openSEA Software Development Kit

Reference Guide

Subject: *This document provides reference information to integrators about the openSEA Suite of software applications. Definitions of terms used in our SDK, a note about Greensea's core technology, and a communications overview are included*

Revision History

Revision 005

Revision	Initials	Date	Comments
001	Meb	09/02/14	Initial.
002	Jdr	09/11/14	Added content.
003	Bwk	09/20/14	Reviewed.
004	Meb	09/22/14	Edited content.
005	Meb	06/18/15	Edited content.

Table 1: Revision history.

Table of Contents

Index of Tables

Illustration Index

The openSEA Suite of Applications

Software Development Kit

1 Introduction

The openSEA Suite of applications provides standard commercial control and navigation applications with well-defined interfaces that can be used stand-alone or as parts of a larger system. It's this foundation of enabling software our Balefire Operator Workspace is built upon.

The openSEA Suite employs openDEVICE, openINS, openMNGR and openCMD. openDEVICE consists of a library of sensor, actuator, and other device interfaces as well as a sheriff and her deputies who monitor the running status of the devices, starting and stopping them as necessary. These device interfaces provide data in a common format that can be used by openINS or other consumers in the system. openINS is the stand-alone application that navigation sensor data and calculates a final state estimate using a multi-state Kalman filter engine. openMNGR provides the high-level layers of autonomy. The openCMD application handles vehicle control by calculating a maneuvering solution to achieve the commanded vehicle state.

When everything comes together you will want for nothing. At least not in the realm of commercial control and navigation.

2 Purpose

This document is an introduction to the openSEA Suite. We'll provide an overview of the technology, descriptions of how it fits into a vehicle system, and explanations of how openSEA uses its open communications architecture to create an effective, accurate, and flexible navigation and control solution.

We want to help users with installing and integrating the applications that make up the openSEA Suite and assume basic working knowledge of computer operating systems, network communications and even a little programming. If you do not have such working knowledge, or are simply searching for an operator manual, you will probably get overwhelmed and might bug out a little.

Don't bug out. We have documentation for all levels of users, so go find that manual.

3 Overview

We at Greensea developed, use, and own the Open Software and Equipment Architecture (openSEA) and believe it's pretty awesome. It consists of a core library and several standard software applications that provide an abstract framework to customize the technology for specific applications. The end game? A flexible and powerful development technology for unmanned underwater vehicles providing device interfaces, navigation, control, and a robust software architecture.

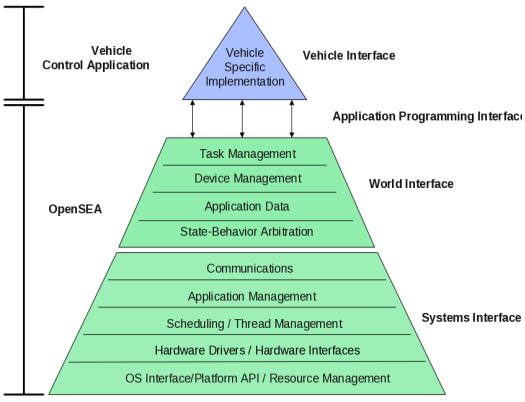

Illustration 1: General openSEA architecture

3.1 The openSEA Library

All openSEA applications are built on the framework provided by the openSEA library. openSEA's Application Programming Interface (API) provides error handling, fault mitigation, communications, thread management, and state behavior control mechanisms. openSEA also provides device interfaces to hundreds upon hundreds of common industry devices as well as a comprehensive numerical package.

3.2 Setup

Getting started is easy. Since there is no single-point of entry for the library, users utilize the components as needed. Simply install the library on your development system and get coding.

3.3 Installation

LibopenSEA and the openSEA SDK are installed using pre-built installation scripts. A typical installation on a Linux system might look like:

1) Open a terminal emulator

2) Change directory to the location of the installation script

$ cd /path/to/install/script

3) Run the installation script

```
$ ./install_XXX.run
```

The library will be installed into /usr/local/lib and the header files will be installed into /usr/local/include.

3.4 The openSEA Application Suite

The openSEA suite of applications provides commercial control and navigation applications with well-defined interfaces that can be used stand-alone or as parts of a larger system.

