TTA Standard

정보통신단체표준(국문표준)

TTAx.xx-xx.xxxx/R1 제정일: 20xx년 xx월 xx일

개방형 로봇 소프트웨어 플랫폼 - 제6부 : 장치 추상화를 위한 공통 로봇 인터페이스

Open Platform for Robotic Services - Part 6 : Common Robot Interface for Device Abstraction

한국정보통신기술협회
Telecommunications Technology Association

표준초안 검토 위원회 지능형 로봇 프로젝트그룹(PG413)

표준안 심의 위원회 정보기술 융합 기술위원회(TC4)

	성명	소 속	직위	위원회 및 직위	표준번호
표준(과제) 제안	박홍성	강원대학교	교수	-	
표준 초안 작성자	박홍성	강원대학교	교수	-	
	김미숙	강원대학교	선임	-	
	조영조	ETRI	책임	-	
	성기엽	KAR	대리	-	
사무국 담당	강석규	TTA	선임	-	

본 표준 발간 이전에 접수된 지식재산권 확약서 정보는 본 표준의 '부록(지식재산권 확약서 정보)'에 명시하고 있으며, 이후 접수된 지식재산권 확약서는 TTA 웹사이트에서 확인할 수 있습니다.
본 표준과 관련하여 접수된 확약서 외의 지식재산권이 존재할 수 있습니다.

발행인 : 한국정보통신기술협회 회장
발행처 : 한국정보통신기술협회
13591, 경기도 성남시 분당구 분당로 47
Tel : 031-724-0114, Fax : 031-724-0109
발행일 : 20xx.xx

서 문

1 표준의 목적

이 표준의 목적은 로봇 응용 프로그램을 다수의 로봇 플랫폼에 손쉽게 적용할 수 있으며, 재사용이 가능하도록 로봇 부속 장치에 대한 의존적인 부분을 추상화하기 위하여 작성하였다.

2 주요 내용 요약

이 표준은 개방형 로봇 소프트웨어 플랫폼 상에서 개발된 로봇 시스템의 로봇 부속 장치에 대한 추상화 인터페이스를 규정한다.

3 인용 표준과의 비교 (

3.1 인용 표준과의 관련성

해당 사항 없음

Preface

1 Purpose

The standard is stated to abstract dependent factors of the robot device for its reusability when robot applications having this device are applied into many different robot platforms easily.

2 Summary

The standard regulates abstraction interfaces for robot devices in robot system developed in Open Platform for Robotic Services.

3 Relationship to Reference Standards

None

목 차

개방형 로봇 소프트웨어 플랫폼 -
제6부 : 장치 추상화를 위한 공통 로봇 인터페이스
(Open Platform for Robotic Services -
Part 6 : Common Robot Interface for Device Abstraction)

1 적용 범위

이 표준은 개방형 로봇 소프트웨어 플랫폼 상에서 개발된 로봇 시스템의 로봇 부속 장치에 대한 추상화 인터페이스를 규정한다.

2 인용 표준

KS B 7305-2 개방형 로봇 소프트웨어 플랫폼 - 제2부 : 컴포넌트
KOROS 1067-2 개방형 로봇 소프트웨어 플랫폼 - 제2부 : 컴포넌트

3 용어 정의

3.1 장치 추상화 (Device Abstraction)
제조사의 의존성 없이 동종의 로봇 부속 장치 제어가 가능하도록 하는 작업

3.2 공통 로봇 인터페이스 (Common Robot Interface)
개방형 로봇 소프트웨어 플랫폼 상에서 로봇 부속 장치에 대한 추상화 인터페이스, 공통 로봇 API

3.3 장치 API (Device Application Programming Interface)
장치 API는 원하는 작업을 수행할 수 있도록 설계된 하드웨어 장치들을 위한 API

3.4 액추에이터 장치 (Actuator Device)
액추에이터는 전기, 유압, 공기압 등을 이용하여 로봇을 움직이게 하는 장치

3.5 액추에이터 기저 인터페이스 (Actuator Base Interface)
구동기 장치들에 공통으로 최소한 적용되어야 할 인터페이스

3.6 서보 액추에이터 (Servo Actuator)

속도와 위치 제어가 가능한 장치

3.7 조작 제어기 (Manipulator Controller)
다수의 자유도를 가진 로봇 팔의 구동부

3.8 이동 제어기 (Mobile Controller)
다수의 자유도를 가진 로봇 이동 장치의 구동부

3.9 센서 장치 (Sensor Device)
센서는 물리량을 측정하는 장치

3.10 센서 기저 인터페이스 (Sensor Base Interface)
입출력 장치는 신호를 입출력할 수 있는 장치

3.11 가속도 센서 (Acceleration Sensor)
이동하는 물체의 충격이나 가속도를 측정하는 장치

3.12 범퍼 터치 센서 (Bumper Touch Sensor)
눌림을 감지하여 상태정보를 반환하는 장치

- 비고 범퍼 센서 또는 터치 센서 모두에 적용

3.13 자이로 센서 (Gyro Sensor)
각속도를 측정하기 위한 장치

3.14 관성 측정 장치 (Inertial Measurement Unit Device)
이동관성을 측정할 수 있는 가속도계와 회전관성을 측정할 수 있는 자이로계, 방위각을 측정할 수 있는 자이로계, 방위각을 측정할 수 있는 자계로 이루어진 하나의 통합 장치

3.15 근접센서 (Proximity Sensor)
거리를 측정할 수 있는 장치

- 비고 적외선 센서 또는 초음파 센서 모두에 적용

3.16 레이저 거리 측정기 (Laser Range Finder)
일정한 각도 범위 내에서 다수의 포인트로 거리를 측정하는 장치

3.17 위치센서 (Position Sensor)
기준점으로부터 로봇까지의 위치를 측정하는 장치

3.18 지자기 센서 (Geo-magnetic Sensor)

자기장의 세기를 측정하기 위하여 X, Y, Z 방향으로 3개가 부착된 장치

3.19 카메라 (Camera)

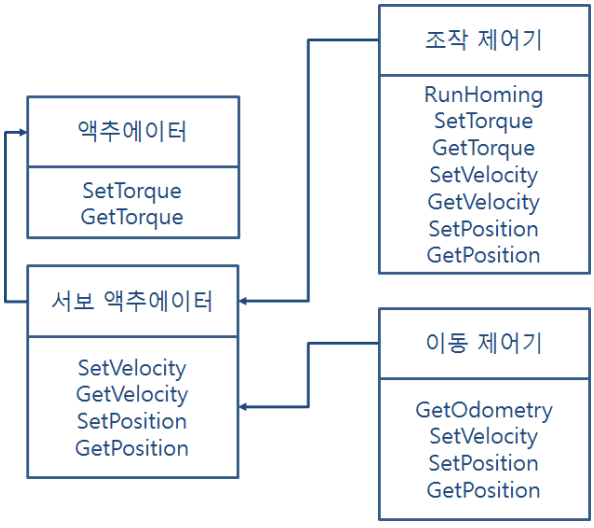
하나의 영상신호를 획득하는 장치

- 비고 스테레오 카메라 장치 제외

4 공통 로봇 인터페이스 개요

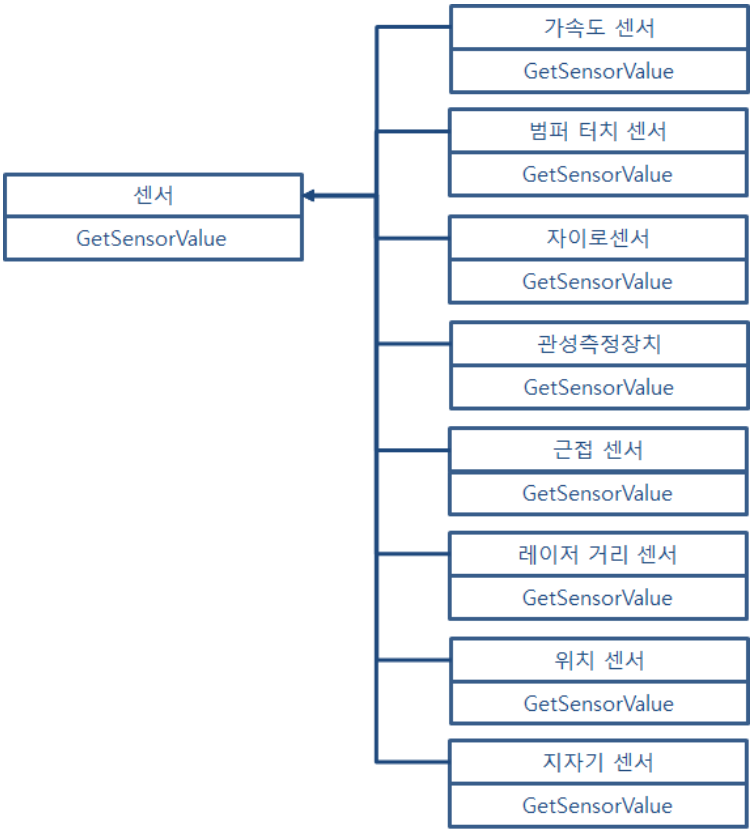
4.1 개요

OPRoS 장치를 위해 사용되는 OPRoS 장치 API는 OPRoS 기반 장치를 위한 표준 API이다. OPRoS에서 장치를 제어하기 위한 API는 다음 공통 장치 함수로 구성되어 있으며, 이들은 다음과 같다: API의 초기화, API 종료, API 활성화, API 비활성화, 속성 설정 및 속성 읽기에 관한 함수로 구성되어 있으며, 속성으로는 장치의 동작에 필요한 내용 혹은 값들 혹은 그 장치를 활용하고자 알고리즘을 위한 값들을 가지고 있다.



