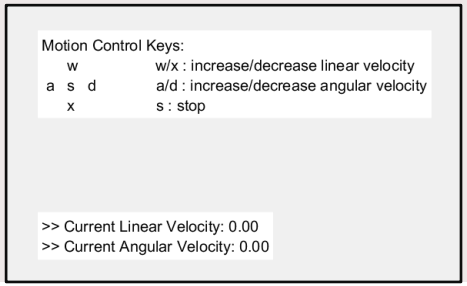
ROS-based Multiple TurtleBot Simulator in MATLAB

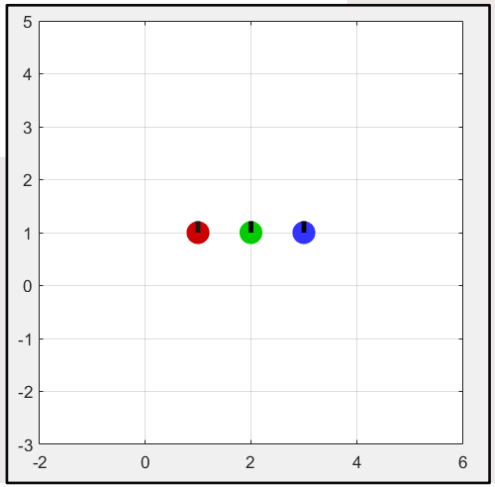
- Run **help demo_threeturtlebot_simulator**

```
>> help demo_threeturtlebot_simulator

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLEASE MAKE SURE YOUR ROS SETTINGS ARE CORRECT! %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID);
   demo_threeturtlebot_simulator()
```





- Run **help demo_ros2_topic_list**

```
>> help demo_ros2_topic_list
Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID);
   ros2 topic list
```

```
>> ros2 topic list
/mocap/turtlebot_blue/pose
/mocap/turtlebot_green/pose
/mocap/turtlebot_red/pose
/parameter_events
/rosout
/turtlebot_blue/cmd_vel_ctrl
/turtlebot_green/cmd_vel_ctrl
/turtlebot_red/cmd_vel_ctrl
```

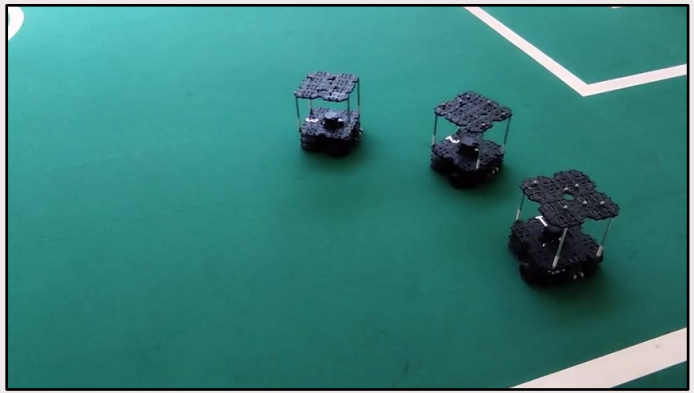
- Run **help demo_turtlebot_teleop**

```
>> help demo_ros2_topic_list
Example:
   ROS_DOMAIN_ID = "0";
   setenv("ROS_DOMAIN_ID", ROS_DOMAIN_ID);
   turtlebotName = 'turtlebot_green';
   ctrlTopic = sprintf('/%s/cmd_vel_ctrl', turtlebotName);
   demo_turtlebot_teleop(ctrlTopic);
```

Part V – TurtleBot Challenge

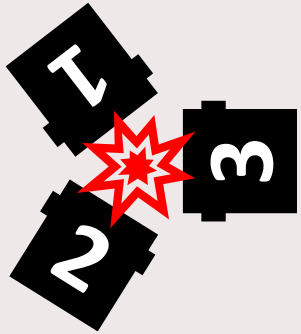
Safe Leader-Follower with Three Turtlebots

- First Leader-Follower Pair:
  Leader: TurtleBot1, Follower: TurtleBot2
- Second Leader-Follower Pair
  Leader: TurtleBot2, Follower: Turtlebot3
- Make sure robots don't collide with each other.



Follow the Leader!

Avoid Collisions!



TU/e


Thanks for your attention!