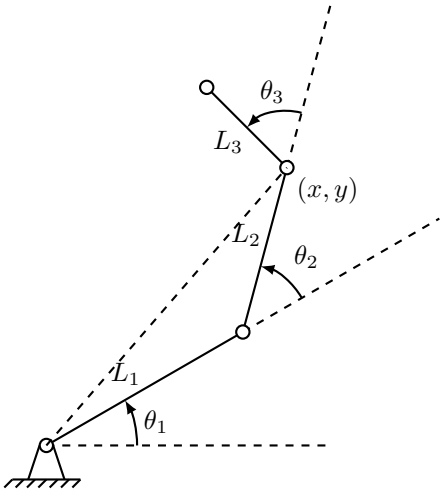
Inverse kinematica

De opdracht van vandaag is het maken van een node die de benodigede robot pose berekent en vervolgens publiseert zodat we het resultaat zichtbaar kunnen maken in Rviz. Voordat we van start gaan volgt nog een korte samenvatting van de manier waarop we dit kunnen berekenen. Hierbij concentreren we ons eerst op het geval van drie in serie geschakelde links die geschakeld zijn met joints die kunnen roteren. Dit is in lijn met de Parol6 robot als de vrijheidsgraad in de onderarm niet geroteerd is ten opzichte van de elleboog. In dat geval is de situatie zoals aangegeven in Figuur 1



Figuur 1: Overzicht robot met drie rotatie-joints.

Starten we met (x, y) dan volgt

$$x = L_1 \cos\theta_1 + L_2 \cos\theta_{12}$$
$$y = L_1 \sin\theta_1 + L_2 \sin\theta_{12}$$

waar $\theta_{12} = \theta_1 + \theta_2$. Kwadrateren van x en y levert

$$x^2 = (L_1 \cos\theta_1 + L_2 \cos\theta_{12})^2 = L_1^2 \cos^2\theta_1 + 2L_1L_2 \cos\theta_1 \cos\theta_{12} + L_2^2 \cos^2\theta_{12}$$
$$y^2 = (L_1 \sin\theta_1 + L_2 \sin\theta_{12})^2 = L_1^2 \sin^2\theta_1 + 2L_1L_2 \sin\theta_1 \sin\theta_{12} + L_2^2 \sin^2\theta_{12}$$

waarbij de kruistermen kunnen worden vereenvoudigd door het optellen van x^2 en y^2 omdat

$$\cos\theta_1 \cos\theta_{12} = \cos^2\theta_1 \cos\theta_2 - \cos\theta_1 \sin\theta_1 \sin\theta_2$$
$$\sin\theta_1 \sin\theta_{12} = \sin\theta_1 \cos\theta_1 \sin\theta_2 + \sin^2\theta_1 \cos\theta_2$$

optellen van deze twee termen levert

$$\cos\theta_1 \cos\theta_{12} + \sin\theta_1 \sin\theta_{12} = \cos\theta_2$$

waarna volgt dat

$$\cos\theta_1 \cos\theta_{12} + \sin\theta_1 \sin\theta_{12} = \cos^2\theta_1 \cos\theta_2 + \sin^2\theta_1 \cos\theta_2 = \cos\theta_2$$

waarin $\cos^2\theta_1 + \sin^2\theta_1 = 1$ zodat uiteindelijk

$$x^2 + y^2 = L_1^2 + L_2^2 + 2L_1L_2\cos\theta_2$$

herschrijven levert

$$\cos\theta_2 = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}.$$

We kunnen nu de conclusie trekken dat

$$-1 \leq \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \leq 1$$

omdat de functie $-1 \leq \cos\theta_2 \leq 1\ \forall\theta_2$. Uit $\cos\theta_2$ kan ook $\sin\theta_2$ bepaald worden door