(그림 4-1) 구동기 인터페이스 상속 관계

액추에이터 기저 인터페이스와 구동기 장치의 인터페이스 상속관계는 (그림 4-1)과 같다.



(그림 4-2) 센서 인터페이스의 상속관계

센서 기저 인터페이스와 각 센서들의 인터페이스의 상속 관계는 (그림 4-2)와 같다.

4.2 속성

OPRoS 장치 API는 Property 타입으로 정의된 속성인 props를 포함한다. 속성은 문자열 타입의 인덱스와 값의 쌍으로 이루어진 클래스 형태의 데이터 타입이다. 사용자는 parameter에서 “size", "1"와 같은 형태로 데이터를 저장하고, "size"라는 인덱스를 이용하여 ”1“을 얻을 수 있다. ”1“을 정수형으로 사용하고 할 때는 atoi와 같은 함수를 이용하여 문자열을 숫자로 변환하여 사용하면 된다. props에는 API의 종류에 따라 정의된 인덱스가 포함되어야 하며, SetProperty 인터페이스를 호출할 때 정의된 인덱스가 없을 경우 에러를 반환한다.

Attribute	Description
Property props	장치의 동작에 필요한 정보를 가지고 있는 변수

4.3 Return Type

OPRoS 컴포넌트에서 사용되는 리턴 데이터 값이다. 아래의 값의 사용처는 컴포넌트의 GetError() 및 장치 API의 호출 결과와 OPRoS 도메인에서 사용되는 변수인 status의 값으로 사용된다.

Name	Value	Description
OPRoS 도메인에서 정의한 데이터 형에서 사용하는 변수 status의 값		
OPROS_ERROR	302	데이터상에 오류가 있는 경우
OPROS_VALIDATA	300	유효한 데이터가 존재하는 경우
OPROS_INVALIDATA	301	유효하지 않은 데이터가 존재하는 경우
공통적으로 활용되는 값		
OPROS_SUCCESS	200	동기 호출 성공
UNKNOWN_ERROR	210	알려지지 않은 에러
OPRoS 엔진에서 활용되는 값		
OPROS_SUCCESS_SYNC	202	비동기 호출 성공, 처리 중
OPROS_CALLER_ERROR	400	일반적인 호출자 오류
OPROS_BAD_INPUT_PARAMETER	401	입력 오류
OPROS_INPUT_OUT_OF_RANGE	402	입력 범위 초과
OPROS_INPUT_NULL	403	입력에서 NULL레퍼런스 포함
OPROS_UNAUTHORIZED	404	권한 오류
OPROS_PRECONDITION_NOT_MET	405	적절치 못한 함수 호출 순서
OPROS_CALLEE_ERROR	500	일반적인 피호출자 오류
OPROS_UNSUPPORTED_METHOD	501	현재 인터페이스에는 있으나, 지원하지 않음
OPROS_INTERNAL_FAULT	502	내부 처리 도중 문제 발생
OPROS_OUT_OF_RESOURCES	408	시스템 리소스 부족으로 처리 불가
OPROS_VERSION_MISMATCH	505	버전 다름
장치 API에서 정의된 값		
OPROS_DESTORY_ERROR	0x10000000	복구할 수 없는 에러
OPROS_FIND_DLL_ERROR	0x10000001	DLL 파일을 찾을 수 없는 경우
OPROS_LOAD_DLL_ERROR	0x10000002	DLL에서 제공하는 인터페이스를 로드할 수 없는 경우
OPROS_FIND_PROPERTY_ERROR	0x10000003	컴포넌트에서 필요로 하는 속성 값을 찾을 수 없는 경우
OPROS_INITIALIZE_API_ERROR	0x10000100	API를 초기화 시키는 도중 에러가 발생한 경우
OPROS_FINALIZE_API_ERROR	0x10000101	API를 종료 시키는 도중 에러가 발생한 경우
OPROS_RESET_API_ERROR	0x10000102	API를 재기동 시키는 도중 에러가 발생한 경우
OPROS_RECOVER_API_ERROR	0x10000103	API를 복구시키는 도중 에러가 발생한 경우
OPROS_RECOVER_ERROR	0x20000000	복구할 수 있는 에러
OPROS_ENABLE_API_ERROR	0x20000100	API를 활성화 시키는 도중 에러가 발생한 경우

OPROS_DISABLE_API_ERROR	0x20000101	API를 비활성화 시키는 도중 에러가 발생한 경우
OPROS_CALL_API_ERROR	0x20000102	API의 인터페이스를 호출하는 도중 에러가 발생한 경우
OPROS_API_NOT_SUPPORTED_ERROR	0x20000103	API에서 인터페이스를 지원하지 않는 경우
OPROS_API_EXECUTING	0x20000104	현재 API를 수행중인 경우, 예를 들어 머니퓰레이터가 초기위치로 이동 중인 경우
OPROS_API_SUCCESS	200	동기 호출 성공
OPROS_API_ERROR	302	데이터상에 오류가 있는 경우
API_SUCCESS	200	동기 호출 성공
API_NOT_SUPPORTED	0x20000103	API에서 인터페이스를 지원하지 않는 경우
API_ERROR	302	데이터상에 오류가 있는 경우
API_EXECUTING	0x20000104	현재 API를 수행중인 경우, 예를 들어 머니퓰레이터가 초기위치로 이동 중인 경우

4.4 데이터 타입

데이터 형을 명확하기 위해서 일반적으로 사용되는 데이터 형을 OPRoS에서는 다음과 같이 사용한다. 예를 들면 32비트 정수형을 사용하는 CPU를 가진 경우의 예는 다음과 같다.

```
typedef              char     char_t;
typedef signed       char     int8_t;
typedef signed       short    int16_t;
typedef signed       int      int32_t;
typedef signed       long     int64_t;
typedef unsigned     char     uint8_t;
typedef unsigned     short    uint16_t ;
typedef unsigned     int      uint32_t;
typedef unsigned     long     uint64_t;
typedef              float    float32_t;
typedef              double   float64_t;
typedef long                  double float128_t;
```

본 문서에서 정의한 데이터 타입은 OPRoS라는 네임스페이스 안에서 사용 가능하다.

4.4.1 기본 형태

C++에서 기본적으로 제공하는 데이터 타입으로 변수와 이 변수의 값이 유효한지 여부와

에러 여부를 확인할 수 있는 변수로 이루어진 클래스 형태의 데이터 형식이다.

Class	Name	Data Type	Description
Bool	status	ReturnType	데이터의 에러 및 유효 여부
	data	bool	bool 타입의 데이터 변수
Char	status	ReturnType	데이터의 에러 및 유효 여부
	data	char_t	char 타입의 데이터 변수
Int8	status	ReturnType	데이터의 에러 및 유효 여부
	data	int8_t	char 타입의 데이터 변수
UInt8	status	ReturnType	데이터의 에러 및 유효 여부
	data	uint8_t	unsigned char 타입의 데이터 변수
Int16	status	ReturnType	데이터의 에러 및 유효 여부
	data	int16_t	short 타입의 데이터 변수
UInt16	status	ReturnType	데이터의 에러 및 유효 여부
	data	uint16_t	unsigned short 타입의 데이터 변수
Int32	status	ReturnType	데이터의 에러 및 유효 여부
	data	int32_t	int 타입의 데이터 변수
UInt32	status	ReturnType	데이터의 에러 및 유효 여부
	data	uint32_t	uint32_t 타입의 데이터 변수
Int64	status	ReturnType	데이터의 에러 및 유효 여부
	data	int64_t	long 타입의 데이터 변수
UInt64	status	ReturnType	데이터의 에러 및 유효 여부
	data	uint64_t	unsigned long 타입의 데이터 변수
Float32	status	ReturnType	데이터의 에러 및 유효 여부
	data	float32_t	float 타입의 데이터 변수
Float64	status	ReturnType	데이터의 에러 및 유효 여부
	data	float64_t	double 타입의 데이터 변수
Float128	status	ReturnType	데이터의 에러 및 유효 여부
	data	float128_t	double 타입의 데이터 변수
String	status	ReturnType	데이터의 에러 및 유효 여부
	data	std::string	string 타입의 데이터 변수

4.4.2 Array Type

기본 형태에서 데이터가 배열형태로 되어 있는 데이터 타입이다. 배열은 STL(Standard Template Library)의 vector를 사용한다.

