Basic Toolkit Package
by SMSL

Version: 1.33

1. About this document

This is the usage documentation of the package of RTMaps by Suzuka Management Science Laboratory (SMSL).
Package Name: SMSL_BasicToolkit.pck

Author: IDA Shogo / the director of SMSL

Contact email: support@suzuka-msl.jp

2. Disclaimer

This documentation and the associated RTMaps package are provided "as is" without any warranties, express or implied. While every effort has been made to ensure the accuracy and functionality of the components included in this package, the author and distributor assume no liability for any direct, indirect, incidental, or consequential damages arising from the use or misuse of the software.
This software is designed to operate within the RTMaps (Real-Time Multisensor applications) environment, and it may rely on specific hardware, drivers, or third-party libraries. It is the responsibility of the user to ensure system compatibility and perform adequate testing before deploying this software in any safety-critical or production system.
Users are advised that performance may vary depending on the underlying system configuration and that use in real-time systems requires appropriate validation and certification by the deploying organization.
This package may include or interface with third-party components. Users must ensure that they comply with the license terms of any third-party software involved.
The open source software included in this package (GeographicLib, YDLidar-SDK) and its licenses are listed in LICENSE.txt in the Release folder of the package.

This publication and its contents are subject to change without notice.

RTMaps is a registered trademark of Intempora S.A.S.
All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Table of contents

1. About this document 2
2. Disclaimer 2
3. SMSL_BasicToolkit Package 5
3.1 Package Overview 5
3.2 Generator components 6
SineCosineGenerator 6
GaussianNoiseGenerator 8
StaticImageGenerator 10
3.3 GNSS components 12
GeoDistance 12
NMEA_GNS_Decoder 14
NMEA_RMC_Decoder 16
3.4 Sensor Driver components 18
YDLidarX2_Driver 18
WitMotion_WT901B 21
Hi_Link_LD2410C 23
3.5 Signal Processing components 26
UnitDelay 26
SkipEarlyFrames 29
SampleHistoryVectorizer 31
MaxMinMeanforArray 33
MaxMinMeanforMultipleSignals 34
NumericalTextWriter 36
bits2Byte 38
Byte2bits 40

imu_to_pose6D 41
imu_orientation_estimator 43
3.6 Hardware Specification components 46
RobotSpecManager 46
4. Macro Component 49
SineCosineGenMacro 49
GaussianNoiseGenMacro 51
PresenceAlarm 53
5. Demo Diagrams 55
DSB_AM_demo 55
MaxMinMean_demo 56
GeoDistance_demo 57
YDLidarDataAcquisition_demo 58
WitMotion_WT901B_imu_demo 58
Hi_Link_LD2410C_demo 59
6. Conclusion 61
7. Revision History 61

3. SMSL_BasicToolkit Package

3.1 Package Overview

This package includes basic application components by SMSL which work on RTMaps4. SMSL provides comprehensive signal generators, GNSS data processing functions and some help functions in this package.

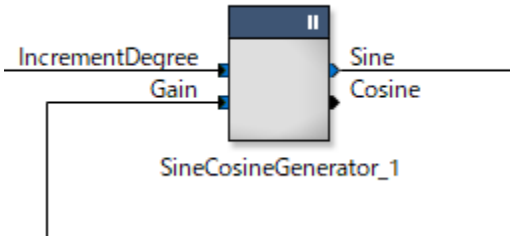
The package is evaluated on the following OS versions:

- Windows11 64bit version.
- Ubuntu 24.04 LTS 64bit version

3.2 Generator components

These generator components are useful to validate the Kalman filter components or create phase signal.

SineCosineGenerator



Generates a sine and a cosine wave (90° apart) with variable gain, initial angle and DC offset.

Processing

- Each IncrementDegree sample advances the internal angle: θ += IncrementDegree (degree), starting at InitAngleBias.

- Sine = Gain·sin(θ) + DC_bias, Cosine = Gain·cos(θ) + DC_bias.

Usage

- Typical inputs: number_generator.o_float64 → IncrementDegree; number_generator.o_float64 → Gain; SineCosineGenerator.Sine → Gain.

- Typical outputs: Sine → Oscilloscope.inputA; Sine → SineCosineGenerator.Gain.

Data format

- Inputs: scalars. Outputs: Float64 scalars.

Notes

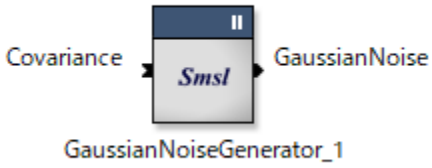
- The output frequency is (IncrementDegree / 360) × trigger rate.

Input Ports	Data type	Description
IncrementDegree	Float64	Angle increment in degree per sample. Each sample triggers one output.
Gain	Float64	Amplitude of the output waves.

Output Ports	Data type	Description
Sine	Float64	Gain·sin(θ) + DC_bias.
Cosine	Float64	Gain·cos(θ) + DC_bias.

Properties	Data type	Description
InitAngleBias	Float64	Initial angle θ in degree. (default: 0.0)
DC_bias	Float64	Offset added to both outputs. (default: 0.0)

GaussianNoiseGenerator



Generates Gaussian (normal) random noise with a given covariance (variance per sample), one sample per input sample.

Processing

- One noise sample is output each time a Covariance sample arrives (Covariance is the triggering input), so the output rate is the rate of the Covariance input.

- Output = sqrt(|Covariance|) * n, where n is a standard normal random number (MT19937 uniform random numbers and the polar method). The variance of each sample is |Covariance|, whatever the sampling rate.

Data format

- Input: scalar (vector size 1; otherwise the component stops with an error). Output: Float64 scalar with the timestamp of the Covariance input.

Notes

- Feed Covariance from a periodic source (e.g. a number generator) at the desired output rate.

- To simulate the measurement noise of a Kalman filter, give Covariance the same value as the filter's measurement noise covariance R.

- The random sequence starts from the same seed at every start, so a run gives the same noise each time, on Windows and Linux alike.

References

- [1] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator," ACM Transactions on Modeling and Computer Simulation, vol. 8, no. 1, pp. 3-30, 1998. Used through std::mt19937 of the C++ standard library. https://doi.org/10.1145/272991.272995 — Used for: uniform random numbers (MT19937).

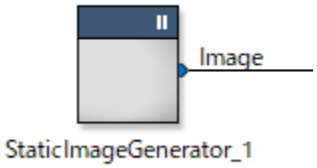
- [2] G. Marsaglia and T. A. Bray, "A Convenient Method for Generating Normal Variables," SIAM Review, vol. 6, no. 3, pp. 260-264, 1964. https://doi.org/10.1137/1006063 — Used for: normal random numbers (polar method).

Input Ports	Data type	Description
Covariance	Float64	Covariance (variance per sample) of the output noise. Each sample triggers one output sample.

Output Ports	Data type	Description
GaussianNoise	Float64	Gaussian noise sample (mean 0, variance |Covariance|).

Properties	Data type	Description
-		

StaticImageGenerator



Outputs a uniform (single-color) grayscale or RGB image, once or periodically, e.g. as a base image for overlays.

Processing

- An image of X_pixels x Y_pixels filled with Brightness (GRAY) or (Red, Green, Blue) (RGB) is output every pPeriod microseconds, or once at the start with output_mode = Once. With Once, later changes of the properties are not output.

Data format

- Output: IplImage, 8-bit GRAY or RGB.

Notes

- Used as the map background for laser scan overlays (e.g. with DrawingObjects).

Input Ports	Data type	Description
–		

Output Ports	Data type	Description
Image	IplImage	Generated image.

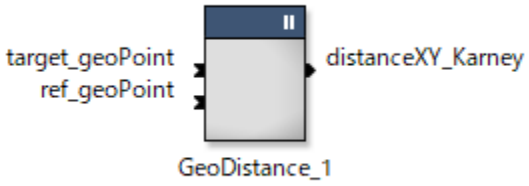
Properties	Data type	Description
X_pixels	Integer	Image width in pixels. (default: 640)
Y_pixels	Integer	Image height in pixels. (default: 480)
output_mode	ENUM	Once: output once at the start. Repeat: output at every pPeriod. Default: Repeat, so that saved diagrams do not change. Options: Once, Repeat. (default: Repeat)
image_format	ENUM	Pixel format of the image. Options: GRAY, RGB. (default: GRAY)
pPeriod	Integer	Output period in microseconds (1 or more). Only with output_mode = Repeat. (default: 100000)
Brightness	Integer	Gray level (0 to 255) in GRAY mode. (default: 0)

Red	Integer	Red level (0 to 255) in RGB mode. (default: 0)
Green	Integer	Green level (0 to 255) in RGB mode. (default: 0)
Blue	Integer	Blue level (0 to 255) in RGB mode. (default: 0)

3.3 GNSS components

These GNSS components are useful to calculate physical signal from GNSS.

GeoDistance



Computes the distance between two geographic points (latitude, longitude) on the WGS84 ellipsoid with Karney's algorithm (GeographicLib), and optionally with the Haversine formula.

Processing

- distanceXY_Karney: geodesic distance from GeographicLib (accurate to nanometers).

- distanceXY_Haversine (optional): great-circle distance on a sphere (faster, less accurate).

Usage

- Typical inputs: UnitDelay.output → target_geoPoint; data_vectorizer.o_data → ref_geoPoint.