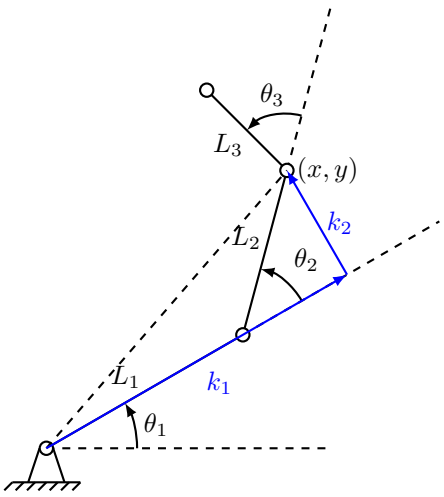
$$\sin\theta_2 = \sqrt{1 - \cos^2\theta_2}$$

zodat vervolgens volgt dat

$$\theta_2 = \arctan 2\,(\sin\theta_2, \cos\theta_2)$$

Merk op dat de functie $\arctan 2$ niet het zelfde is als arctan aangezien we bij het bepalen van een hoek onderscheid willen kunnen maken tuseen het eerste en het derde kwadrant of het tweede en het vierde kwadrant. Door het delen van twee getallen zou dit onderscheid verloren knnen gaan. De volgende stap is het definiëren van twee nieuwe variabelen k_1 en k_2 zie Figuur 2. Het volgt nu dat

$$k_1 = L_1 + L_2\cos\theta_2$$
$$k_2 = L_2\sin\theta_2$$



Figuur 2: Overzicht robot met drie rotatie-joints.

Met k_1 en k_2 kunnen we de straal r bereken en de hoek γ zodat

$$r = \sqrt{k_1^2 + k_2^2}$$
$$\gamma = \text{Atan}\,2\,(k_2, k_1)$$

Het volgt dat

$$k_1 = r\cos\gamma$$
$$k_2 = r\sin\gamma$$

en dus dat

$$x = k_1\cos\theta_1 - k_2\sin\theta_1$$
$$y = k_1\sin\theta_1 + k_2\cos\theta_1$$

invullen levert

$$\frac{x}{r} = \cos\gamma\cos\theta_1 - \sin\gamma\sin\theta_1$$
$$\frac{y}{r} = \cos\gamma\sin\theta_1 + \sin\gamma\cos\theta_1$$

waarna met standaard goniometrische formules volgt dat

$$\cos(\gamma + \theta_1) = \frac{x}{r}$$
$$\sin(\gamma + \theta_1) = \frac{y}{r}$$

en θ_1 gereconstrueerd kan worden met

$$\gamma + \theta_1 = \text{Atan}\,2\left(\frac{y}{r}, \frac{x}{r}\right) = \text{Atan}\,2(y, x)$$

en

$$\theta_1 = \arctan 2(y, x) - \arctan 2(k_2, k_1)$$

Tot zo ver de standaard afleiding (zie ook John J Craig, Introduction to Robotics, Mechanics and Control). Om de hoeken geschikt te maken voor de robot moeten we de hoeken aanpassen volgens

$$\theta_0 = \theta_0$$
$$\theta_1 = \theta_1 - \tfrac{1}{2}\pi$$
$$\theta_2 = \theta_2$$
$$\theta_3 = -\theta_1 - \theta_2 - \tfrac{1}{2}\pi$$
$$\theta_4 = \theta_0$$
$$\theta_6 = 0.0091$$

Merk op dat deze waarden geschikt zijn voor de robot die gebruikt wordt om ROS2 te demonstreren en ze nog moeten worden vertaald naar de Parol6 robot die gebruikt wordt om deze opdracht op te baseren.