Class	Name	Data Type	Description
BoolArray	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<bool>	bool 타입의 데이터 변수
CharArray	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<char_t>	char 타입의 데이터 변수
Int8Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<int8_t>	signed char 타입의 데이터 변수
UInt8Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<uint8_t>	unsigned char 타입의 데이터 변수
Int16Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<int16_t>	short 타입의 데이터 변수
Uint16Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<uint16_t>	unsigned short 타입의 데이터 변수
Int32Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<int32_t>	int 타입의 데이터 변수
UInt32Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<uint32_t>	uint32_t 타입의 데이터 변수
Int64Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<int64_t>	long 타입의 데이터 변수
UInt64Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<uint64_t>	unsigned long 타입의 데이터 변수
Float32Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<float32_t>	float 타입의 데이터 변수
Float64Array	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<float64_t>	double 타입의 데이터 변수
StringArray	status	ReturnType	데이터의 에러 및 유효 여부
	data	vector<std::string>	string 타입의 데이터 변수

4.4.3 Class Type

4.4.3.1 속성

속성은 문자열 형식의 이름과 값으로 이루어진 데이터 형식이다. 내부적으로 map<string, string>으로 구현되어 있다. SetValue 함수를 이용하여 이름과 값을 설정할 수 있고, GetValue 함수를 이용하여 이름에 해당되는 값을 얻을 수 있다. FindName 함수를 이용하여 해당되는 이름이 존재하는지 여부를 확인한다. SetProperty 함수를 이용하여 내부의 map<string, string> 데이터를 설정할 수 있다. map<string, string> 형태의

데이터는 내부함수인 getProperty()와 setProperty(), getPropertyMAP()을 통하여 얻을 수 있다.

Name	Data Type	Description
status	ReturnType	상태 정보
data	map<std::string, std:: string>	인덱스와 데이터가 저장되는 변수
Function(s)		
void SetProperty(map<string, string> data)		Property의 전체 데이터를 설정한다. 서비스포트를 통한 서비스 메소드로 제공함.
map<string, string> GetProperty(void)		Property의 전체 데이터를 반환한다. 서비스포트를 통한 서비스 메소드로 제공
void SetValue(string name, string value)		name이라는 이름으로 value라는 값을 저장한다. 서비스포트를 통한 서비스 메소드로 제공하지 않음
string GetValue(string name)		name에 해당되는 값을 얻는다. 해당되는 name이 없을 경우 빈 문자열을 반환한다. 서비스포트를 통한 서비스 메소드로 제공하지 않음
bool FindName(string name)		name이라는 이름이 존재하는지 여부를 반환한다. true가 반환되면 해당되는 이름이 존재하는 것이고, false가 반환되면 해당되는 이름이 존재하지 않는다는 것이다. 서비스포트를 통한 서비스 메소드로 제공하지 않음

4.4.3.2 ObjectPosition

로봇(모바일 플랫폼 포함)을 포함한 장치들의 3차원 위치를 표현하기 위한 데이터 타입이다. 장치들의 위치는 로봇 원점을 기준으로 한다. 로봇(모바일 플랫폼 포함)의 위치는 월드좌표계를 기준으로 한다. parameter 정의에 사용됨

Name	Data Type	Description
x	float64_t	X축 방향의 위치[meter]
y	float64_t	Y축 방향의 위치[meter]
z	float64_t	Z축 방향의 위치[meter]
roll	float64_t	X축을 중심으로한 각도[degree]
pitch	float64_t	Y축을 중심으로한 각도[degree]
yaw	float64_t	Z축을 중심으로한 각도[degree]

4.4.3.3 ObjectPositionData

위치 추정 장치들로부터 로봇의 위치를 받기 위한 정보이다. 로봇(모바일 플랫폼 포함)의 위치는 월드좌표계를 기준으로 한다. ObjectPosition 데이터 형에서 status를 추가한 것이다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
x	float64_t	X축 방향의 위치[meter]
y	float64_t	Y축 방향의 위치[meter]
z	float64_t	Z축 방향의 위치[meter]
roll	float64_t	X축을 중심으로한 각도[degree]
pitch	float64_t	Y축을 중심으로한 각도[degree]
yaw	float64_t	Z축을 중심으로한 각도[degree]

4.4.3.4 ObjectVelocityData

장치의 속도로 사용하고자 할 경우 아래의 표와 같다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
x	float64_t	X축 방향의 속도[meter/sec]
y	float64_t	Y축 방향의 속도[meter/sec]
z	float64_t	Z축 방향의 속도[meter/sec]
roll	float64_t	X축을 중심으로한 각속도[degree/sec]
pitch	float64_t	Y축을 중심으로한 각속도[degree/sec]
yaw	float64_t	Z축을 중심으로한 각속도[degree/sec]

4.4.3.5 ObjectAccelerationData

장치의 가속도를 사용하고자 할 경우 아래의 표와 같다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
x	float64_t	X축 방향의 가속도[meter/sec^2]
y	float64_t	Y축 방향의 가속도[meter/sec^2]
z	float64_t	Z축 방향의 가속도[meter/sec^2]
roll	float64_t	X축을 중심으로한 각가속도[degree/sec^2]
pitch	float64_t	Y축을 중심으로한 각가속도[degree/sec^2]
yaw	float64_t	Z축을 중심으로한 각가속도[degree/sec^2]

4.4.3.6 MobilePositionData

모바일 로봇의 2차원 좌표를 표현하기 위한 데이터 타입이다. 이 타입을 이용하여 모바일 로봇의 위치를 표현할 수 있다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
x	float64_t	X축 방향의 위치[meter]
y	float64_t	Y축 방향의 위치[meter]
theta	float64_t	Z축을 중심으로한 각도[degree]

4.4.3.7 MobileVelocityData

모바일 로봇의 2차원 좌표를 표현하기 위한 데이터 타입이다. 이 타입을 이용하여 모바일 로봇의 속도를 표현할 수 있다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
x	float64_t	X축 방향의 속도[meter/sec]
y	float64_t	Y축 방향의 속도[meter/sec]
theta	float64_t	Z축을 중심으로한 각속도[degree/sec]

4.4.3.8 MobileAccelerationData

모바일 로봇의 2차원 좌표를 표현하기 위한 데이터 타입이다. 이 타입을 이용하여 모바일 로봇의 가속도를 표현할 수 있다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
x	float64_t	X축 방향의 가속도[meter/sec^2]
y	float64_t	Y축 방향의 가속도[meter/sec^2]
theta	float64_t	Z축을 중심으로한 각가속도[degree/sec^2]

4.4.3.9 AccelerationSensorData

가속도 센서의 데이터를 저장하기 위한 데이터 타입이다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
xAcceleration	float64_t	X축 가속도
yAcceleration	float64_t	Y축 가속도
zAcceleration	float64_t	Z축 가속도

4.4.3.10 GeoMagneticData

지자기 센서의 데이터를 저장하기 위한 데이터 타입이다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
xGauss	float64_t	X축 방향의 지자기값
yGauss	float64_t	Y축 방향의 지자기값
zGauss	float64_t	Z축 방향의 지자기값
xGaussRate	float64_t	X축 방향의 지자기값 단위
yGaussRate	float64_t	Y축 방향의 지자기값 단위
zGaussRate	float64_t	Z축 방향의 지자기값 단위
xAlpha	float64_t	X축 지자기 보정값
yAlpha	float64_t	Y축 지자기 보정값
zAlpha	float64_t	Z축 지자기 보정값

4.4.3.11 GpsData

GPS 데이터를 저장하기 위한 데이터 타입이다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
latitude	float64_t	위도
longitude	float64_t	경도
time	std::string	시간
velocity	float64_t	속도

4.4.3.12 GyroSensorData

자이로 센서의 데이터를 저장하기 위한 데이터 타입이다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
xAngle	float64_t	X축을 중심으로한 각도
yAngle	float64_t	Y축을 중심으로한 각도
zAngle	float64_t	Z축을 중심으로한 각도
xAngleRate	float64_t	X축을 중심으로한 각속도
yAngleRate	float64_t	Y축을 중심으로한 각속도
zAngleRate	float64_t	Z축을 중심으로한 각속도

4.4.3.13 ImuData

관성측정장치의 데이터를 저장하기 위한 데이터 타입이다

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
pRate	float64_t	Roll축 각속도
qRate	float64_t	Pitch축 가속도
rRate	float64_t	Yaw축 가속도
xAcceleration	float64_t	X축 가속도
yAcceleration	float64_t	Y축 가속도
zAcceleration	float64_t	Z축 가속도
roll	float64_t	장치의 Roll 좌표
pitch	float64_t	장치의 Pitch 좌표
yaw	float64_t	장치의 Yaw 좌표

4.4.3.14 CameraData

카메라의 영상을 저장하기 위한 데이터 타입이다.

Name	Data Type	Description
status	ReturnType	데이터의 에러 및 유효 여부
format	int32_t	이미지의 종류 0: Gray 1: YCbCr 2: RGB 3: JPEG 4 : PNG
width	int32_t	가로 해상도
height	int32_t	세로 해상도
data	vector<uint8_t>	이미지 데이터

4.5 공통 장치 API

4.5.1 Initialize

인자로 넘어가는 속성 값들을 활용하여 장치 및 API 함수들을 초기화한다.

Declaration	int32_t Initialize(Property props)
Arguments	props : 장치 및 API의 동작 혹은 초기화에 필요한 정보
Return Value	API_SUCCESS : 성공적으로 API를 초기화한 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우

4.5.2 Finalize

장치 및 API를 종료시킨다. 장치 및 API가 종료될 때 API가 비활성화 되어 있지 않은 경우 Disable 인터페이스를 호출한 후에 API를 종료시킨다. 이 인터페이스 호출 후 에러가 반환되면 프로그램을 종료시켜야 한다.

Declaration	int32_t Finalize(void)
Arguments	없음
Return Value	API_SUCCESS : 성공적으로 API를 종료한 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우 API_ERROR : API를 종료하는 도중 에러가 발생한 경우

4.5.3 Enable

API를 활성화 시킨다. 그러나 장치 및 API가 초기화 되지 않은 상태에서 이 인터페이스를 호출하면 에러를 반환한다.

