9/2/2-25

Edited /home/con/rock/base/types/src/samples/RigidBodyState.hpp

to overload the << operator so you can send the rigid body state data

```
231         std::string printRBSHeaders() const;
232         inline friend std::ostream& operator<<(std::ostream& os, const RigidBodyState rbs)
233     {
234         static const Eigen::IOFormat OneLine(Eigen::StreamPrecision,
235                                              Eigen::DontAlignCols,
236                                              " ", " ",  // coeffSeparator, rowSeparator
237                                              "", "", "", "");
238 
239         os <<
240         //rbs.time.toString() <<
241         rbs.time <<
242         rbs.sourceFrame <<
243         rbs.targetFrame <<
244         rbs.position.transpose() <<
245         rbs.cov_position.format(OneLine) <<
246         rbs.orientation.coeffs().transpose() <<
247         rbs.cov_orientation.format(OneLine) <<
248         rbs.velocity.transpose() <<
249         rbs.cov_velocity.format(OneLine) <<
250         rbs.angular_velocity.transpose() <<
251         rbs.cov_angular_velocity.format(OneLine);
252 
253         return os;
254     }
```

Eigen::IOFormat

/home/con/rock/base/types/src/samples/RigidBodyState.cpp

```cpp
std::string RigidBodyState::printRBSHeaders() const
{
    std::stringstream ss;
    ss <<
    "Time " <<
    "Source Frame " <<
    "Target Frame: " <<
    "Position: " <<
    "Covar Position " <<
    "Orientation " <<
    "Covar Orientation " <<
    "Velocity " <<
    "Covar Velocity " <<
    "Angular Velocity " <<
    "Covar Ang Velocity";
    return ss.str();
}
```

/home/con/rock/slam/orogen/uwv_kalman_filters/tasks/PoseEstimator.hpp

```cpp

#include <boost/asio.hpp>

```

```cpp
    protected:
        boost::asio::io_service io_service_;
        boost::asio::ip::udp::socket socket_;
        boost::asio::ip::udp::endpoint multicast_endpoint_;

        boost::shared_ptr<uwv_kalman_filters::PoseUKF> pose_filter;
        boost::shared_ptr<pose_estimation::StreamAlignmentVerifier> verifier;
```

/home/con/rock/slam/orogen/uwv_kalman_filters/tasks/PoseEstimator.cpp

```cpp
PoseEstimator::PoseEstimator(std::string const &name)
    : PoseEstimatorBase(name)
    , socket_(io_service_)
{
}

PoseEstimator::PoseEstimator(std::string const &name, RTT::ExecutionEngine *engine)
    : PoseEstimatorBase(name, engine)
    , socket_(io_service_)
{
}
```

```cpp
//con: 8/29/25
bool has_valid_poses = false;
for (const auto& pose : marker_poses_stamped_samples.marker_poses)
{
    if (pose.hasValidPosition() && pose.hasValidOrientation())
    {
        has_valid_poses = true;
        break;
    }
}
if (!has_valid_poses) {
    return;
}
```

```cpp
    if (!initialized)
        return;
    try {
        // Create measurement using filter's existing types
        PoseUKF::XY_Position xy_measurement;
        xy_measurement.mu = imu_in_nwu_final.translation().head<2>();
        xy_measurement.cov = Eigen::Matrix2d::Identity() * 0.1;

        // Integrate position measurement
        pose_filter->integrateMeasurement(xy_measurement);
    }
    catch (const std::runtime_error &e)
    {
        LOG_ERROR_S << "Failed to integrate marker pose measurement: " << e.what();
    }
```