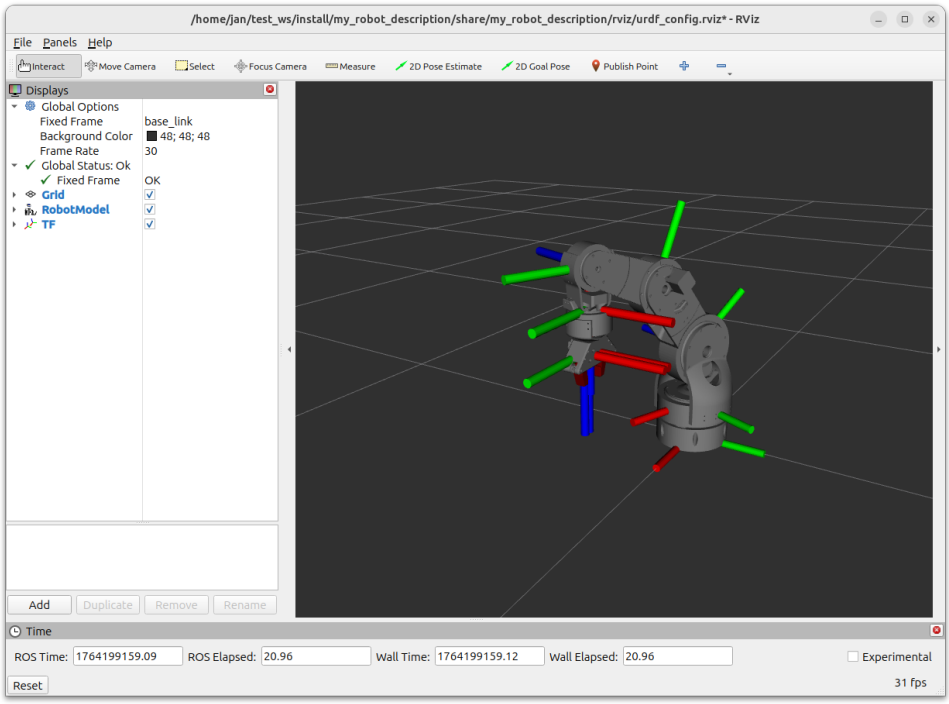
1 Joint state publisher node

Begin met het aanmaken van een package met een node op basis van c++. In dit geval heet het package my_robot_ik_node met daarin in de src directory de ik_node.cpp. De package.xml en de CMakeLists.txt zijn hieronder gegeven. In het volgend hoofdstuk is een template gegeven voor het publishen van een pose naar het JointState topic. Gebruik dit template als basis voor je inverse kinematica node. Om het te testen gebruiken we een aangepaste launch file zodat de

```
<launch>
    <let name="urdf_path" value="$(find-pkg-share my_robot_description)/urdf/
        skyentific_robot.urdf.xacro"/>
    <let name="rviz_config_path" value="$(find-pkg-share my_robot_description)/
        rviz/urdf_config.rviz"/>

    <node pkg="robot_state_publisher" exec="robot_state_publisher">
        <param name="robot_description" value="$(command 'xacro $(var urdf_path)
            ')" />
    </node>
    <!--<node pkg="joint_state_publisher_gui" exec="joint_state_publisher_gui" />
        -->
    <node pkg="rviz2" exec="rviz2" output="screen" args="-d $(var
        rviz_config_path)" />
</launch>
```

Als we de joint_state_publisher_gui niet starten dan zal rviz een foutmelding geven die wordt veroorzaakt omdat er niets gepubliceerd is en robot niet geïnitialiseerd wordt. Zodra we ik_node starten met ros2 run zal dit veranderen en zal de robot een pose aannemen. vanaf de commandline starten we de node met het onderstaande commando. Het resulaat is te zien in Figuur 3.

```
ros2 run my_robot_ik_node ik_node --ros-args -p "theta_1":=-
    0.3805 -p "theta_2":=-0.6531 -p "theta_3":=-0.5451 -p "
    theta_4":=-1.9434 -p "theta_5":=-0.3805 -p "theta_6":=
    0.0091
```



Figuur 3: Rviz laat het resultaat zien van het opstarten van de ik_node.