- Typical outputs: distanceXY_Karney → SkipEarlyFrames.input_0; distanceXY_Haversine → SkipEarlyFrames.input_1.

Data format

- Inputs: [latitude, longitude] in degree. Outputs: distance in meter.

Notes

- See https://geographiclib.sourceforge.io/.

References

- [1] C. F. F. Karney, "Algorithms for Geodesics," Journal of Geodesy, vol. 87, no. 1, pp. 43-55, 2013. https://doi.org/10.1007/s00190-012-0578-z — Used for: geodesic distance on the WGS84 ellipsoid (distanceXY_Karney).

- [2] C. F. F. Karney, GeographicLib (bundled in 3rdparty/GeographicLib). https://github.com/geographiclib/geographiclib-cpp — Used for: implementation of Karney's algorithm.

- [3] R. W. Sinnott, "Virtues of the Haversine," Sky and Telescope, vol. 68, no. 2, p. 159, 1984. — Used for: Haversine distance (distanceXY_Haversine).

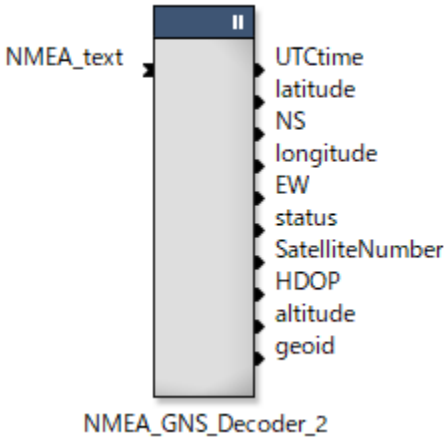
Though the Karney method achieves very precise calculation, the component can output Haversine approximation with the check in the property. You can compare the other calculations with the outputs.

Input Ports	**Data type**	**Description**
target_geoPoint	Float64	Target point [latitude, longitude] in degree.
ref_geoPoint	Float64	Reference point [latitude, longitude] in degree.

Output Ports	**Data type**	**Description**
distanceXY_Karney	Float64	Geodesic distance in meter (Karney).
distanceXY_Haversine	Float64	Great-circle distance in meter (Haversine). Present when Output_Haversine is true.

Properties	**Data type**	**Description**
Output_Haversine	Bool	Adds the Haversine distance output. (default: false)

NMEA_GNS_Decoder



Decodes the GNS sentence of an NMEA 0183 text stream into physical signals (tested with FURUNO PT-G1).

Processing

- Each received GNS sentence is parsed and its fields are output on separate ports with the timestamp of the text sample.

Data format

- Character fields (N/S, E/W, mode) are output as their ASCII codes (e.g. N = 78, S = 83, E = 69, W = 87).

Notes

- Connect the text output of an NMEA serial/GPS component.

References

- [1] National Marine Electronics Association, NMEA 0183 Standard for Interfacing Marine Electronic Devices (GNS and RMC sentences). https://www.nmea.org/ — Used for: GNS sentence format.

Reference in FURUNO's website:

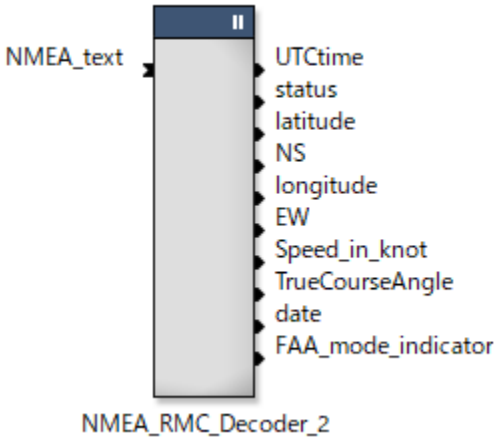
https://www.furuno.com/jp/gnss/datadownload/#PT-G1

Input Ports	Data type	Description
NMEA_text	Text	NMEA 0183 text stream.

Output Ports	Data type	Description
UTCtime	Float64	UTC time (hhmmss.ss).
latitude	Float64	Latitude in decimal degree (converted from NMEA ddmm.mmmm).
NS	Integer32	North/South indicator as ASCII code (78 = N, 83 = S).
longitude	Float64	Longitude in decimal degree (converted from NMEA dddmm.mmmm).
EW	Integer32	East/West indicator as ASCII code (69 = E, 87 = W).
status	Integer32 vector	Mode indicators as ASCII codes (e.g. 65 = A autonomous, 68 = D differential, 78 = N no fix), one per constellation.
SatelliteNumber	Integer32	Number of satellites in use (0 to 24).
HDOP	Float64	Horizontal dilution of precision (0.0 to 50.0).
altitude	Float64	Orthometric height in meter.
geoid	Float64	Geoidal separation in meter.

Properties	Data type	Description
DebugOut	Bool	If true, the received NMEA text is dumped to the console for debugging. (default: false)

NMEA_RMC_Decoder



Decodes the RMC sentence of an NMEA 0183 text stream into physical signals (tested with FURUNO PT-G1).

Processing

- Each received RMC sentence is parsed and its fields are output on separate ports with the timestamp of the text sample.

Usage

- Typical inputs: GPS_NMEA0183.oNMEAMessages → NMEA_text.

- Typical outputs: latitude → data_vectorizer.i_data_1; longitude → data_vectorizer.i_data_2; latitude → QMLViewer.latitude; longitude → QMLViewer.longitude.

Data format

- Character fields are output as ASCII codes (e.g. A = 65, V = 86, N = 78, S = 83, E = 69, W = 87).

References

- [1] National Marine Electronics Association, NMEA 0183 Standard for Interfacing Marine Electronic Devices (GNS and RMC sentences). https://www.nmea.org/ — Used for: RMC sentence format.

Reference in FURUNO's website:

https://www.furuno.com/jp/gnss/datadownload/#PT-G1

Input Ports	Data type	Description
NMEA_text	Text	NMEA 0183 text stream.

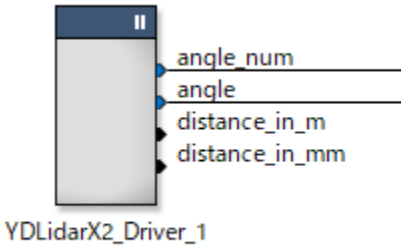
Output Ports	Data type	Description
UTCtime	Float64	UTC time (hhmmss.ss).
status	Integer32	Status as ASCII code (65 = A valid, 86 = V invalid).
latitude	Float64	Latitude in decimal degree (converted from NMEA ddmm.mmmm).
NS	Integer32	North/South indicator as ASCII code (78 = N, 83 = S).
longitude	Float64	Longitude in decimal degree (converted from NMEA dddmm.mmmm).
EW	Integer32	East/West indicator as ASCII code (69 = E, 87 = W).
Speed_in_knot	Float64	Speed over ground in knot.
TrueCourseAngle	Float64	Course over ground in degree (0 to 360).
date	Integer32	Date (ddmmyy).
FAA_mode_indicator	Integer32	FAA mode as ASCII code (65 = A, 68 = D, 69 = E, 78 = N).

Properties	Data type	Description
DebugOut	Bool	If true, the received NMEA text is dumped to the console for debugging. (default: false)

3.4 Sensor Driver components

The driver component for popular and low-cost sensors.

YDLidarX2_Driver



Driver for the YDLIDAR X2 low-cost 2-D LiDAR connected by USB serial; outputs one full scan per revolution.

Processing

- Connects at 115200 baud to DeviceName and decodes the scan packets with the YDLIDAR driver code (MIT License).

- Inverted is passed to the YDLIDAR driver. The X2 measures its angle clockwise; Inverted = true (default) turns it into a counter-clockwise positive angle, which follows the coordinate convention of this toolkit (ROS REP-103: X forward, Y left, angle 0 on +X). Only the angle of each point is changed ($2\pi - \theta$); the points stay in the measurement (time) order, so the angle decreases along the array. Set it to false only when the raw clockwise angle is needed.

- StartAtZeroAngle rotates the array so that it starts at the point closest to angle 0. The points keep the measurement (time) order and are not sorted by value; use a sorting component for that. FixedOutputSize keeps the same number of points per revolution.

Usage

- Typical outputs: angle_num → DataViewer.input_0; angle → DataViewer.input_1;

distance_in_m → laser_scan_to_xyz_points.dist_points.

Data format

- angle (rad or deg, in [-pi, pi)), distance_in_m (Float64) and distance_in_mm (Integer32) have angle_num elements (at most 2048).

- Points out of the measuring range (below 0.05 m or above 64 m) are kept with distance 0. All outputs carry the time at which the scan was received.

Notes

- Tested with X2; other YDLIDAR variants may work at your own risk.

- Version 1.01: AngleSort was renamed StartAtZeroAngle, and Inverted was added with the default true. Version 1.0 output the clockwise angle; set Inverted = false to match data recorded with it.

References

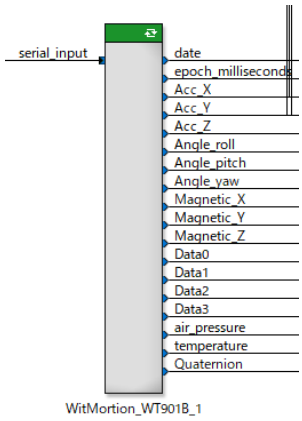
- [1] EAI (YDLIDAR), YDLidar-SDK (bundled in 3rdparty/YDLidar-SDK). https://github.com/YDLIDAR/YDLidar-SDK — Used for: sensor connection and scan acquisition (CYdLidar).