Declaration	int32_t Enable(void)
Arguments	없음
Return Value	API_SUCCESS : 성공적으로 API를 활성화 시킨 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우 API_ERROR : API를 활성화 시키는 도중 에러가 발생한 경우

4.5.4 Disable

API를 비활성화 시킨다. 장치 및 API가 초기화 되지 않은 상태에서 이 인터페이스를 호

출하면 에러를 반환한다. API가 비활성화 상태인 경우에는 API_SUCCESS를 반환한다.

Declaration	int32_t Disable(void)
Arguments	없음.
Return Value	API_SUCCESS : 성공적으로 API를 비활성화 시킨 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우 API_ERROR : API를 비활성화 시키는 도중 에러가 발생한 경우

4.5.5 SetProperty

SetProperty는 장치와 API가 초기화된 이후에 장치 및 API의 속성을 변경할 때 사용하는 인터페이스이다. 이 함수가 호출될 때 API가 활성화 상태이면 API를 비활성화 시킨 후에 파라미터를 설정한다. 설정이 완료되면 다시 API를 활성화 시켜야 한다.

Declaration	int32_t SetProperty(Property props)
Arguments	props : API의 속성값들을 저장한 변수
Return Value	API_SUCCESS : 성공적으로 API를 속성을 변경한 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우 API_ERROR : 속성을 변경하는 도중 에러가 발생한 경우

4.5.6 GetProperty

API의 파라미터를 읽는다.

Declaration	int32_t GetProperty(Property&props)
Arguments	props : API의 속성 값들이 저장될 변수
Return Value	API_SUCCESS : 성공적으로 API를 속성을 읽어온 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우 API_ERROR : 속성을 얻어 오는 도중 에러가 발생한 경우

4.6 사용 방법

4.6.1 장치 및 API초기화

Initialize 함수를 호출하여 장치 및 API를 초기화 시킨다. 이 함수에서 오류가 생기면 API_ERROR를 반환한다.

4.6.2 API 활성화

Enable 함수를 호출하여 API를 활성화 시킨다. 이 함수가 호출되기 전에 Initialize 함수가 호출되지 않았다면 API_ERROR를 반환한다.

4.6.3 API 비활성화

Disable 함수를 호출하여 API를 비활성화 시킨다. 이 함수가 호출되기 전에 Initialize 함수가 호출되지 않았다면 API_ERROR를 반환한다.

4.6.4 API 종료

Finalize 함수를 호출하여 API 및 장치를 종료시킨다. 이 함수가 호출되기 전에 Disable 함수가 호출되지 않은 경우 내부적으로 Disable 함수를 호출한 후 Finalize 함수의 다른 기능이 동작한다. 이 함수에서 오류가 생기면 API_ERROR를 반환한다.

4.6.5 장치 및 API의 속성 변경

SetProperty 함수를 호출하여 장치 및 API의 속성을 변경한다. API가 활성화되었을 경우 Disable 함수를 호출하여 비활성화 시킨 후 속성을 설정한다. 파라미터 설정이 완료되면 Enable 함수를 호출하여 API를 활성화 시킨다. 이 함수에서 오류가 생기면 API_ERROR를 반환한다.

4.6.6 장치 및 API의 속성 읽기

GetProperty 인터페이스를 호출하여 장치 및 API의 속성 값을 읽는다. 이 함수에서 오류가 생기면 API_ERROR를 반환한다.

5 액추에이터 API

액추에이터는 로봇의 팔, 바퀴와 같이 구동부를 제어하는 장치이다. 액추에이터는 일반적으로 사용되는 모터뿐만 아니라 유압제어장치, 공압제어장치도 포함한다. 액추에이터는 동작 형태에 따라 회전운동형태와 직선운동형태으로 분류된다. 일반적으로 로봇에서는 액추에이터만을 사용하지 않고 액추에이터와 감속기(기어, 하모닉 드라이브 등)의 조합으로 사용된다. 때문에 액추에이터에는 감속기 정보가 포함되어 있어야 한다. 액추에이터는 출력(토크, 힘 등)만을 제어할 수 있는 액추에이터와 출력 뿐만 아니라 속도 및 위치를 제어할 수 있는 액추에이터로 구분된다. 본 문서에서는 출력만을 제어할 수 있는 장치는 액추에이터, on-off 액추에이터, 속도 및 위치를 제어할 수 있는 장치를 서보 액추에이터로 분류하고, 동작 단위는 회전운동형태의 경우 degree와 second 단위를 사용하고 직

선운동형태의 경우 meter와 second 단위를 사용한다.

5.1 액추에이터 기저 API

5.1.1 개요

로봇을 움직이기 위해서는 액추에이터를 제어해야한다. 액추에이터에는 여러 가지 종류가 있고 제어방법도 여러 가지가 있다. 본문에서는 액추에이터를 제어하기 위해 출력을 제어하는 인터페이스를 설명한다. 파라미터로는 액추에이터의 최대 출력과 출력 값의 단위가 필요하다.

5.1.2 속성

액추에이터 기저 API는 장치 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
ActuatorType	uint32_t	액추에이터 형태 0 : Revolute Actuator 1 : Prismatic Actuator
OutputType	uint32_t	0 : Nm 또는 N 1 : 출력비(%)

5.1.3 인터페이스

5.1.3.1 SetTorque

액추에이터의 출력(힘과 토크)을 설정한다. 출력단위는 액추에이터의 형태와 출력형태에 따라 달라진다. ActuatorType이 0이고, OutputType이 0인 경우 Nm이다.

Declaration	int32_t SetTorque(OPRoS::Float64 torque)
Arguments	torque : 액츄에이터의 출력 값
Return Value	API_SUCCESS : 성공적으로 액츄에이터의 출력을 설정한 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 혹은 액츄에이터의 출력을 제어하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 해당 API를 지원하지 않는 경우

5.1.3.2 GetTorque

액추에이터의 현재 출력을 얻어온다. 출력단위는 각 타입에 따라 달라질 수 있다. 회전 운동의 경우 토크로써 Nm이다.

Declaration	int32_t GetTorque(OPRoS::Float64 &power)
Arguments	torque : 액츄에이터의 출력 값이 저장될 변수
Return Value	API_SUCCESS : 성공적으로 액츄에이터의 출력을 읽어온 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 혹은 액츄에이터의 출력을 제어하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 해당 API를 지원하지 않는 경우

5.1.4 사용방법

액추에이터 기저 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

5.1.4.1 액추에이터의 출력제어

액추에이터 관련 장치 및 API가 활성화된 상태에서 SetTorque 함수를 호출한다. 장치 및 API가 활성화되지 않았다면 에러를 반환한다.

5.1.4.2 액추에이터 출력 상태

액추에이터 기저 API가 활성화된 상태에서 GetTorque 함수를 호출한다. 이 장치 및 API가 활성화되지 않았다면 에러를 반환한다.

5.2 서보 액추에이터 API

5.2.1 개요

서보 액추에이터 API는 모터 등의 제어를 위한 모터드라이버 등과 같은 구동부를 위한 인터페이스이다. 모든 서보 액추에이터는 서보 액추에이터 API를 이용해야 한다. 액추에이터는 Torque만으로 제어하는 구동부인 반면에, 서보 액추에이터는 속도제어 위치제어도 가능한 구동부이다. SetProprty 함수가 호출되면 내부적으로 서보 액추에이터의 최대 출력, 최대속도, 가속도, 위치 제한 등의 속성 값을 설정해야 한다. 서보 액추에이터 API는 액추에이터 기저 API를 상속받는다.

5.2.2 속성

서보 액추에이터 API는 액추에이터 기저 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
ActuatorType	uint32_t	액추에이터 형태 0 : Revolute Actuator 1 : Prismatic Actuator
OutputType	uint32_t	0 : Nm 또는 N 1 : 출력비(%)

5.2.3 인터페이스

5.2.3.1 SetVelocity

서보 액추에이터의 속도를 제어한다. Actuatorype이 0 인 경우 단위가 degree/s이고 1 인 경우 단위가 m/s이다.

Declaration	int32_t SetVelocity(OPRoS::Float64 velocity)
Arguments	velocity : 서보 액추에이터의 속도
Return Value	API_SUCCESS : 성공적으로 서보 액추에이터의 속도를 설정한 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 혹은 액츄에이터의 속도를 제어하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 해당 API를 지원하지 않는 경우

5.2.3.2 GetVelocity

서보 액추에이터의 속도를 얻어온다. ActuatorType이 0 인 경우 단위가 degree/s이고, 1 인 경우 단위가 m/s이다.

Declaration	int32_t GetVelocity(OPRoS::Float64 &velocity)
Arguments	velocity : 서보 액추에이터의 속도가 저장될 변수
Return Value	API_SUCCESS : 성공적으로 서보 액추에이터의 속도를 읽어 온 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 혹은 액츄에이터의 속도를 얻거나 출력을 제어하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 해당 API를 지원하지 않는 경우

5.2.3.3 SetPosition

서보 액추에이터의 위치를 제어한다. ActuatorType이 0 인 경우 단위가 degree이고, 1인 경우 단위가 m이다.

