Implementation of GPS in competition environment

July 20, 2022

Aim:

To add a GPS to the environment so that the co-ordinates of any robot can be obtained by echoing a ROS2 topic.

NavSat:

While navigation in a static map can be accomplished by using AMCL, here in a simulation of a dynamic environment which is not mapped out, we use a pre-existing sensor called NavSat to obtain the latitude and longitude of the robot.

NavSat is a sensor provided by ROS2 and Ignition Gazebo, which allows us to obtain the co-ordinates of any object in the given world. NavSat fix provides the Navigation Satellite fix for any Global Navigation Satellite System and is specified using the World Geodetic System (WGS84) reference ellipsoid. The sensor message consists of the following:

1. Satellite fix status information.
2. Latitude in degrees. Positive is north of equator; negative is south.
3. Longitude in degrees. Positive is east of prime meridian; negative is west.
4. Altitude in meters. Positive is above the WGS 84 ellipsoid
5. Position covariance [m^2] defined relative to a tangential plane through the reported position. The components are East, North, and Up (ENU), in row-major order.

The NavSat sensor provides a convenient inbuilt way to keep track of the location of any robot and vessel in the competition environment in an absolute sense

Implementation:

Adding the inbuilt NavSat plugin to the ros_ign bridge and altering the the sdformat files of the demo world and the robot model, in this case the quadrotor, to allow for echoing ROS2 topic '/gps/fix' to get the latitude, longitude and altitude of the robot.

Change log:

Github repository containing the files for the environment with changes that allow for the use of the NavSat sensor

1. ros_ign_bridge/include/ros_ign_bridge/convert/sensor_msgs.hpp:

Header files for both NavSatFix for ROS2 and NavSat for Ignition Transport, as well as declaration of functions to convert ros to ign and vice versa.

```
#include <sensor_msgs/msg/nav_sat_fix.hpp>
// Ignition messages
#include <ignition/msgs/navsat.pb.h>
template<>
void
convert_ros_to_ign(
   const sensor_msgs::msg::NavSatFix & ros_msg,
   ignition::msgs::NavSat & ign_msg);

template<>
void
convert_ign_to_ros(
   const ignition::msgs::NavSat & ign_msg,
   sensor_msgs::msg::NavSatFix & ros_msg);
```

2. ros_ign_bridge/ros_ign_bridge/mappings.py :

```
Mapping('NavSatFix', 'NavSat'),
```

Include a mapping between the ROS2 and Ignition versions in the sensor_msgs list.

3. ros_ign_bridge/src/convert/sensor_msgs.cpp:

Implementation of functions for both ros to ign and vice versa, including latitude, longitude and altitude.

```
template<>
void
convert_ros_to_ign(
   const sensor_msgs::msg::NavSatFix & ros_msg,
   ignition::msgs::NavSat & ign_msg)
{
   convert_ros_to_ign(ros_msg.header, (*ign_msg.mutable_header()));
   ign_msg.set_latitude_deg(ros_msg.latitude);
   ign_msg.set_longitude_deg(ros_msg.longitude);
```

```
   ign_msg.set_altitude(ros_msg.altitude);
   ign_msg.set_frame_id(ros_msg.header.frame_id);

   // Not supported in sensor_msgs::NavSatFix.
   ign_msg.set_velocity_east(0.0);
   ign_msg.set_velocity_north(0.0);
   ign_msg.set_velocity_up(0.0);
}

template<>
void
convert_ign_to_ros(
   const ignition::msgs::NavSat & ign_msg,
   sensor_msgs::msg::NavSatFix & ros_msg)
{
   convert_ign_to_ros(ign_msg.header(), ros_msg.header);
   ros_msg.header.frame_id = frame_id_ign_to_ros(ign_msg.frame_id());
   ros_msg.latitude = ign_msg.latitude_deg();
   ros_msg.longitude = ign_msg.longitude_deg();
   ros_msg.altitude = ign_msg.altitude();

   // position_covariance is not supported in Ignition::Msgs::NavSat.
   ros_msg.position_covariance_type = sensor_msgs::msg::NavSatFix::COVARIANCE_TYPE_UNKNOWN;
   ros_msg.status.status = sensor_msgs::msg::NavSatStatus::STATUS_FIX;
}
```