Input Ports	Data type	Description
-		

Output Ports	Data type	Description
angle_num	Integer32	Number of points in the current revolution.
angle	Float64 vector	Angle of each point, in the unit selected by AngleUnit. Counter-clockwise positive when Inverted is true (default).
distance_in_m	Float64 vector	Distance of each point in meter.
distance_in_mm	Integer32 vector	Distance of each point in millimeter.

Properties	Data type	Description
DeviceName	Text	Serial device of the LiDAR (COMx on Windows, /dev/tty* on Linux). (default: COM1)
FixedOutputSize	Bool	If true, every revolution is output with the same number of points. (default: false)
StartAtZeroAngle	Bool	If true, the output starts at the point closest to angle 0 (cyclic shift, measurement order kept); otherwise the driver order is kept. (default: false)
AngleUnit	ENUM	Unit of the angle output. Options: radian, degree. (default: radian)
Inverted	Bool	Passed to the YDLIDAR driver. true (default): counter-clockwise positive angle (ROS REP-103), because the X2 measures clockwise. false: raw clockwise angle of the X2. Either way the points keep the measurement (time) order. (default: true)

WitMotion_WT901B



Decodes the serial data of the WitMotion WT901B IMU (acceleration, angular rate, angles, magnetic field, pressure, temperature, quaternion).

Processing

- The byte stream is split into WitMotion packets and each packet type updates the corresponding outputs.

Usage

- Typical inputs: SerialPort.streamOutput → serial_input.

- Typical outputs: date → DataViewer.input_0; epoch_milliseconds → DataViewer.input_1; Acc_X → DataViewer.input_2; Acc_Y → DataViewer.input_3.

Data format

- Input: UInt8 serial bytes (9600 baud). Outputs: scalars and a 4-element quaternion.

Notes

- Connect the output of a serial port component; see the example WitMotion_WT901B_imu_demo.

References

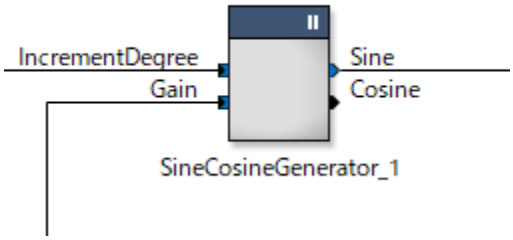
- [1] WitMotion Shenzhen Co., Ltd., WT901 series serial communication protocol (data packet format used by WT901B). https://www.wit-motion.com/ — Used for: serial data packet format.

Input Ports	Data type	Description
serial_input	UInt8	Raw bytes from the IMU serial port.

Output Ports	Data type	Description
date	UInt32	Date since IMU boot (20YYMMDD).
epoch_milliseconds	UInt64	Time in millisecond since IMU boot.
Acc_X	Float64	Acceleration along X in m/s².
Acc_Y	Float64	Acceleration along Y in m/s².
Acc_Z	Float64	Acceleration along Z in m/s².
Gyro_X	Float64	Angular rate about X in deg/s.
Gyro_Y	Float64	Angular rate about Y in deg/s.
Gyro_Z	Float64	Angular rate about Z in deg/s.
Angle_roll	Float64	Roll angle in degree.
Angle_pitch	Float64	Pitch angle in degree.
Angle_yaw	Float64	Yaw angle in degree.
Magnetic_X	Float64	Magnetic field along X.
Magnetic_Y	Float64	Magnetic field along Y.
Magnetic_Z	Float64	Magnetic field along Z.
Data0	UInt16	Data port channel 0 value.
Data1	UInt16	Data port channel 1 value.
Data2	UInt16	Data port channel 2 value.
Data3	UInt16	Data port channel 3 value.
air_pressure	UInt32	Air pressure in Pa.
temperature	Float64	Temperature in degree Celsius.
Quaternion	Float64 vector	Orientation quaternion (4 elements).

Properties	Data type	Description
-		

Hi_Link_LD2410C



Driver for the Hi-Link HLK-LD2410C 24 GHz mmWave human presence radar connected through a serial port; outputs the decoded target state, distances and energies.

Processing

- At start the port is opened at 256000 baud (8N1). The sensor is put into configuration mode, the firmware version, distance resolution and current detection settings (maximum gates, sensitivities, no-one duration) are written to the console, engineering mode is switched on or off according to engineering_mode, and configuration mode is closed. Every command waits for its acknowledgement; if one does not come, the component stops with an error.

- If apply_sensor_settings is true, the detection settings that differ from the properties are written to the sensor and read back for checking. These settings are stored in the sensor and remain after power-off. If it is false, the sensor settings are not changed.

- The serial port is read every polling_ratio_in_ms. Report frames (header F4 F3 F2 F1, tail F8 F7 F6 F5) are extracted from the received bytes and checked (length, AA/55/00 markers, tail). Malformed frames are dropped; their number is written to the console when the component stops.

- Each valid frame is output once; all outputs of a frame carry the time at which the bytes were read.

- polling_ratio_in_ms is limited to 50 ms (20 Hz, twice the report rate), so a read normally contains at most one frame. Reads that still contained several frames (they share one timestamp) are counted and written to the console when the component stops.

Usage

- Typical outputs: moving_distance_in_m → Oscilloscope.inputA; stationary_distance_in_m → Oscilloscope.inputB; detection_distance_in_m → Oscilloscope.inputC; target_state → Oscilloscope.inputA.

Data format

- target_state: 0 no target, 1 moving, 2 stationary, 3 moving and stationary (4 to 6 only during background noise detection).

- Distances are converted from cm to m. Energies are 0 to 100.

- Engineering mode only: moving_gate_energy and stationary_gate_energy have one value per distance gate 0 to N (9 values with the factory setting; 0.75 m per gate), light_level is 0 to 255 and out_pin is the state of the OUT pin. In basic mode these four outputs are not written.

Notes

- Set port_name to the serial device (e.g. COM3 or /dev/ttyUSB0). The sensor must use its factory baud rate 256000.

- The detection settings are changed only when apply_sensor_settings is true. The property defaults equal the factory settings; run once with them and apply_sensor_settings = true to restore the factory settings.

- The sensor reports at about 10 Hz.

- Version 1.02: the raw data output was replaced by the decoded outputs; engineering_mode and the detection setting properties were added.

- Version 1.03: polling_ratio_in_ms must be 1 to 50; a larger value (e.g. 100 saved by version 1.01) stops the component with an error.

References

- [1] Shenzhen Hi-Link Electronic Co., Ltd., HLK-LD2410C Serial Communication Protocol, V1.07, 2024-08-05. — Used for: serial frames, commands and report data.

Input Ports	Data type	Description
-		

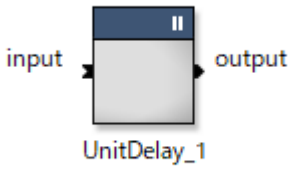
Output Ports	Data type	Description
target_state	Integer32	Target state: 0 none, 1 moving, 2 stationary, 3 both (4 to 6: background noise detection).
moving_distance_in_m	Float64	Distance of the moving target in meter.
moving_energy	Integer32	Energy of the moving target (0 to 100).
stationary_distance_in_m	Float64	Distance of the stationary target in meter.
stationary_energy	Integer32	Energy of the stationary target (0 to 100).
detection_distance_in_m	Float64	Detection distance in meter.
moving_gate_energy	Integer32 vector	Motion energy of each distance gate 0 to N (engineering mode only).
stationary_gate_energy	Integer32 vector	Stationary energy of each distance gate 0 to N (engineering mode only).
light_level	Integer32	Value of the light sensor, 0 to 255 (engineering mode only).
out_pin	Integer32	State of the OUT pin, 1 when a person is detected (engineering mode only).

Properties	Data type	Description
port_name	Text	Serial port name (e.g. COM3 or /dev/ttyUSB0). Must not be empty. (default: COM1)
engineering_mode	Bool	If true, the sensor is put into engineering mode and the per-gate energies, light level and OUT pin state are output. (default: true)
polling_ratio_in_ms	Integer	Period at which the serial port is read, in millisecond (1 to 50; otherwise the component stops with an error). The limit keeps the polling at 20 Hz or faster, twice the report rate of about 10 Hz. (default: 10)
apply_sensor_settings	Bool	If true, the five detection settings below are written to the sensor (only the values that differ). They are stored in the sensor and remain after power-off. (default: false)
max_moving_gate	Integer	Farthest gate (2 to 8) for moving targets. Used only when apply_sensor_settings is true. (default: 8)
max_stationary_gate	Integer	Farthest gate (2 to 8) for stationary targets. Used only when apply_sensor_settings is true. (default: 8)
no_one_duration_in_s	Integer	Time in second (0 to 65535) after the last detection before the state becomes no target. Used only when apply_sensor_settings is true. (default: 5)
moving_sensitivity	Text	Motion sensitivity of gates 0 to 8: nine comma-separated integers from 0 to 100. A target is detected when its energy exceeds the value; 100 disables the gate. Used only when apply_sensor_settings is true. (default: 50,50,40,30,20,15,15,15,15)
stationary_sensitivity	Text	Stationary sensitivity of gates 0 to 8 (same format). Gates 0 and 1 cannot be set and are ignored. Used only when apply_sensor_settings is true. (default: 0,0,40,40,30,30,20,20,20)

3.5 Signal Processing components

You can use the help components on demand.