Declaration	int32_t SetPosition(OPRoS::Float64 position, uint32_t time)
Arguments	position : 서보 액추에이터의 위치 time : 현 위치에서 지정된 위치로 이동하는데 소요되는 이동 시간이며 단위는 ms이다. 값이 0인 경우는 time을 사용하지 않음
Return Value	API_SUCCESS : 성공적으로 서보 액추에이터의 위치를 설정한 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 혹은 액츄에이터의 위치를 제어하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 해당 API를 지원하지 않는 경우

5.2.3.4 GetPosition

서보 액추에이터의 위치를 읽어온다. ActuatorType이 0인 경우 단위가 각도(degree)이고, 1인 경우 단위가 거리(m)이다.

Declaration	int32_t GetPosition(OPRoS::Float64 &position)
Arguments	position : 서보 액추에이터의 현재 위치가 저장될 변수
Return Value	API_SUCCESS : 성공적으로 서보 액추에이터의 현재 위치를 읽어 온 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 혹은 액츄에이터의 위치를 제어하거나 읽어오는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 해당 API를 지원하지 않는 경우

5.2.4 사용 방법

서보 액추에이터 API 중 액추에이터 API에서 상속받은 인터페이스는 액추에이터 API와 동일한 방법으로 사용된다.

5.2.4.1 서보 액추에이터의 속도제어

SetVelocity 인터페이스를 호출하여 서보 액추에이터의 속도를 제어할 수 있다. 현재 속도를 확인하기 위해서는 GetVelocity 인터페이스를 호출하면 된다. 서보 액추에이터 API가 활성화 되어 있지 않은 경우 에러를 반환한다.

5.2.4.2 서보 액추에이터의 위치제어

SetPosition 인터페이스를 호출하여 서보 액추에이터의 위치를 제어할 수 있다. 현재 위치를 확인하기 위해서는 GetPosition 인터페이스를 호출하면 된다. ServoActuator API가

활성화 되지 않는 경우 에러를 반환한다.

5.3 조작 제어기 API

5.3.1 개요

조작부는 다수의 자유도를 가진 구동장치이다. 일반적으로 로봇의 팔을 이루는 주된 장치가 된다. 조작 제어기 인터페이스는 2개 이상의 액추에이터가 조합된 장치를 위한 인터페이스이다.

5.3.2 속성

조작 제어기 API는 장치 API의 속성을 상속받아 사용하며, 속성의 인덱스는 다음과 같다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
NumberOfActuators	uint32_t	머니퓰레이터의 액추에이터 개수
ActuatorType0	uint32_t	액추에이터 형태 0 : Revolute Actuator 1 : Prismatic Actuator
OutputType0	uint32_t	0 : Nm 또는 N 1 : Rate(%)

5.3.3 인터페이스

5.3.3.1 RunHoming

조작 제어기의 각 조인트들을 초기위치로 이동시킨다.

Declaration	int32_t RunHoming(void)
Arguments	없음
Return Value	API_SUCCESS : 머니퓰레이터의 각 조인트들이 초기위치로 이동한 경우 API_EXECUTING : 현재 API를 수행중인 경우, 여기서는 머니퓰레이터의 각 조인트들이 초기 위치로 이동중인 경우 API_ERROR : 머니퓰레이터의 각 조인트들이 초기위치로 이동하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

5.3.3.2 SetTorque

조작 제어기의 출력을 각 조인트별로 제어한다. torque의 단위는 ActuatorType과 OutputType에 의해서 결정된다.

Declaration	int32_t SetTorque(std::vector<OPRoS::Float64> torque)
Arguments	torque : 액추에이터로 지정된 출력 값을 유지한다.
Return Value	API_SUCCESS : 성공적으로 머니퓰레이터의 출력을 설정한 경우 API_ERROR : 머니퓰레이터의 출력을 제어하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

5.3.3.3 GetTorque

조작 제어기의 조인트별 출력을 읽어온다.

Declaration	int32_t GetTorque(std::vector<OPRoS::Float64> &torque)
Arguments	torque : 액추에이터의 출력 값이 저장될 변수
Return Value	API_SUCCESS : 성공적으로 머니퓰레이터의 출력을 읽어온 경우 API_ERROR : 머니퓰레이터의 출력을 읽는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

5.3.3.4 SetVelocity

조작 제어기의 조인트별 속도를 제어한다. ActuatorType이 0(Revolute Actuator)인 경우 degree/s이고 1(Prismatic Actuator)의 경우 m/s이다.

Declaration	int32_t SetVelocity(std::vector<OPRoS::Float64> velocity)
Arguments	velocity : 서보 액추에이터의 속도
Return Value	API_SUCCESS : 성공적으로 머니퓰레이터의 속도를 설정한 경우 API_ERROR : 머니퓰레이터의 속도를 설정하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

5.3.3.5 GetVelocity

조작 제어기의 조인트별 속도를 읽어온다. 단위는 SetVelocity 인터페이스와 동일하다.

Declaration	int32_t GetVelocity(std::vector<OPRoS::Float64> &velocity)
Arguments	velocity : 서보 액추에이터의 속도가 저장될 변수
Return Value	API_SUCCESS : 성공적으로 머니퓰레이터의 속도를 읽어 온 경우 API_ERROR : 머니퓰레이터의 속도를 읽어오는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

5.3.3.6 SetPosition

조작 제어기의 조인트별 위치를 제어한다. ActuatorType이 0(Revolute Actuator)인 경우 degree이고 1(Prismatic Actuator)의 경우 m이다.

Declaration	int32_t SetPosition(std::vector<OPRoS::Float64> position, std::vector<uint32_t> time)
Arguments	position : 각 서보 액추에이터의 위치를 표시하는 벡터 time : 각 서보 액추에이터의 이동시간을 표시하는 벡터이며 단위는 ms이며, 값이 0인 경우는 time을 사용하지 않음. 예를 들면, 머니퓰레이터의 각 조인트가 현 위치에서 지정된 위치로 이동하는데 소요되는 이동 시간들의 벡터이다.
Return Value	API_SUCCESS : 성공적으로 머니퓰레이터의 위치를 설정한 경우 API_ERROR : 머니퓰레이터의 위치를 설정하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

5.3.3.7 GetPosition

조작 제어기의 조인트별 위치를 읽어온다. 단위는 SetPostion 인터페이스와 동일하다.

Declaration	int32_t GetPosition(std::vector<OPRoS::Float64> &position)
Arguments	position : 머니퓰레이터의 현재 위치가 저장될 변수
Return Value	API_SUCCESS : 성공적으로 머니퓰레이터의 현재 위치를 읽어 온 경우 API_ERROR : 머니퓰레이터의 현재 위치를 읽는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

5.3.4 사용방법

조작 제어기 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

5.3.4.1 조작 제어기의 출력제어

조작 제어기 API가 활성화된 상태에서 SetPower 인터페이스를 호출하여 출력을 제어할 수 있다. 현재 출력을 확인하기 위해서는 GetPower 인터페이스를 호출하면 된다. 이 API가 활성화되지 않았다면 에러를 반환한다. 또한 index가 설정된 파라미터의 "size" 항목의 범위 안에 포함되지 않는다면 에러를 반환한다.

5.3.4.2 조작 제어기의 속도제어

조작 제어기 API가 활성화된 상태에서 SetVelocity 인터페이스를 호출하여 속도를 제어할 수 있다. 현재 속도를 확인하기 위해서는 GetVelocity 인터페이스를 호출하면 된다. 이 API가 활성화되지 않았다면 에러를 반환한다. 또한 index가 설정된 파라미터 “size" 항목의 범위 안에 포함되지 않는다면 에러를 반환한다.

5.3.4.3 조작 제어기의 위치제어

조작 제어기 API가 활성화된 상태에서 SetPosition 인터페이스를 호출하여 위치를 제어할 수 있다. 현재 위치를 확인하기 위해서는 GetPosition 인터페이스를 호출하면 된다. 이 API가 활성화되지 않았않았다면 에러를 반환한다. 또한 index가 설정된 파라미터 “size" 항목의 범위 안에 포함되지 않는다면 에러를 반환한다.

5.4 이동 제어기 API

5.4.1 개요

이동부는 바퀴 또는 궤도를 이용하여 이동이 가능한 장치를 의미한다.

5.4.2 속성

이동 제어기 API는 장치 API의 속성을 상속받아 사용하며, 속성의 인덱스는 다음과 같다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함

5.4.3 인터페이스

5.4.3.1 GetOdometry

이동 제어기의 장착된 각 바퀴의 센서 값을 반환하는 인터페이스이다.

Declaration	int32_t GetOdometry(std::vector<OPRoS::Int32> &odometry)	
Arguments	odometry : 모바일베이스의 센서 값들이 저장될 변수	
Return Value	API_SUCCESS	성공적으로 모바일베이스의 바퀴 센서값들을 얻어온 경우
	API_ERROR	모바일베이스의 바퀴 센서값들을 얻어오는 도중 에러가 발생한 경우
	API_NOT_SUPPORTED	API에서 이 인터페이스를 지원하지 않는 경우

5.4.3.2 SetVelocity

이동 제어기의 선속도와 각속도를 제어하는 인터페이스이다. linearVelocity의 단위는 m/s이고, angularVelocity의 단위는 degree/s이다.