Een simpele c++ publisher node

Bijgevoegd is een simpele publiser node die we kunnen gebruiken als basis voor de te ontwikkelen inverse kinematica node.

```
#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/joint_state.hpp>
#include <tf2_ros/transform_listener.h>
#include <tf2_ros/buffer.h>
#include <geometry_msgs/msg/transform_stamped.hpp>

class JointPublisherNode : public rclcpp::Node
{
public:
    JointPublisherNode() : Node("ik_node")
    {
        //Declare and get parameters
        this->declare_parameter<double>("theta_1", 0.0);
        this->declare_parameter<double>("theta_2", 0.0);
        this->declare_parameter<double>("theta_3", 0.0);
        this->declare_parameter<double>("theta_4", 0.0);
        this->declare_parameter<double>("theta_5", 0.0);
        this->declare_parameter<double>("theta_6", 0.0);


        theta_1_ = this->get_parameter("theta_1").as_double();
        theta_2_ = this->get_parameter("theta_2").as_double();
        theta_3_ = this->get_parameter("theta_3").as_double();
        theta_4_ = this->get_parameter("theta_4").as_double();
        theta_5_ = this->get_parameter("theta_5").as_double();
        theta_6_ = this->get_parameter("theta_6").as_double();

        // Initialize the joint state publisher
        joint_state_publisher_ = this->create_publisher<sensor_msgs::msg::
            JointState>("joint_states", 10);

        // Initialize the transform buffer and listener
        tf_buffer_ = std::make_shared<tf2_ros::Buffer>(this->get_clock());
        tf_listener_ = std::make_shared<tf2_ros::TransformListener>(*tf_buffer_);

        RCLCPP_INFO(this->get_logger(), "Joint states (%f %f %f %f %f %f) were
            sent to the system."
                                        ,theta_1_, theta_2_, theta_3_, theta_4_,
                                            theta_5_, theta_6_);

        // Set up a timer to periodically check for the transform and broadcast it
        timer_ = this->create_wall_timer(
            std::chrono::milliseconds(100),
            std::bind(&JointPublisherNode::broadcastJointState, this));
    }

private:
    void broadcastJointState()
    {
        // Create a JointState message
        auto joint_state = sensor_msgs::msg::JointState();
        joint_state.header.stamp = this->get_clock()->now();
        joint_state.name = {"axis_1", "axis_2", "axis_3", "axis_4", "axis_5", "
            axis_6"};
        joint_state.position = {theta_1_, theta_2_, theta_3_, theta_4_, theta_5_,
             theta_6_};

        // Publish the joint state
```

```
        joint_state_publisher_->publish(joint_state);

    }

    double theta_1_;
    double theta_2_;
    double theta_3_;
    double theta_4_;
    double theta_5_;
    double theta_6_;
    rclcpp::Publisher<sensor_msgs::msg::JointState>::SharedPtr
         joint_state_publisher_;
    std::shared_ptr<tf2_ros::Buffer> tf_buffer_;
    std::shared_ptr<tf2_ros::TransformListener> tf_listener_;
    rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char **argv)
{

    rclcpp::init(argc, argv);
    auto node = std::make_shared<JointPublisherNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

Merk op dat de node actief blijft nadat de message op het topic is gepublished en dat we deze moeten afsluiten met CTRL+C om een nieuwe waarde te kunnen publiceren.

Code package.xml en CMakeLists.txt

Hieronder zijn de gebruikte package.xml en CMakeLists.txt gegeven. Pas de code aan om deze geschikt te maken voor je eigen situtatie.

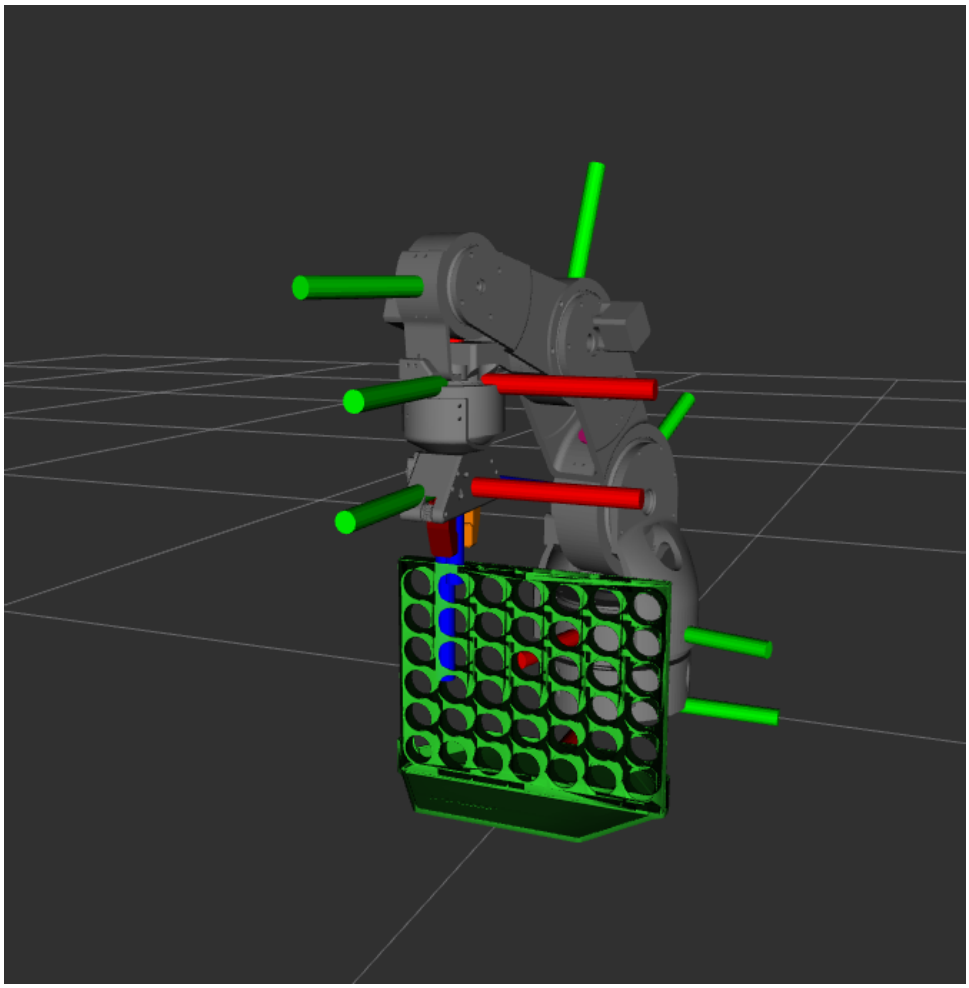
```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
     schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_robot_ik_node</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="student@todo.todo">jan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