UnitDelay



Delays a signal by delay_length samples.

Processing

- Samples are stored in a buffer of delay_length; each new input outputs the sample received delay_length samples earlier.

- The output carries the timestamp of the sample just received (the value is the one received delay_length samples earlier).

- For the first delay_length samples, zeros of the same length as the input are output.

Usage

- Typical inputs: data_vectorizer.o_data → input.

- Typical outputs: output → GeoDistance.target_geoPoint.

Data format

- One input/output pair of the type selected by data_type is created; vectors are supported.

- The vector length may change from sample to sample, up to the buffer size of the first sample; a delayed sample keeps its own length.

- delay_length must be 1 or more.

Input Ports	Data type	Description
input_int8	Int8	Input signal (present when data_type = Int8).

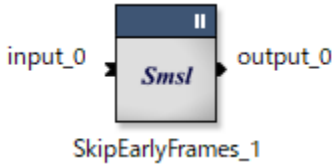
input_int16	Int16	Input signal (present when data_type = Int16).
input_int32	Integer32	Input signal (present when data_type = Int32).
input_int64	Integer64	Input signal (present when data_type = Int64).
input_uint8	UInt8	Input signal (present when data_type = Uint8).
input_uint16	UInt16	Input signal (present when data_type = Uint16).
input_uint32	UInt32	Input signal (present when data_type = Uint32).
input_uint64	UInt64	Input signal (present when data_type = Uint64).
input_float32	Float32	Input signal (present when data_type = Float32).
input_float64	Float64	Input signal (present when data_type = Float64).
input		Signal to delay, of the type selected by data_type (vectors supported).

Output Ports	**Data type**	**Description**
output_int8	Int8 vector	Delayed signal (present when data_type = Int8).
output_int16	Int16 vector	Delayed signal (present when data_type = Int16).
output_int32	Integer32 vector	Delayed signal (present when data_type = Int32).
output_int64	Integer64 vector	Delayed signal (present when data_type = Int64).
output_uint8	UInt8 vector	Delayed signal (present when data_type = Uint8).
output_uint16	UInt16 vector	Delayed signal (present when data_type = Uint16).
output_uint32	UInt32 vector	Delayed signal (present when data_type = Uint32).
output_uint64	UInt64 vector	Delayed signal (present when data_type = Uint64).
output_float32	Float32 vector	Delayed signal (present when data_type = Float32).
output_float64	Float64 vector	Delayed signal (present when data_type = Float64).
output		Input delayed by delay_length samples; timestamp = that of the sample just received.

Properties	**Data type**	**Description**

delay_length	Integer	Delay in number of samples (1 = one-sample delay). (default: 1)
data_type	ENUM	Data type of the input and output. Options: Int8, Int16, Int32, Int64, Uint8, Uint16, Uint32, Uint64, Float32, Float64. (default: Int8)

SkipEarlyFrames



Drops the first n_skip_early_frames frames of each input and passes the following frames unchanged.

Processing

- Place it right after a sensor so that every downstream component receives the same frames (for example, frames that are broken just after the sensor starts).

- nb_channels input/output pairs are created; each pair counts its own frames from the start of the diagram and is not reset.

- Data, timestamp, quality, frequency and Misc1 to Misc3 are copied; the output takes the type, type name and unit of its input at the first frame it passes.

Usage

- Typical inputs: hdl_64e_s2_convert_to_XYZ.XYZ_points → input_0; pandar_xt_convert_to_XYZ.o_XYZ_points → input_0; pandarX_udp_decoder.o_intensities → input_1; ouster_lidar.pointcloud_ret1 → input_0.

- Typical outputs: output_0 → Rotation3D.xyz_points_in; output_0 → DataViewer.input_0; output_0 → GL2_3D_Viewer.object_2_data; output_0 → rings_transpose.pc.

Data format

- Supported inputs: integer and float vectors (Integer8/16/32/64, UnsignedInteger8/16/32/64, Float32, Float64), IplImage and custom structures without pointers (for example SMSL_MapSpec).

- The vector length may change from frame to frame, up to the buffer size of the first frame passed; an IplImage must keep its size, channels and depth.

- Other types, or a change of type or size after the first frame passed, stop the component with an error.

Notes

- Frames are counted, not time: with n_skip_early_frames = 0 every frame is passed.

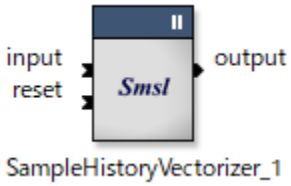
Input Ports	Data type	Description
input	Any	Inputs; nb_channels ports named input_0, input_1, ...

input_$1		Input $1 (integer or float vector, IplImage or custom structure).

Output Ports	**Data type**	**Description**
output	AnyType vector	Outputs; nb_channels ports named output_0, output_1, ... (output_k passes input_k).
output_$1		input_$1 without its first n_skip_early_frames frames; same type and timestamps as the input.

Properties	**Data type**	**Description**
nb_channels	Integer	Number of input/output pairs (1 or more). (default: 1)
n_skip_early_frames	Integer	Number of frames dropped at the start, per input (0 or more; 0 = none). (default: 3)

SampleHistoryVectorizer



Keeps the last max_samples samples of one numeric input and outputs them joined in one vector, oldest first, each time a sample arrives.

Processing

- Each sample is a vector of K values; the output is [sample(t-n+1), ..., sample(t-1), sample(t)] with n = 1, 2, ... up to max_samples, so its length is K x n. When max_samples samples are kept, the oldest one is dropped (ring buffer).

- Use it to draw a trajectory from poses that arrive one by one, or to feed a moving window of one signal to MaxMinMeanforArray.

- reset: while the last value received is not 0, and in a frame where a non-zero value arrived (also a short pulse), the kept samples and the current sample are dropped and nothing is output. After the reset, the output grows again one sample at a time.

Data format

- Supported inputs: integer and float vectors (Integer8/16/32/64, UnsignedInteger8/16/32/64, Float32, Float64). The output takes the type, type name and unit of the input at the first sample.

- K is set by the first sample and must not change; a change of length or type stops the component with an error.

Notes

- Samples are joined over time on one input; to join several inputs of the same time, use the RTMaps Vectorizer.

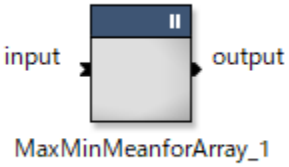
Input Ports	Data type	Description
input	Numbers	Signal to keep (integer or float vector of fixed length K).
reset	UInt8	Reset signal (optional, UnsignedInteger8): non-zero drops the kept samples; nothing is output while it is non-zero.

Output Ports	Data type	Description

output	AnyType vector	Last n samples (n <= max_samples) joined oldest first; length K x n; same type as the input.

Properties	**Data type**	**Description**
max_samples	Integer	Number of samples kept (1 or more). (default: 10)
output_timestamp	ENUM	Timestamp of the output: newest = the sample just received, oldest = the oldest sample in the output (first group). Options: newest, oldest. (default: newest)

MaxMinMeanforArray



Computes the maximum, minimum or mean of the elements of a vector.

Usage

- Typical inputs: number_generator.o_float64 → input.

- Typical outputs: output → DataViewer.input_0; output → DataViewer.input_1; output → DataViewer.input_2.

Data format

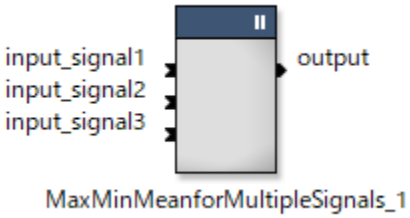
- Input: Float64 vector. Output: Float64 scalar.

Input Ports	Data type	Description
input	Float64	Input vector.

Output Ports	Data type	Description
output	Float64	Max, min or mean of the input elements.

Properties	Data type	Description
mode	ENUM	Statistic to compute. Options: max, min, mean. (default: max)

MaxMinMeanforMultipleSignals



Computes the maximum, minimum or mean across several scalar signals.

Processing

- n_input_signal inputs are created; the statistic is computed over their current values.

Usage

- Typical inputs: number_generator.o_float64 → input_signal1; number_generator.o_float64 → input_signal2; number_generator.o_float64 → input_signal3.

- Typical outputs: output → DataViewer.input_0; output → DataViewer.input_1; output → DataViewer.input_2.

Data format

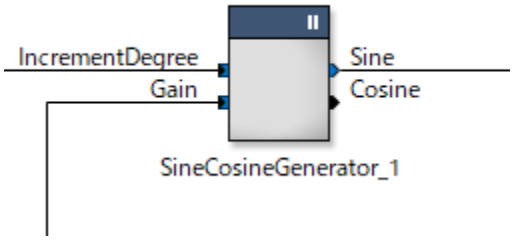
- Inputs: Float64 scalars (vector size 1). Output: Float64 scalar.

Input Ports	Data type	Description
input	Float64	Scalar input signals; n_input_signal ports named input_signal1, input_signal2, ...
input_signal$1		Scalar input signal number $1 (vector size 1).

Output Ports	Data type	Description
output	Float64	Max, min or mean of the inputs.