- **openCMD:** The control application for the openSEA Suite. The application provides station keeping, autopilots, route following, and dynamic positioning for ROVs and AUVs. Using the navigation solution from openINS or a third-party navigation system, openCMD calculates the multi-state control solution for all six degrees of freedom based on given input commands.
- **openDEVICE:** A device server for sensors and actuators that provides data management, communication, forwarding, time-stamping, and aggregation. Interfacing with each device in an independent process over the device's native proprietary protocol implemented in TCP/IP, RS232, RS485, CAN, Modbus, or USB. Using the open interface provided by openDEVICE, client applications can move between various specific devices within a category (compasses, motor controllers, IMUs, etc.) without changing a line of code.
- **openINS:** Our software-based Inertial Navigation System. The application uses an inertial measurement unit (IMU) as the core process and accepts data from any available sensors, such as USBL, DVL, LBL, GPS, depth sensors, and encoders to aid the final navigation solution. Using the numerical package offered by openSEA, openINS provides dynamic multi-state Kalman, extended Kalman, and other digital filters.
- **openMNGR:** A high-level supervisory application that coordinates mission planning, execution, and monitoring. openMNGR provides a solution for complex route planning, surveys, search grids, sonar-based target tracking, task management, and full vehicle autonomy. Utilizing the same architecture

employed by openSEA, openMNGR is flexible for specific applications.

Here is how they can fit together:

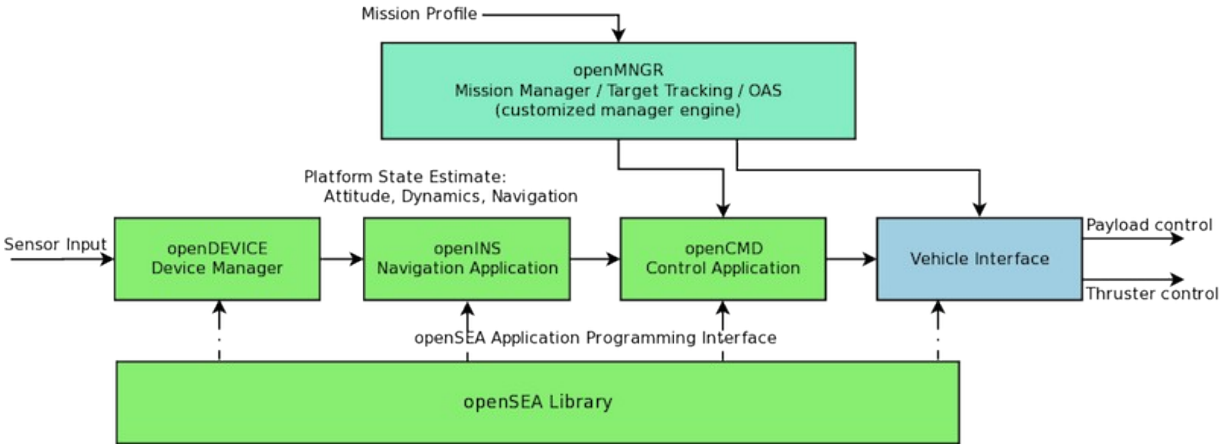

Illustration 2: Flow chart that shows how the openSEA applications work together.

4 How openSEA Applications Communicate

All openSEA applications communicate using a publish/subscribe method aided by the Lightweight Communications and Marshalling (LCM) framework. How this communication process works and how to configure openSEA applications for communication is discussed in the next section. Hang tight.

4.1 What is LCM?

Originally designed by the MIT DARPA Urban Challenge Team, LCM is a set of utilities and libraries that uses UDP multicast to move data between applications on a computer and between computers on a LAN. It does this by encoding or decoding the data into a unified, internally readable format.

The UDP protocol allows for low latency by allowing certain messages to be dropped and new messages taken in, out of order. The LCM messaging protocol is stateless by design. The structure of the LCM code generating utility allows the system to create types from multiple programming languages by adding LCM bindings native to that programming language.

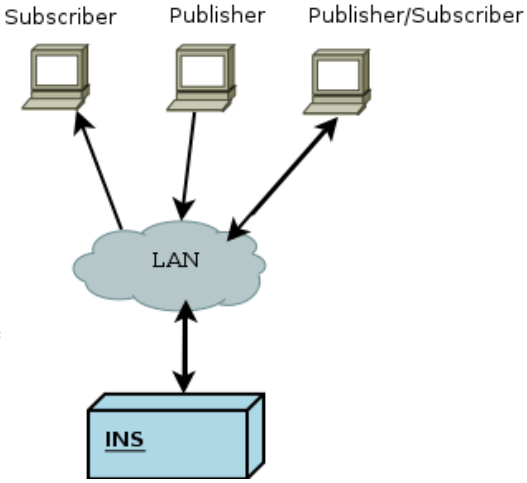

Illustration 3: openINS publish/subscribe method.

4.1.1 Primitive LCM Types

This table is a complete list of primitive LCM types. Not all are used by the openSEA Suite.