  <buildtool_depend>ament_cmake</buildtool_depend>

  <depend>rclcpp</depend>
  <depend>tf2_ros</depend>
  <depend>geometry_msgs</depend>
  <depend>sensor_msgs</depend>

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

```
cmake_minimum_required(VERSION 3.8)
project(my_robot_ik_node)
```

```
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(tf2_ros REQUIRED)
find_package(geometry_msgs REQUIRED)
find_package(sensor_msgs REQUIRED)

# set dependencies
set(dependencies
rclcpp
tf2_ros
geometry_msgs
sensor_msgs
)

# add executables for scripts
add_executable(ik_node src/ik_node.cpp)
ament_target_dependencies(ik_node ${dependencies})

# install the scripts
install(TARGETS
  ik_node
  DESTINATION lib/${PROJECT_NAME})

install(
  DIRECTORY launch
  DESTINATION share/${PROJECT_NAME}
)

ament_package()
```

Het assignment voor vandaag

Het assignment van vandaag bestaat uit het vertalen van de wiskunde naar een private method zodat we in plaats van de hoeken een case nummer kunnen opgeven bij het starten van de node. Het is de bedoeling dat de basis van de berekening cartesische coordinaten zijn die vervolgens worden omgerekend naar hoeken die we publishen op het joint state topic. Je zal hiervoor de URDF van de Parol6 robot moeten bestuderen. Het resultaat moet je kunnen demonstreren binnen Rviz.



Figuur 4: Rviz laat het resultaat zien van het opstarten van de ik_node.