Properties	Data type	Description
n_input_signal	Integer	Number of input ports to create. (default: 1)
mode	ENUM	Statistic to compute. Options: max, min, mean. (default: max)

NumericalTextWriter



Writes numerical signals to a CSV file as hexadecimal or binary text.

Processing

- A file <output_filename_header>_<start time in ns>.csv is created in output_path.

- Each received sample is written as a row: timestamp, signal name, then the values in hex (0x..) or binary (0b..).

Data format

- Inputs: UInt8 vectors; n_input_signal ports are created.

Notes

- Useful to log bit/byte-level data, e.g. with Byte2bits/bits2Byte.

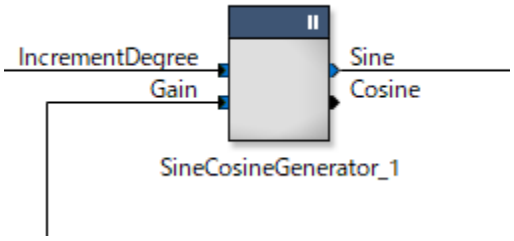
Input Ports	Data type	Description
input	UInt8	Signals to write; n_input_signal ports named input1, input2, ...
input$1		Signal number $1 to write to the CSV file.

Output Ports	Data type	Description
-		

Properties	Data type	Description
n_input_signal	Integer	Number of input ports to create. (default: 1)
output_format	ENUM	Number format of the values. Options: hex, bin. (default: hex)
add_format_identifier	Bool	If true, values are prefixed with 0x or 0b. (default: true)
output_path	Text	Output folder (created if necessary).
output_filename_header	Text	Prefix of the output file name. (default: rtmaps)
signal_name	Text	Signal name written in each row. (default: signal_name)

input$1_name	Text	Signal name written in the CSV rows of input$1.

bits2Byte



Composes one byte from 8 bit signals.

Processing

- Byte = Σ (bit_k & 1) << k, k = 0..7 (bit_0 is the LSB).

Usage

- Typical inputs: Devectorizer.element0 → bit_0; Devectorizer.element1 → bit_1; Devectorizer.element2 → bit_2; Devectorizer.element3 → bit_3.

- Typical outputs: Byte → DataViewer.input_0.

Data format

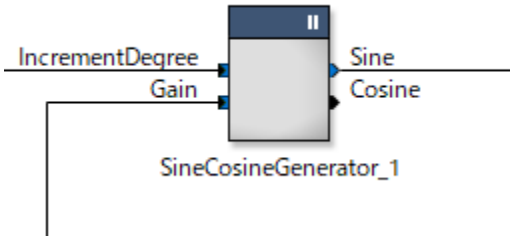
- Inputs: UInt8 scalars (0 or 1). Output: UInt8 scalar.

Input Ports	Data type	Description
bit_0	UInt8	Bit 0 (LSB).
bit_1	UInt8	Bit 1.
bit_2	UInt8	Bit 2.
bit_3	UInt8	Bit 3.
bit_4	UInt8	Bit 4.
bit_5	UInt8	Bit 5.
bit_6	UInt8	Bit 6.
bit_7	UInt8	Bit 7 (MSB).

Output Ports	Data type	Description
Byte	UInt8	Composed byte.

Properties	Data type	Description
-		

Byte2bits



Separates one byte into its 8 bits.

Processing

- bits[k] = (Byte >> k) & 1, k = 0..7 (element 0 is the LSB).

Usage

- Typical inputs: number_generator.o_uint8 → Byte.

- Typical outputs: bits → Devectorizer.iVector.

Data format

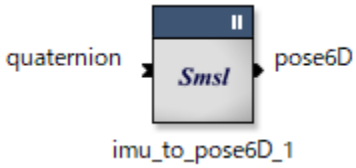
- Input: UInt8 scalar. Output: UInt8 vector of 8 elements.

Input Ports	Data type	Description
Byte	UInt8	Input byte (first element used).

Output Ports	Data type	Description
bits	UInt8 vector	Bits of the byte, LSB first.

Properties	Data type	Description
-		

imu_to_pose6D



Converts the orientation quaternion of an IMU or AHRS to pose6D, whatever the sensor.

Processing

- The quaternion is normalized and converted to a rotation matrix, then to roll, pitch and yaw (R = Rz(yaw) Ry(pitch) Rx(roll)).

- With reference_frame = NED the rotation is converted to world ENU and body FLU: the roll keeps its sign, the pitch changes its sign and yaw = 90 deg - heading.

- The output carries the timestamp of the input.

- Sensor fusion is not done here: use a sensor that outputs its orientation, or imu_orientation_estimator.

Data format

- quaternion must have 4 elements; other sizes stop the component with an error.

- pose6D is [0, 0, 0, roll, pitch, yaw] in m and rad. The translation is always 0 because an IMU cannot measure a position.

Notes

- A sample that is not finite, or with a norm below 1e-6, is skipped; a warning is reported once.

- The output is the orientation of the IMU itself. Apply the mounting of the IMU, from Sensor Calibration File Parser, where the orientation is used.

- The reference of the yaw depends on the sensor; a 6-axis sensor drifts in yaw.

References

- [1] T. Foote and M. Purvis, REP 103: Standard Units of Measure and Coordinate Conventions, ROS Enhancement Proposals. https://www.ros.org/reps/rep-0103.html — Used for: world ENU and body FLU axes, units.

- [2] REP 145: Conventions for IMU Sensor Drivers, ROS Enhancement Proposals. https://www.ros.org/reps/rep-0145.html — Used for: conventions of the orientation given by IMU drivers.

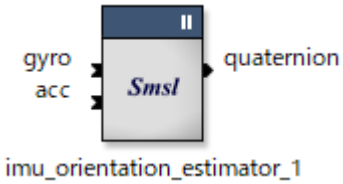
- [3] sensor_msgs/Imu message definition, ROS common_msgs. https://docs.ros.org/en/api/sensor_msgs/html/msg/Imu.html — Used for: xyzw order of the quaternion.

Input Ports	Data type	Description
quaternion	Float64	Orientation quaternion (4 elements), in the order set by quaternion_order.

Output Ports	Data type	Description
pose6D	Float64 vector	Orientation of the IMU: [0, 0, 0, roll, pitch, yaw] (m, rad).

Properties	Data type	Description
quaternion_order	ENUM	Order of the elements: wxyz (scalar first) or xyzw (scalar last, as in ROS sensor_msgs/Imu). Options: wxyz, xyzw. (default: wxyz)
reference_frame	ENUM	Frame of the input: ENU (world ENU, body FLU, as in ROS) or NED (world NED, body FRD). Options: ENU, NED. (default: ENU)

imu_orientation_estimator



Estimates the orientation quaternion from a gyro and an accelerometer, optionally with a magnetometer, without GNSS.

Processing

- Each gyro sample moves the orientation by the mean of the last two samples, then the latest acceleration corrects roll and pitch.

- filter_mode selects the correction: Mahony (complementary filter, also estimates the gyro bias) or Madgwick (gradient descent).

- With use_magnetometer the latest field also corrects the yaw, which then does not drift.

- The first gyro sample gives the first orientation from the acceleration (and the field); without a magnetometer the yaw starts at 0.

- gyro, acc and mag need not have the same timestamps; the output carries the timestamp of gyro.

Data format

- gyro, acc and mag have 3 elements; other sizes stop the component with an error. Sensors with one output per axis: gather them with the Vectorizer of RTMaps (Sampling mode = Synchronized).

- acc is the specific force: +Z when level at rest. acc and mag are used for their direction only, in any unit.

- quaternion is [w, x, y, z], body to world. With body axes X forward, Y left, Z up, connect it to imu_to_pose6D with wxyz and ENU.

Notes

- Without a magnetometer the yaw is relative and drifts. With one, the yaw is referenced to the magnetic north; magnetic_declination_in_deg (east positive) makes it the true north. See the topic Magnetic declination.

- Accelerations of the motion are taken as gravity: roll and pitch have an error during strong acceleration.

- The magnetometer must be calibrated beforehand; motors, steel and batteries disturb it.

- A sample of acc or mag that is not finite, or with a norm below 1e-6, is ignored: the last valid one is used. A gyro sample that is not finite is skipped. A warning is reported once.

- With GNSS, use an EKF: this component does not use a position.

References

- [1] R. Mahony, T. Hamel and J.-M. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," IEEE Transactions on Automatic Control, vol. 53, no. 5, pp. 1203-1218, 2008. https://doi.org/10.1109/TAC.2008.923738 — Used for: Mahony update.

- [2] S. O. H. Madgwick, A. J. L. Harrison and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," 2011 IEEE International Conference on Rehabilitation Robotics, pp. 1-7, 2011. https://doi.org/10.1109/ICORR.2011.5975346 — Used for: Madgwick update.

- [3] T. Foote and M. Purvis, REP 103: Standard Units of Measure and Coordinate Conventions, ROS Enhancement Proposals. https://www.ros.org/reps/rep-0103.html — Used for: axes and units.

Input Ports	Data type	Description
gyro	Float64	Angular rate [x, y, z] in gyro_unit. Each sample updates the orientation and gives one output.
acc	Float64	Acceleration [x, y, z], any unit. The latest valid sample is used.
mag	Float64	Magnetic field [x, y, z], any unit, calibrated. Only with use_magnetometer. The latest valid sample is used.

Output Ports	Data type	Description
quaternion	Float64 vector	Orientation [w, x, y, z], body to world.