Declaration	Int32_t SetVelocity(OPRoS::MobileVelocityData velocity)	
Arguments	velocity : 모바일의 이동속도 velocity.x : 모바일의 X 축 선속도 (+ : 전진, - 후진) velocity.y : 모바일의 Y 축 선속도 (+ : 왼쪽, - : 오른쪽) velocity.theta : 모바일의 각속도 (+ : 반시계. - : 시계) X,Y는 로봇 로컬좌표계기준	
Return Value	API_SUCCESS	성공적으로 휠 컨트롤러의 속도 명령을 전달한 경우
	API_ERROR	휠 컨트롤러의 속도 명령을 전달하는 도중 에러가 발생한 경우
	API_NOT_SUPPORTED	API에서 이 인터페이스를 지원하지 않는 경우

5.4.3.3 SetPosition

이동 제어기의 위치를 제어하는 인터페이스이다. MobilePositionData는 모바일 로봇의 2차원 좌표를 표현하기 위하여 OPRoS 도메인에서 정의된 데이터 타입이다. 단위는 meter와 degree을 사용한다.

Declaration	int32_t SetPosition(OPRoS::MobilePositionData position)	
Arguments	position : 설정하고자하는 모바일제어기의 위치 position.status : reserved position.x : X축 방향의 위치[meter] position.y : Y축 방향의 위치[meter] position.theta : Z축을 중심으로한 각도[degree] X,Y는 로봇 로컬좌표계기준	
Return Value	API_SUCCESS	성공적으로 휠 컨트롤러의 속도 명령을 전달한 경우
	API_ERROR	휠 컨트롤러의 속도 명령을 전달하는 도중 에러가 발생한 경우
	API_NOT_SUPPORTED	API에서 이 인터페이스를 지원하지 않는 경우

5.4.3.4 GetPosition

이동 제어기의 위치를 읽어오는 인터페이스이다. MobilePositionData는 모바일 로봇의 2차원 좌표를 표현하기 위하여 OPRoS 도메인에서 정의된 데이터 타입이다. 단위는 meter와 degree을 사용한다.

Declaration	int32_t SetPosition(OPRoS::MobilePositionData &position)	
Arguments	MobilePositionData : 모바일제어기의 위치 MobilePositionData.status : reserved MobilePositionData.x : X축 방향의 위치[meter] MobilePositionData.y : Y축 방향의 위치[meter] MobilePositionData.theta : Z축을 중심으로한 각도[degree] X,Y는 로봇 로컬좌표계기준	
Return Value	API_SUCCESS	성공적으로 휠 컨트롤러의 속도 명령을 전달한 경우
	API_ERROR	휠 컨트롤러의 속도 명령을 전달하는 도중 에러가 발생한 경우
	API_NOT_SUPPORTED	API에서 이 인터페이스를 지원하지 않는 경우

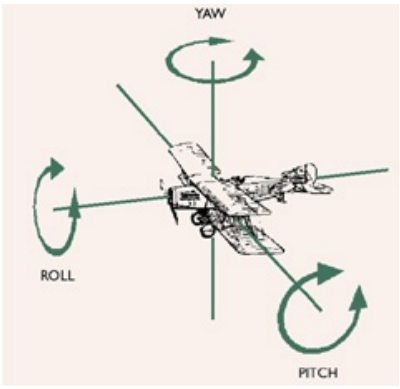
6 센서 API

6.1 센서기저 API

6.1.1 개요

센서는 외부로부터 측정된 아날로그 값을 받아들이는 장치이다. 본 문서에서는 초음파센서, 적외선 센서, 레이저 스캐너, 가속도센서, 각속도센서, 관성측정장치, 힘센서, 토크센서, 범퍼센서, 터치센서 등의 API를 다룬다.
센서는 자신이 설치된 위치 정보를 설정해야 한다. 센서의 위치 정보는 x, y, z, roll,

pitch, yaw로 표현될 수 있다. x, y, z는 3차원 공간상의 좌표이고, roll, pitch, yaw는 아래의 그림과 같이 방향을 나타낸다.



각 센서 장치의 위치 정보는 ObjectPositionData이라는 데이터형으로 표시한다. 참고로 다음 표와 같다.

Name	Data Type	Description
status	ReturnType	
x	float64_t	장치의 X 좌표
y	float64_t	장치의 Y 좌표
z	float64_t	장치의 Z 좌표
roll	float64_t	장치의 Roll 좌표
pitch	float64_t	장치의 Pitch 좌표
yaw	float64_t	장치의 Yaw 좌표

6.1.2 속성

센서 기저 API는 장치 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
Count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를 표시
Type1	string	ObjSen1의 위치에 있는 장치의 종류
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
Type2	string	ObjSen2의 위치에 있는 장치의 종류
...	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
...	string	ObjSen i의 위치에 있는 장치의 종류
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시
TypeN	string	ObjSenN의 위치에 있는 장치의 종류
주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) 장치 종류인 Type의 예로는 “GYRO-1Axis”, “GYRO-2Axis”, “GYRO-3Axis”, “GPS”, “BUMPER”, “ULTRASONIC” 등으로 표시되며 이러한 문자열의 정의는 따로 정의할 필요가 있다.		

주) 로봇의 기준점과 홈잉 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다.

6.1.3 인터페이스

센서 기저 API의 인터페이스는 장치 API와 동일하다.

6.1.4 사용 방법

센서 기저 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

6.2 가속도센서 API

6.2.1 개요

가속도센서는 이동하는 물체의 충격이나 가속도를 측정하는 센서이다. 가속도 센서의 단위는 m/s20이다.

6.2.2 속성

가속도센서 API는 센서 기저 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type1	string	ObjSen1의 위치에 있는 장치의 종류
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type2	string	ObjSen2의 위치에 있는 장치의 종류
...	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
...	string	ObjSen i의 위치에 있는 장치의 종류
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시
typeN	string	ObjSenN의 위치에 있는 장치의 종류
주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) AccelerationSensor 장치 Type은 “ACC-1”(1축 가속도센서), “ACC-2”(2축 가속도센서), “ACC-3”(3축 가속도센서)으로 정의한다.		

6.2.3 인터페이스

6.2.3.1 GetSensorValue

가속도 센서의 모든 센서들의 측정된 가속도 값들을 읽는다. 이 때 가속도 센서의 Type에 따라 각 축의 가속도 값의 유효성이 달라진다. OPRoS::AccelerationSensorData의 변수 중 status 변수는 OPROS_VALIDATA(성공적으로 값들을 읽어 온 경우), OPROS_ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 OPROS_INVALIDATA(데이터가 검출되지 않는 경우)로 표시된다.

Declaration	int32_t GetSensorValue(vector<OPRoS::AccelerationSensorData> &sensorValue)
Arguments	sensorValue : 가속도 센서에서 측정된 가속도 값을 담을 변수
Return Value	API_SUCCESS : 성공적으로 API를 초기화한 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우

6.2.4 사용방법

가속도센서 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

6.2.4.1 가속도센서 값 읽기

가속도센서 API가 활성화된 상태에서 GetAccelerationSensorData 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

6.3 범퍼터치센서 API

6.3.1 개요

범퍼터치센서의 종류에는 터치 셈서와 범퍼센서가 있으며, 센서의 눌림에 의해 무엇이 있다는 것을 인지하는 센서이다. 예를 들면, 로봇이 이동 중에 센서가 눌리게 되면 이동 경로에 장애물이 있다는 인식을 하게 된다. 범퍼 혹은 터치 센서는 센서의 눌림의 유무에 따라 ON/OFF 방식으로 신호를 처리한다. 이러한 센서 유형으로는 터치 센서도 있음. 즉, 이 절은 범퍼 센서 및 터치 센서를 위한 API를 정의한다.

6.3.2 속성

범퍼 터치 센서 API는 센서 기저 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를 표시
Type1	string	ObjSen1의 위치에 있는 장치의 종류
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
Type2	string	ObjSen2의 위치에 있는 장치의 종류
...	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
...	string	ObjSen i의 위치에 있는 장치의 종류
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시
TypeN	string	ObjSenN의 위치에 있는 장치의 종류
주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) BumperTouchSensor 장치 Type은 각각 “BUMPER”, “TOUCH"로 정의할 수 있는데, 사용하는 형태에 따라 각각 알맞게 정의하지만, 동작은 같음.		

6.3.3 인터페이스

6.3.3.1 GetSensorValue

범퍼터치센서의 모든 센서들의 측정된 값들을 읽는다. 범퍼(혹은 터치)센서가 다른 물체에 닿은 상태면 TRUE고, 닿지 않으면 FALSE이다. OPRoS::Bool의 변수 중 status 변수는 OPROS_VALIDATA(성공적으로 값들을 읽어 온 경우), OPROS_ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 OPROS_INVALIDATA(데이터가 검출되지 않는 경우)로 표시된다.

Declaration	int32_t GetSensorValue(vector<OPRoS::Bool> &sensorValue)
Arguments	sensorValue : 각 범퍼(ObjSen1, ObjSen2,... ObjSenN)의 상태이며, 상태 값은 SUCCESS(성공적으로 적외선센서의 값들을 읽어 온 경우), ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 NOT_DETECTED(데이터가 검출되지 않는 경우)이다.
Return Value	API_SUCCESS : 성공적으로 범퍼의 상태 데이터를 읽어 온 경우 API_ERROR : 데이터를 읽는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

6.3.4 사용방법

범퍼 터치 센서 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

6.3.4.1 범퍼센서의 데이터 읽기

범퍼 터치 센서 API가 활성화된 상태에서 GetSensorValue 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

6.4 자이로센서 API

6.4.1 개요

로봇 혹은 자이로 센서가 부착된 부분의 현재 자세를 오일러 각으로 roll, pitch, yaw의 순서로 값이 출력됩니다.