4. ros_ign_bridge/test/utils/ign_test_msg.cpp :
   Test message for Ignition to make sure the syntax of the actual message matches the expected syntax of the created test message.

```
void createTestMsg(sensor_msgs::msg::NavSatFix & _msg)
{
   std_msgs::msg::Header header_msg;
   createTestMsg(header_msg);

   _msg.header = header_msg;
   _msg.status.status = sensor_msgs::msg::NavSatStatus::STATUS_FIX;
   _msg.latitude = 0.00;
   _msg.longitude = 0.00;
   _msg.altitude = 0.00;
   _msg.position_covariance = {1, 2, 3, 4, 5, 6, 7, 8, 9};
   _msg.position_covariance_type = sensor_msgs::msg::NavSatFix::COVARIANCE_TYPE_UNKNOWN;
}

void compareTestMsg(const std::shared_ptr<sensor_msgs::msg::NavSatFix> & _msg)
```

```
{
   sensor_msgs::msg::NavSatFix expected_msg;
   createTestMsg(expected_msg);

   compareTestMsg(_msg->header);
   EXPECT_EQ(expected_msg.status, _msg->status);
   EXPECT_FLOAT_EQ(expected_msg.latitude, _msg->latitude);
   EXPECT_FLOAT_EQ(expected_msg.longitude, _msg->longitude);
   EXPECT_FLOAT_EQ(expected_msg.altitude, _msg->altitude);
   EXPECT_EQ(expected_msg.position_covariance_type, _msg->position_covariance_type);
  for (auto i = 0u; i < 9; ++i) {
    EXPECT_FLOAT_EQ(0, _msg->position_covariance[i]);
  }
}
```

5. ros_ign_bridge/test/utils/ros_test_msg.hpp:
   Test message for ROS2 to make sure the syntax of the actual message matches the expected syntax of the created test message.

```
#include <sensor_msgs/msg/nav_sat_fix.hpp>

/// \brief Create a message used for testing.
/// \param[out] _msg The message populated.
void createTestMsg(sensor_msgs::msg::NavSatFix & _msg);

/// \brief Compare a message with the populated for testing.
/// \param[in] _msg The message to compare.
void compareTestMsg(const std::shared_ptr<sensor_msgs::msg::NavSatFix> & _msg);
```

6. mbzirc/mbzirc_ign/worlds/simple_demo.sdf:
   Addition of the NavSat system plugin to allow for the use of GPS in the world, as well as the definiton of the spherical co-ordinates to set the latitude, longitude and altitude of the world using the WGS84 reference ellipsoid.

```
<plugin
      filename="ignition-gazebo-navsat-system"
      name="ignition::gazebo::systems::NavSat">
</plugin>
<spherical_coordinates>
      <surface_model>EARTH_WGS84</surface_model>
               <world_frame_orientation>ENU</world_frame_orientation>
      <latitude_deg>-22.986687</latitude_deg>
      <longitude_deg>-43.202501</longitude_deg>
```

```
      <elevation>0</elevation>
      <heading_deg>0</heading_deg>
</spherical_coordinates>

Note: xml version = 1.0, sdf version = 1.9
```

7. mbzirc/mbzirc_ign/models/mbzirc_quadrotor_base/model.sdf:
   Adding the NavSat sensor to the robot model to allow for obtaining its co-ordinates using the GPS.

```
<sensor name='navsat' type='navsat'>
        <always_on>1</always_on>
        <update_rate>1</update_rate>
 </sensor>
```

8. mbzirc/mbzirc_ign/src/mbzirc_ign/bridges.py:
   Function to add the NavSat sensor bridge, including the IGN and ROS topics. The bridge created is from IGN to ROS.

```
def navsat(world_name, model_name, link_name='base_link'):
    sensor_prefix = prefix(world_name, model_name, link_name)
    return Bridge(
        ign_topic=f'{sensor_prefix}/navsat/navsat',
        ros_topic='gps/fix',
        ign_type='ignition.msgs.NavSat',
        ros_type='sensor_msgs/msg/NavSatFix',
        direction=BridgeDirection.IGN_TO_ROS)
```

9. mbzirc/mbzirc_ign/src/mbzirc_ign/test_bridges.py:
   Test function for the NavSat bridge function in bridges.py. Note that /navsat/navsat is the IGN topic and sensor_msgs/msg/NavSatFix is the ROS sensor message while ignition.msgs.NavSat is the IGN sensor message. We use the IGN topic since the bridge is IGN to ROS.

```
def test_navsat(self):
        bridge = bridges.navsat(self.world_name, self.model_name, self.link_name)
        self.assertEqual(bridge.argument(),
                         f'{self.prefix()}/navsat/navsat'
                '@sensor_msgs/msg/NavSatFix[ignition.msgs.NavSat')
```

10. mbzirc/mbzirc_ign/src/mbzirc_ign/model.py:

```
# NavSat
            mbzirc_ign.bridges.navsat(world_name, self.model_name)
```

Here we include the NavSat function from bridges.py in the list of bridges for the model. It has been defined for all models, not just for the UAV to facilitate usage with any other model.

Working:

Video of output when echoing the gps/fix topic of moving quadrotor