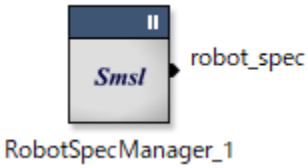
Properties	Data type	Description
filter_mode	ENUM	Mahony or Madgwick. Options: Mahony, Madgwick. (default: Mahony)
gyro_unit	ENUM	Unit of gyro: rad_per_s or deg_per_s. Options: rad_per_s, deg_per_s. (default: rad_per_s)
use_magnetometer	Bool	Corrects the yaw with the input mag. (default: false)
max_gap_in_ms	Integer	When the time between two gyro samples is longer than this, or not positive, the orientation is held for that sample. (default: 500)

mahony_kp	Float64	Strength of the correction (1/s). The time constant is 1/mahony_kp. Must be greater than 0. (default: 1.0)
mahony_ki	Float64	Strength of the estimation of the gyro bias (1/s^2). 0 disables it. Must not be negative. (default: 0.01)
madgwick_beta	Float64	Speed of the correction (rad/s). Must be greater than 0. (default: 0.1)
magnetic_declination_in_deg	Float64	Magnetic declination, east positive, -180 to 180. With 0 the yaw is referenced to the magnetic north. (default: 0.0)

3.6 Hardware Specification components

The components that give the specification of the hardware, such as the robot body, to the other components.

RobotSpecManager



Reads a robot spec XML file and outputs the dimensions and mass of the robot (SMSL_RobotSpec) and, if the file has them, its motion limits (SMSL_RobotMotionLimits), once at the start.

Processing

- At the start the component reads the file given by spec_file, checks it and outputs the robot spec once. It then waits until the diagram stops. The components that use the robot take the values by wire, so they need no properties for the size of the robot.

- The outline (footprint) of the robot body is given as a rectangle (distances from the robot origin to the front, rear, left and right edges), a circle or a polygon (3 to 32 vertices, ROS footprint style). The output outline is always counterclockwise. The inscribed radius is the shortest distance from the origin to an edge and the circumscribed radius is the longest distance from the origin to a vertex. For a circle both are the radius (the outline is a regular 16-gon containing the circle).

- The robot type (differential, omni, ackermann or drone) selects which attributes of the body and motion elements are required, optional or not allowed. The body element gives the height, the mass and center of gravity, and the wheel layout. The motion element gives the limits of the machine: maximum velocities, accelerations, decelerations (including the emergency deceleration), yaw rate and, for a drone, the tilt angle; for an Ackermann robot, the maximum rate of steering. A direction in which the robot cannot move is 0 in the output.

- The values are shown in the log at the start because the structures cannot be seen in DataViewer.

- The component stops with an error when the file cannot be read, when an attribute is missing or does not fit the type, when a value is not a finite number or is out of range, when the polygon crosses itself or has no area, or when the robot origin is not inside the outline.

Usage

- Typical outputs: robot_spec → rings_pose_estimation_icp.robot_spec; robot_motion_limits → rings_pose_estimation_icp.robot_motion_limits; robot_spec →

rings_pose_estimation_p2plane.robot_spec; robot_motion_limits → rings_pose_estimation_p2plane.robot_motion_limits.

Data format

- Robot coordinates: right-handed, X forward, Y left, Z up, in meters. The origin is the point that the robot pose (pose2D) refers to. For a ground robot z = 0 is the ground.

- File: UTF-8 XML. The root element is robot_spec with the attribute type. It has one footprint element (attribute shape: rectangle, circle or polygon), and optionally one body element and one motion element. The type attribute is required when body or motion is written. Unknown elements and attributes are errors.

- Output robot_spec: custom structure SMSL_RobotSpec (632 bytes): vertex count, type, inscribed and circumscribed radii, up to 32 vertices (x, y, counterclockwise), then the values of the body element (0 when not written or not used by the type). Output robot_motion_limits: custom structure SMSL_RobotMotionLimits (120 bytes), made only when the motion element is written; velocities and accelerations are 0 or more.

- Samples for the four types are in the example folder (robot_spec_differential.xml, robot_spec_omni.xml, robot_spec_ackermann.xml, robot_spec_drone.xml).

Notes

- The sensor mounts (position and orientation of each sensor) are not given by this component. Use the RTMaps standard component Sensor Calibration File Parser and connect its matrix output to Rotation3D (Matrix Input). See the Topic "Robot spec and sensor mounting".

- The motion element holds the limits of the machine only. The speed you use today stays in the properties of the component that uses it, below the limit.

- The friction coefficient is not part of the robot spec: it belongs to the floor.

- If you add or remove the motion element in the file, select spec_file again so that the output robot_motion_limits is made or removed.

Input Ports	Data type	Description
-		

Output Ports	Data type	Description
robot_spec	SMSL_RobotSpec vector	Dimensions and mass of the robot as the custom structure SMSL_RobotSpec: outline, inscribed and circumscribed radii, type and the values of the body element. Output once at the start.
robot_motion_limits	SMSL_RobotMotionLimits vector	Motion limits of the robot as the custom structure SMSL_RobotMotionLimits (maximum velocities, accelerations,

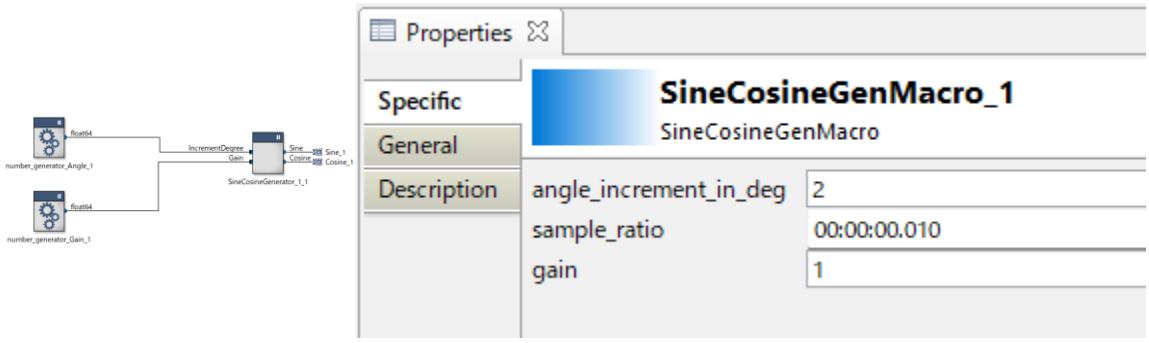
		decelerations, yaw rate, tilt angle). Made only when the motion element is written in the file; output once at the start.

Properties	**Data type**	**Description**
spec_file	Text	Path of the robot spec XML file (UTF-8). See the description of the component for the format.

4. Macro Component

This package provides helpful macro components. You can get hints to use the components in them. The frequently edited properties are listed in the Specific tab in the Properties of the macros.

SineCosineGenMacro



Macro component that outputs a sine and a cosine wave with a given angle step, period and gain.

Processing

- A number_generator outputs the angle increment periodically; each sample advances the angle of SineCosineGenerator by angle_increment_in_deg.

- Another number_generator supplies the gain once at start.

Usage

- Use as a test signal, e.g. Sine_1 → Add_Diff.input1 (add noise) or Oscilloscope.inputA.

- Defined in SMSL_BasicToolkit_Macros.rtmc; used in the RobotPositioningToolkit Kalman filter examples (e.g. KalmanFilter_1D_demo.rtd).

Data format

- Outputs: Float64 scalars, one sample per period.

Notes

- Despite its name, sample_ratio is the output period in microseconds (default 10000 = 100 Hz).

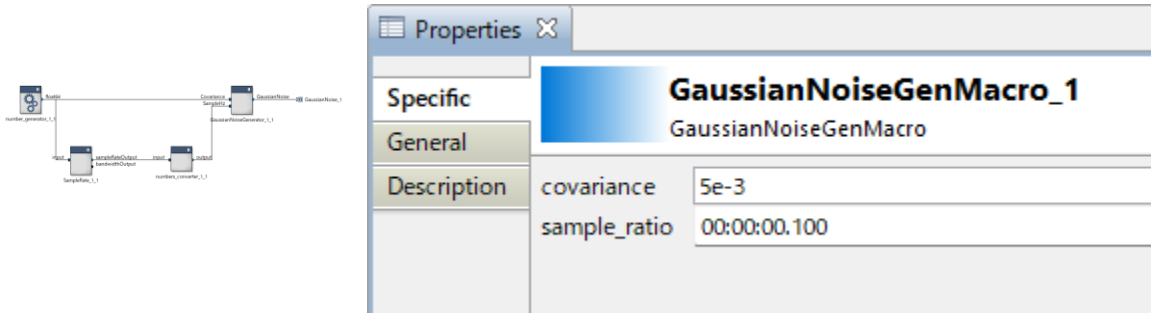
- The wave frequency is angle_increment_in_deg / 360 × (1e6 / sample_ratio) Hz.

Input Ports	Data type	Description
-		

Output Ports	Data type	Description
Sine_1	Float64 vector	Gain·sin(θ).
Cosine_1	Float64 vector	Gain·cos(θ).

Macro Properties	Data type	Description
angle_increment_in_deg	Float64	Angle increment in degree per output sample. (default: 2)
sample_ratio	Integer	Output period in microseconds (period of the internal angle number_generator). (default: 10000)
gain	Float64	Amplitude of the output waves. (default: 1)

GaussianNoiseGenMacro



Macro component that outputs Gaussian noise with a given covariance at a fixed period.