6.4.2 속성

자이로센서 API는 센서 기저 API의 속성을 상속받아 사용하며, 속성 인덱스는 아래의 표와 같이 정의된다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 2년 12월 26일의 버전은 1.1로 함
count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type1	string	ObjSen1의 위치에 있는 장치의 종류
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type2	string	ObjSen2의 위치에 있는 장치의 종류
...	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
...	string	ObjSen i의 위치에 있는 장치의 종류
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시

typeN	string	ObjSenN의 위치에 있는 장치의 종류
주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) 자이로센서 장치 Type은 “GYRO”로 정의한다.		

6.4.3 인터페이스

6.4.3.1 GetSensorValue

자이로 센서의 측정 정보를 읽어오는 함수이다. 또한 OPRoS::GyroSensorData의 변수 중 status 변수는 OPROS_VALIDATA(성공적으로 값들을 읽어 온 경우), OPROS_ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 OPROS_INVALIDATA (데이터가 검출되지 않는 경우)로 표시된다.

Declaration	int32_t GetSensorValue(vector<OPRoS::GyroSensorData> &sensorValue)
Arguments	sensorValue : 자이로 장치의 x, y, z 센서 정보가 저장됨
Return Value	API_SUCCESS : 성공적으로 API를 초기화한 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우

6.4.4 사용방법

자이로 센서 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

6.4.4.1 자이로 값 읽기

자이로 센서 API가 활성화된 상태에서 GetSensorValue 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

6.5 관성측정 장치 API

6.5.1 개요

관성 측정 장치는 가속도 센서와 각속도 센서를 조합한 센서이다. 관성 측정 장치는 Roll, Pitch, Yaw 축의 각속도와 각도, X, Y, Z 축의 가속도가 출력되지만 종류에 따라 9개의 데이터를 전부 제공하는 것이 아니라 6개만을 제공하는 경우도 있다. 이런 경우 지원되지 않는 값은 0.0으로 고정시켜야 한다

6.5.2 속성

관성 측정 장치 API는 센서 기저 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type1	string	ObjSen1의 위치에 있는 장치의 종류
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type2	string	ObjSen2의 위치에 있는 장치의 종류
...	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
...	string	ObjSen i의 위치에 있는 장치의 종류
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시
typeN	string	ObjSenN의 위치에 있는 장치의 종류
주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) 관성측정장치 Type은 “IMU-9”(9개 값이 지원되는 경우), “IMU-6”(9개 값이 지원되는 경우)으로 정의한다.		

6.5.3 인터페이스

6.5.3.1 GetSensorValue

관성 측정 장치의 측정정보를 읽어오는 함수이다. 또한 OPRoS::ImuData의 변수 중 status 변수는 OPROS_VALIDATA(성공적으로 값들을 읽어 온 경우), OPROS_ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 OPROS_INVALIDATA(데이터가 검출되지 않는 경우)로 표시된다.

Declaration	int32_t GetSensorValue(vector<OPRoS::ImuData> &sensorValue)
Arguments	sensorValue : 관성측정 센서에서 측정된 값을 담을 변수
Return Value	API_SUCCESS : 성공적으로 API를 초기화한 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우

6.5.4 사용방법

관성 측정 장치 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

6.5.4.1 관성 측정 장치 값 읽기

관성 측정 장치 API가 활성화된 상태에서 getValue 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

6.6 근접 센서 API

6.6.1 개요

근접 센서의 예에는 적외선 센서 및 초음파 센서가 있다. 적외선 센서는 적외선을 이용하여 거리를 측정하는 센서이다. 초음파 센서는 초음파를 이용하여 거리를 측정하는 센서이다.

6.6.2 속성

근접 센서 API는 센서 기저 API의 속성을 상속받아 사용하며, 속성의 인덱스는 센서 기저 API 절에서 정의한 표와 같이 다음 표와 같이 정의된다. 즉 센서의 위치 정보 데이터를 활용한다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를

		표시
Type1	string	ObjSen1의 위치에 있는 장치의 종류
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
Type2	string	ObjSen2의 위치에 있는 장치의 종류
…	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
…	string	ObjSen i의 위치에 있는 장치의 종류
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시
TypeN	string	ObjSenN의 위치에 있는 장치의 종류
주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) 적외선 및 초음파 장치 Type은 각각 “IR”, “SONIC”으로 정의함		

6.6.3 인터페이스

6.6.3.1 GetSensorValue

근접 센서로부터 모든 센서들의 측정된 값을 읽는다. 센서 값들은 실제 측정한 데이터와 측정 데이터의 오류 여부를 나타내는 상태 데이터의 쌍으로 표시된다. 또한 OPRoS::Float64Array의 변수 중 status 변수는 OPROS_VALIDATA(성공적으로 값들을 읽어 온 경우), OPROS_ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 OPROS_INVALIDATA(데이터가 검출되지 않는 경우)로 표시된다. Float64Array를 Float64로 사용하지 않은 것은 근접 센서의 경우는 여러개의 센서들이 하나의 모듈로써 존재할 수 있기 때문이다. 또한 하나의 모듈에 하나의 근접 센서가 있는 경우에는 한 개의 값만 사용하면 된다. 모듈에 있는 센서의 수는 vector의 내장 함수로 확인할 수 있다.

Declaration	int32_t GetSensorValue(vector<OPRoS::Float64Array> &sensorValue)
Arguments	sensorValue : 측정된 데이터를 저장할 버퍼, 데이터 순서는 속성에서 정의된 ObjSen1, ObjSen2,... ObjSenN 순이다. 측정된 데이터 단위는 m이다.
Return Value	API_SUCCESS : 성공적으로 적외선센서의 값들을 읽어 온 경우 API_ERROR : 현재 데이터를 읽는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

6.6.4 사용방법

근접 센서 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

6.6.4.1 모든 근접 센서 값 읽기

근접 센서 API가 활성화된 상태에서 GetSensorValue 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

6.7 레이저 거리 측정기 API

6.7.1 개요

레이저 거리 측정기는 다수의 포인트로 일정한 범위의 거리를 계측하여 장애물의 유무를 판단할 수 있는 장치이다.

6.7.2 속성

레이저 거리 측정기 API는 센서 기저 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를 표시
LaserScanAng1	LaserScanAngle	ObjSen1이라는 Laser Scanner가 가지는 3종의 속성 정보
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
LaserScanAng2	LaserScanAngle	ObjSen2이라는 Laser Scanner가 가지는 3종의 속성 정보
...	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
..	LaserScanAngle	ObjSen i이라는 Laser Scanner가 가지는 3종의 속성 정보
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시
LaserScanAngN	LaserScanAngle	ObjSenN이라는 Laser Scanner가 가지는 3종의 속성 정보

주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) LaserScanner 장치 Type은 각각 “LASERSCAN”로 정의함

```
typedef struct {
        float64_t scanAngle ;  //  HW 장치가 가질수 있는 스캔 범위(각도)
        float64_t startAngle ;  // 알고리즘이 사용하는 스캔 시작 지점(각도)
        float64_t endAngle ;  // 알고리즘이 사용하는 스캔 종료 지점(각도)
        int32_t   noRay ;      // 알고리즘이 startAngle과 endAngle 사이에서
                                //   사용하는 Ray의 수
    }  LaserScanAngle ;
```

6.7.3 인터페이스

6.7.3.1 GetSensorValue

레이저 거리 측정기에서 모든 센서들의 측정된 값을 얻는다. index에서부터 size 만큼 측정된 상태를 읽는다. OPRoS::Float64Array의 변수 중 status 변수는 OPROS_VALIDATA(성공적으로 값들을 읽어 온 경우), OPROS_ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 OPROS_INVALIDATA(데이터가 검출되지 않는 경우)로 표시된다.

Declaration	int32_t GetSensorValue(vector<OPRoS::Float64Array> &sensorValue)
Arguments	sensorValue : 측정된 데이터를 저장할 버퍼, 데이터 순서는 속성에서 정의된 ObjSen1, ObjSen2,... ObjSenN 순이다. 측정된 데이터 단위는 m이다.
Return Value	API_SUCCESS : 성공적으로 레이저 스캐너의 값들을 읽어 온 경우 API_ERROR : 데이터를 읽는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

6.7.4 사용방법

레이저 거리 측정기 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

6.7.4.1 모든 레이저 스캐너 값 읽기

레이저 거리 측정기 API가 활성화된 상태에서 GetSensorValue 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

6.8 위치 센서 API

6.8.1 개요

현재 위치 값을 읽어오는 장치 API 이다. 값은 월드좌표 기준에서 장치 프레임의 x, y, z 입니다. 이 API를 사용하는 장치의 예는 초음파 센서, IR 센서, starGaze 등으로, 이러한 센서들을 활용하여 로봇의 위치를 파악할 때 사용한다.

6.8.2 속성

위치 센서 API는 센서 기저 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type1	string	ObjSen1의 위치에 있는 장치의 종류
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type2	string	ObjSen2의 위치에 있는 장치의 종류
...	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
...	string	ObjSen i의 위치에 있는 장치의 종류
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시
typeN	string	ObjSenN의 위치에 있는 장치의 종류
주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) 위치 센서 장치 Type은 “POS”로 정의한다.		

6.8.3 인터페이스

6.8.3.1 GetSensorValue

위치 센서 장치의 측정 값들을 읽어오는 함수이다. OPRoS::ObjectPositionData의 변수 중 status 변수는 OPROS_VALIDATA(성공적으로 값들을 읽어 온 경우), OPROS_ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 OPROS_INVALIDATA (데이터가 검출되지 않는 경우)로 표시된다.