Type	*Description*
int8_t	8-bit signed integer
int16_t	16-bit signed integer
int32_t	32-bit signed integer
int64_t	64-bit signed integer
float	32 bit IEEE floating point value
double	64 bit IEEE floating point value
string	UTF-8 string
boolean	True/false logical value
byte	8-bit value

Table 2: Primitive LCM types.

4.1.2 Other LCM Types

Non-primitive types are also used in defining LCM type files. These types can consist of multi-dimensional arrays that contain primitives, constant declarations, and structs. Constants can also be declared inside of a struct as numerical, bitfields, or enumerations. Namespaces are another form of packaging data into structs to be used in an environment where other structs may have the same name.

4.2 General LCM Types for openSEA Applications

So we know LCM types are a collection of data types specific to the system configuration. That means each of these data type files is a collection of definitions and measuring tools packaged together to define that data type internal as an LCM readable type. We've detailed the general types used as building blocks by the openSEA applications in a bunch of tables. To find out the specific types for an application, pick up that application's ICD

4.3 PCOMMS LCM type

PCOMMS is the building block for most of our LCM traffic. It is made up of a vector of analog_t and a vector of digital_t types with some header information. There are three different data structures that comprise the pcomms_t type: struct digital_t, struct analog_t, and struct pcomms_t.

4.3.1 struct digital_t

Variable	*Type*	*Description*
name	string	A structure content for the digital values: The name of the device.
value	boolean	A structure content for the digital values: The Boolean value of the device.

Table 3: LCM type pcomms_t digital signal variables.

4.3.2 struct analog_t

Variable	*Type*	*Description*
name	string	The name of the analog string.
value	double	The value of the analog signal.

Table 4: LCM type pcomms_t analog signal variables.

4.3.3 struct pcomms_t

Variable	*Type*	*Description*
unix_time	double	Unix Time stamps. The message data time stamp in seconds since January 1, 1970, 12 p.m. GMT.
sender_id	string	The ID string of the message passing device.
num_analogs	int32_t	The number of analog values.
analogs[num_analogs]	analog_t vector	Vector of analog values. The vector contains a name and a double value.
num_digitals	int32_t	The number of digital values.
digitals[num_digitals]	digital_t vector	Vector of digital values. The vector contains a name and a double value.

Table 5: LCM type pcomms_t structure.

5 Definitions

Several acronyms are used throughout this document.

AHRS: Attitude and Heading Reference System.

DAQ: Data Acquisition. Device used to interface with discreet analog and digital devices.

DARPA: Defense Advanced Research Projects Agency. Research and development arm of the United States Department of Defense.

DVL : Doppler Velocity Log.

ECEF: Earth Centered Earth Fixed. Common frame of reference used by openSEA in both openINS and openCMD applications. A transform configuration provides the relationship of each sensor to the vehicle frame and openINS transforms the vehicle frame to ECEF.

IMU: Inertial measurement unit. The primary sensor for directly sensing vehicle motion. Provides accelerations and rotational rates. openINS uses data from the IMU as the high-rate process in its estimate filter.

INS: Inertial Navigation System. The set of software specific to providing the navigation and sensor fusion functions for the vehicle as well as a final state estimate.

LAN: Local Area Network.

LBL: Long Base Line underwater acoustic positioning.

LCM: Lightweight Communications and Marshalling. A real-time messaging and data communications system implemented via User Datagram Protocol (See UDP).

MIT: Massachusetts Institute of Technology.

openINS: The Greensea proprietary openSEA software application used for inertial navigation.

openCMD: The Greensea proprietary openSEA software application used for vehicle control.

openDEVICE: The Greensea proprietary openSEA software application used for device management.

openMNGR: The Greensea proprietary openSEA software application that coordinates mission planning, execution, and monitoring.

openSEA: The Open Software and Equipment Architecture. The fundamental software library developed, used, and owned by Greensea for navigation and control applications.

PCOMMS: Primary Communications. “Primary Communications”, or PCOMMS, in openSEA parlance refers generally to the traditional communications from a topside operator interface to the subsea system. Roughly, it is the generic communications between the vehicle components (devices, navigation, and control) and topside components (operator interfaces).

UDP: User Datagram Protocol. An Open Systems Interconnection (OSI) transport layer protocol.

USBL: Ultra Short Baseline underwater acoustic positioning.

YAML: YAML Ain't Markup Language (.yml file format). A markup language used for text-based configuration files in the openSEA system.

6 Go

There are handy example programs in specific application documentation that demonstrate how to craft a program that sends or receives LCM traffic. You should now be ready to dig in to any of our openSEA documentation that you want. Go on. Get after it.

End Page