Processing

- A number_generator outputs the covariance value periodically; each sample triggers GaussianNoiseGenerator.

- The variance of each noise sample is the covariance value, whatever the period.

Usage

- Add the output to a clean signal to simulate measurement noise, e.g. GaussianNoise_1 → Add_Diff.input2.

- Defined in SMSL_BasicToolkit_Macros.rtmc; used in the RobotPositioningToolkit Kalman filter examples (e.g. KalmanFilter_1D_demo.rtd).

Data format

- Output: Float64 scalar, one sample per period.

Notes

- Despite its name, sample_ratio is the output period in microseconds (default 100000 = 10 Hz).

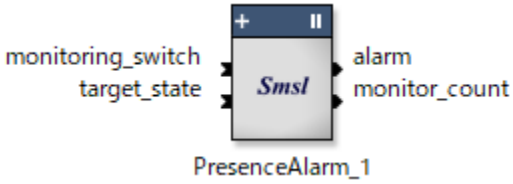
Input Ports	Data type	Description
-		

Output Ports	Data type	Description
GaussianNoise_1	Float64 vector	Gaussian noise sample.

Macro Properties	Data type	Description
covariance	Float64	Covariance (variance) of the Gaussian noise. (default: 5e-3)

sample_ratio	Integer	Output period in microseconds (period of the internal number_generator). (default: 100000)

PresenceAlarm



Macro component that raises an alarm when a person is still present while monitoring is on (left-behind alarm), from the target state decoded by Hi_Link_LD2410C.

Processing

- Condition_count (RTMaps Condition) counts the samples with the condition (a != 0) * ((s >= 1 && s <= 3) * result + 1): 0 while monitoring_switch is 0, 1 while monitoring and nobody is present, n + 1 after a person has been present for n consecutive samples (target_state 1 to 3).

- Condition_alarm latches the alarm with the condition c != 0 && (result != 0 || c > 200): the alarm turns on when the count exceeds 200 samples (about 20 seconds at the report rate of the sensor, about 10 Hz) and stays on until monitoring_switch returns to 0.

- Both Condition components are evaluated whenever an input arrives (Synchro tolerance = -1) and output the value of the condition every time.

Usage

- Connect Hi_Link_LD2410C.target_state → target_state, and a 0/1 switch signal that means that nobody should be here from now on (door lock, door-closed switch, ignition off, end of a task; a Bi-state button for tests) → monitoring_switch. Connect alarm to an LED, a buzzer or a recorder.

- A person who leaves the detection range after monitoring starts becomes absent after the no-one duration of the sensor (default 5 s) and the count returns to 1, so no alarm is raised when leaving plus the no-one duration takes less than about 20 seconds. A person who stays (even motionless, detected by breathing) raises the alarm.

- Defined in SMSL_BasicToolkit_Macros.rtmc.

Data format

- Inputs and outputs: Integer32 scalars, one output sample per evaluation.

Notes

- Time is counted in samples, so the alarm delay changes with the report rate of the sensor. The threshold (200 samples) is in the condition of Condition_alarm; the macro has no properties, so edit the macro (.rtmc) to change it.

- target_state 4 to 6 (background noise detection) counts as absent.

- When the sensor stops reporting, the outputs stop and keep their last values.

- Not a safety device (not certified for functional safety); to protect people, use it only as an aid to a certified sensor.

Input Ports	Data type	Description
monitoring_switch	Integer32	Monitoring switch: non-zero (1) = on, 0 = off. Turning it off resets the count and the alarm.
target_state	Integer32	Target state from Hi_Link_LD2410C: 0 none, 1 moving, 2 stationary, 3 both (4 to 6: background noise detection, counted as absent).

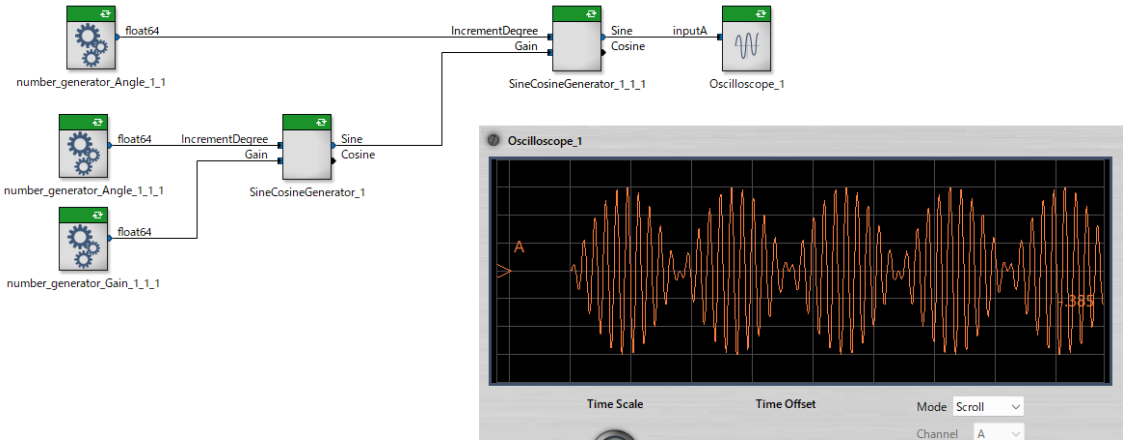
Output Ports	Data type	Description
alarm	Integer32 vector	Alarm: 1 when a person has been present for more than 200 samples while monitoring, latched until monitoring_switch returns to 0; otherwise 0.
monitor_count	Integer32 vector	Count: 0 = not monitoring, 1 = monitoring and nobody present, n + 1 = a person has been present for n consecutive samples.

Properties	Data type	Description
-		

5. Demo Diagrams

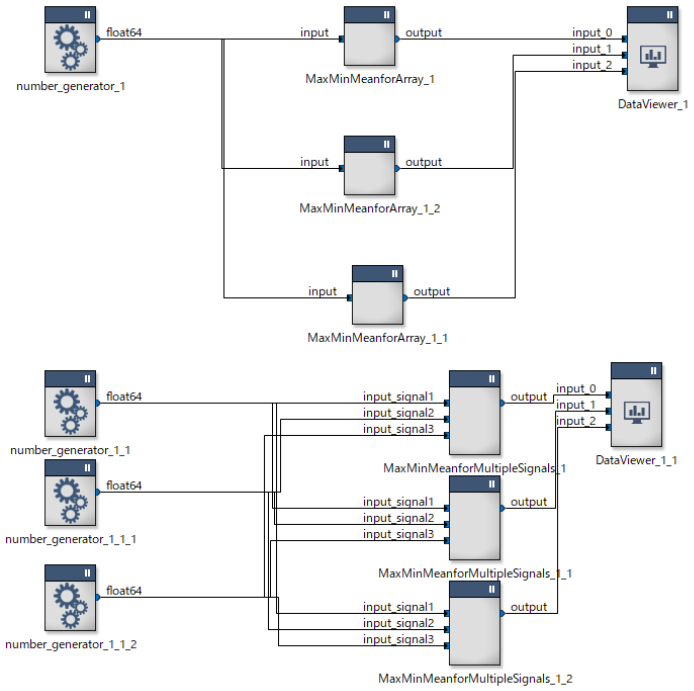
This package provides primitive demo RTMaps diagrams for each component. You can refer to the usage example in them.

DSB_AM_demo



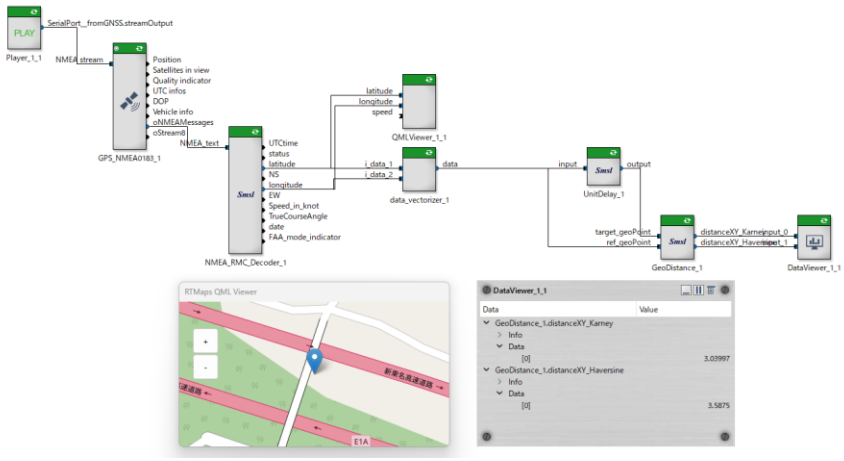
DSB_AM_demo.rtd is an example to create double side band amplitude modulation with two SineCosineGenenerator components.
The first generator determines the modulation amplitude for the second sine generator.

MaxMinMean_demo



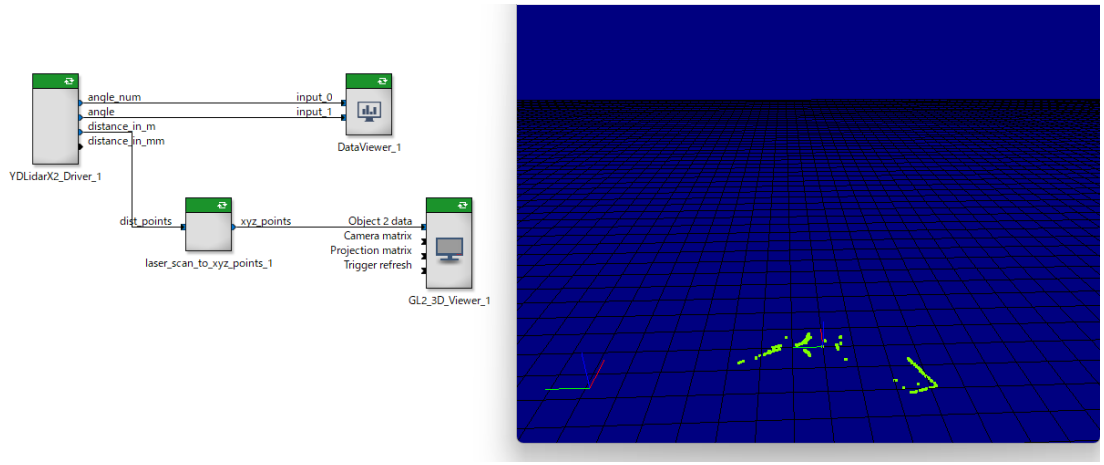
MaxMinMean_demo.rtd is an example to execute the basic calculation for numerical data. Two variation is ready for input data formation, vector signal or multiple signals.

GeoDistance_demo



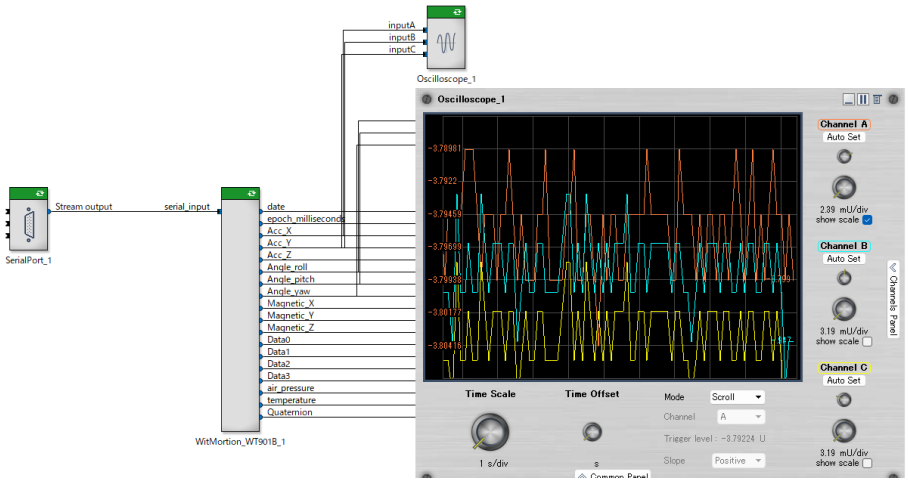
GeoDistance_demo.rtd is an example to calculate the distance between 2 Geographic points with latitude and longitude.
The Geographic points in the input signal moves. The diagram calculate the moving distance by steps with the UnitDelay.
Unfortunately the input signal is not provided by SMSL so far, you need to use any data from GNSS.

YDLidarDataAcquisition_demo



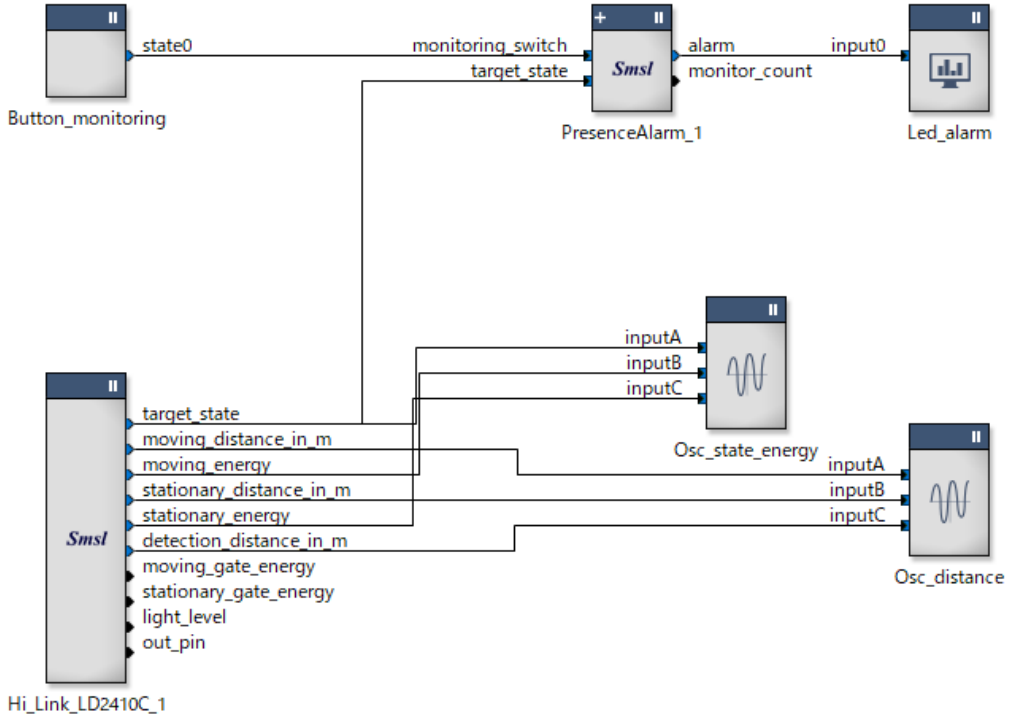
YDLidarDataAcquisition_demo.rtd is an example to data acquisition from YDLidar X2.
You can get 1-D array signal as a laserscan from the driver component.

WitMotion_WT901B_imu_demo



WitMotion_WT901B_imu_demo.rtd is an example to data acquisition from WT901B IMU.
You can get acceleration and angle values of the IMU from the decoder component.

Hi_Link_LD2410C_demo



Hi_Link_LD2410C_demo.rtd is an example to display the detection results of the Hi-Link LD2410C 24 GHz mmWave human presence radar. The target state, distances and energies decoded by the Hi_Link_LD2410C component are shown in Oscilloscopes (distances; target state and energies) and a DataViewer (all outputs, including the energy of each gate, the light sensor and the OUT pin).

The LD2410C is a human presence sensor: it reports whether a person is present (moving / stationary) and the approximate distance (0.75 m per gate, up to about 6 m). It does not measure the distance to walls or objects. On a mobile platform, use it while the platform is stopped:

- Checking the surroundings before starting (AGV, AMR, work vehicles): while stopped, check that nobody is crouching or standing still nearby before moving off. The radar sees through plastic covers, so it can be mounted inside the body.

- Detecting an approaching person (reception and delivery robots): while waiting, start guidance or open a door when a person comes near.

- In-cabin monitoring: after the occupants leave, check that nobody remains in the cabin by detecting the slight motion of breathing (stationary detection). The cabin is about 2 m deep, so setting the maximum gate to 2 or 3 makes it less sensitive to people passing outside.

While the sensor moves, walls and the floor also look like moving targets and false detections increase, so it cannot be used while driving. It is not a safety device (not certified for functional

safety); to protect people, use it only as an aid to a certified sensor.

6. Conclusion

Now you understand the overall SMSL_BasicToolkit package. The author hopes you will make wonderful use of SMSL_BasicToolkit in your RTMaps applications.

7. Revision History

Version	Date	Author	Description
1.0	25th, Dec, 2025	IDA Shogo	Initial release
1.1	16th, Oct, 2026	IDA Shogo	Updated to version 1.1. Added Hi_Link_LD2410C, bits2Byte, Byte2bits and NumericalTextWriter. Modified SineCosineGenerator and WitMotion_WT901B.
1.2	18th, Oct, 2026	IDA Shogo	Updated to version 1.2. Added the decoded target outputs, engineering_mode and the detection-setting properties to Hi_Link_LD2410C. Added the Inverted property to YDLidarX2_Driver. Modified Hi_Link_LD2410C and YDLidarX2_Driver.
1.3	25th, Oct, 2026	IDA Shogo	Updated to version 1.3. Modified Byte2bits, GaussianNoiseGenerator, GeoDistance, Hi_Link_LD2410C, MaxMinMeanforMultipleSignals, NumericalTextWriter, SineCosineGenerator, WitMotion_WT901B, YDLidarX2_Driver, bits2Byte.
1.31	1st, Nov, 2026	IDA Shogo	Updated to version 1.31. Added RobotSpecManager, SampleHistoryVectorizer, SkipEarlyFrames and the macro PresenceAlarm. Modified UnitDelay: writing outside the buffer with data other than 8 bytes, mixed values when the input length changes, and the output timestamp (now the same as the input).
1.32	2nd, Nov, 2026	IDA Shogo	Updated to version 1.32. Modified RobotSpecManager: renamed the output robot_motion to robot_motion_limits and the element

			<motion> of the spec file to <motion_limits>.
1.33	7th, Nov, 2026	IDA Shogo	Updated to version 1.33. Added imu_orientation_estimator and imu_to_pose6D. Modified StaticImageGenerator: added output_mode (Once / Repeat). Modified WitMotion_WT901B: added the outputs Gyro_X, Gyro_Y and Gyro_Z.