Declaration	int32_t GetSensorValue(vector<OPRoS::ObjectPositionData> &sensorValue)
Arguments	sensorValue : Position Data 정보가 저장됨
Return Value	API_SUCCESS : 성공적으로 API를 초기화한 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우

6.8.4 사용방법

위치 센서 API 중 장치 API에서 상속받은 인터페이스는 장치 API와 동일한 방법으로 사용된다.

6.8.4.1 위치센서 값 읽기

위치 센서 API가 활성화된 상태에서 GetPosData 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

6.9 지자기센서 API

6.9.1 개요

지구에서 발생하는 자기장의 흐름을 파악해 나침반처럼 방위를 탐지할 수 있는 센서이다. 지자기센서는 자기장의 세기를 측정할 수 있는 센서가 X, Y, Z 축 방향으로 3개가 부착되어 있다.

6.9.2 속성

지자기 센서 API는 센서 기저 API의 속성을 상속받아 사용하며, 속성 인덱스는 아래의 표와 같이 정의된다. count를 제외한 항목은 count의 값에 따라 개수가 변화되는 항목이다. 그래서 이 항목들을 설정할 때 ObjectPostion1, ObjectPostion2, ..ObjectPostionN과 표기될 수 있다. 여기서 N은 count, 즉 센서의 개수를 의미한다. ObjectPostion은 실제 사용할 때는 x, y, z, roll, pitch, yaw 값을 나열하여 사용한다. 예를 들면, ObjSen0의 경우 x0, y0, z0, roll0, pitch0, yaw0에 할당시키고, ObjSenN의 경우 xN, yN, zN1, rollN, pitchN, yawN에 값을 할당시킨다.

Index	Type of Value	Description
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
count	int32_t	센서의 개수
ObjSen1	ObjectPosition	3차원 공간상 센서 1의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type1	string	ObjSen1의 위치에 있는 장치의 종류
ObjSen2	ObjectPosition	3차원 공간상 센서 2의 위치 정보 x, y, z, roll, pitch, yaw를 표시
type2	string	ObjSen2의 위치에 있는 장치의 종류
...	ObjectPosition	3차원 공간상 센서 들의 위치 정보 x, y, z, roll, pitch, yaw를 표시
...	string	ObjSen i의 위치에 있는 장치의 종류
ObjSenN	ObjectPosition	3차원 공간상 센서 N의 위치 정보 x, y, z, roll, pitch, yaw를 표시
typeN	string	ObjSenN의 위치에 있는 장치의 종류
주) 로봇의 기준점과 홈잉(Homing) 위치를 기준으로 각 센서들의 ObjectPosition로써 위치 정보를 정한다. 주) 지자기 센서 장치 Type은 “GeoMag”로 정의한다.		

6.9.3 인터페이스

6.9.3.1 GetSensorValue

지자기 센서의 측정 정보를 읽어오는 함수이다. OPRoS::GeoMagneticData의 변수 중 status 변수는 OPROS_VALIDATA(성공적으로 값들을 읽어 온 경우), OPROS_ERROR(현재 데이터를 읽는 도중 에러가 발생한 경우)와 OPROS_INVALIDATA(데이터가 검출되지 않는 경우)로 표시된다.

Declaration	int32_t GetSensorValue(vector<OPRoS::GeoMagneticData> &sensorValue)
Arguments	GeoMagneticData : 지자기 센서의 x, y, z축에 대한 자기장의 크기가 저장됨
Return Value	API_SUCCESS : 성공적으로 API를 초기화한 경우 API_ERROR : API를 초기화하는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 이 API를 지원하지 않은 경우

6.9.4 사용방법

지자기 센서는 장치 API에서 상속받은 인터페이스로 장치 API와 동일한 방법으로 사용된다.

6.9.4.1 지자기 센서 값 읽기

지자기 센서 API가 활성화된 상태에서 GetSensorValue 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

7 카메라 API

7.1 단순 카메라 API

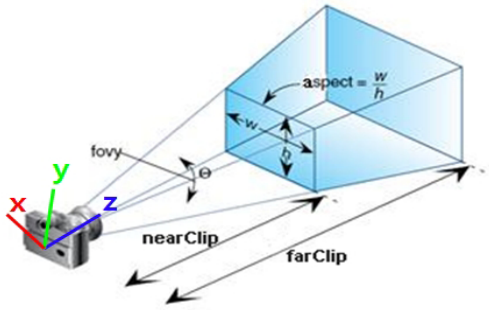
7.1.1 개요

카메라는 외부로부터 얻는 사진 및 영상을 데이터로 기록하는 장치이다. 본 문서에는 카메라로 이미지 데이터를 읽어오는 인터페이스를 다룬다.

7.1.2 속성

카메라 API는 장치 API의 속성을 상속받아 사용하며, 속성의 인덱스는 아래의 표와 같이 정의된다. 카메라가 2개 이상인 경우 하나씩 표현한다. 즉 컴포넌트 한 개당 하나의 카메라 API를 사용하며 스테레오 카메라는 스테레오 카메라 API를 사용하여야 한다. 이러한 속성은 카메라가 제공하는 속성 중에서 설정해야 한다.

Index	Type of Value	Description
ApiName	string	API에서 사용할 dll 파일의 이름이다
Version	string	속성 파일의 버전. 12년 12월 26일의 버전은 1.1로 함
CameraID	string	카메라 ID (사용자가 입력할 수 있음)
ObjSen	ObjectPosition	3차원 공간상 카메라의 위치 정보 x, y, z, roll, pitch, yaw를 표시
Width	int32_t	이미지의 가로 크기
Height	int32_t	이미지의 세로 크기
ColorType	int32_t	Color : 1 BW : 2
CompressType	int32_t	압축파일 포맷. 0: Gray, 1: YCbCr, 2: RGB, 3: JPEG 4 : PNG, 5: BMP, 6: GIF



7.1.3 인터페이스

7.1.3.1 GetImage

버퍼에 저장된 이미지를 가져온다

Declaration	int32_t GetImage(OPRoS::CameraData *Image)
Arguments	image : 이미지 버퍼의 시작 주소 (OPRoS::CameraData)
Return Value	API_SUCCESS : 성공적으로 이미지의 시작 주소를 얻은 경우 API_ERROR : 이미지 시작 주소를 얻는 도중 에러가 발생한 경우 API_NOT_SUPPORTED : 지원하지 않는 경우

7.1.3.2 StartPushingImage

이미지가 처리가 완료되면 해당 함수를 호출함

Declaration	int32_t StartPushImage(int32_t period, int32_t format, int32_t width, int32_t height, void (*callback), void *arg)
Arguments	period: 이미지 전송 주기, 이미지 전송을 위한 전송 단위를 밀리세컨드 (mili-second) 단위로 설정한다. width : 이미지 해상도 height : 이미지 해상도 format: 이미지 포맷 arg : callback 함수를 등록한 인스탄스의 pointer void (*call_back) (int32_t bufsize, uint*_t *imgbuf, void *arg); bufsize: 이미지 버퍼 크기 (byte) imgbuf: 이미지 버퍼 포인터 arg : callback 함수를 등록한 인스탄스의 pointer
Return Value	API_SUCCESS : 성공적으로 이미지의 시작 주소를 얻은 경우 API_ERROR : 이미지 시작 주소를 얻는 도중 에러가 발생한 경우

7.1.3.3 EndPushImage

StartPushImage 함수의 호출을 정지시킴

Declaration	int32_t StopPushImage(void)
Arguments	없음
Return Value	API_SUCCESS : 성공적으로 이미지의 시작 주소를 얻은 경우 API_ERROR : 이미지 시작 주소를 얻는 도중 에러가 발생한 경우

7.1.3.4 getStatus(int32_t &error)

device : Error의 종류를 정의한다. Error의 종류는 사용된 카메라의 에러 종류를 그대로 사용한다. 선택적으로 구현가능함.

7.1.4 사용방법

카메라 API 중 센서 API에서 상속받은 인터페이스는 센서 기저 API와 동일한 방법으로 사용된다.

7.1.4.1.1 이미지 데이터 획득

카메라 API가 활성화된 상태에서 GetImage 인터페이스를 호출한다. 이 API가 활성화되지 않았다면 에러를 반환한다.

7.1.4.2 StartPushImage

이미지를 가져오라는 callback 함수를 등록할 때 사용.

7.1.4.3 EndPushImage

더 이상 StartPushImage에서 등록된 callback 함수를 사용하지 않을 경우에 사용

7.1.4.4 getStatus(int32_t &error)

Error의 종류를 알고 싶을 때 사용하며, 선택적으로 구현 가능함.

부 록 Ⅰ-1

지식재산권 확약서 정보

Ⅰ-1.1 지식재산권 확약서

해당 사항 없음

부 록 Ⅰ-2

시험인증 관련 사항

Ⅰ-2.1 시험인증 대상 여부

해당 사항 없음

Ⅰ-2.2 시험표준 제정 현황

해당 사항 없음

부 록 Ⅰ-3

본 표준의 연계(family) 표준

Ⅰ-3.1 개방형 로봇 소프트웨어 플랫폼 시리즈

개방형 로봇 소프트웨어 플랫폼의 표준은 개방형 로봇 소프트웨어의 컨테이너인 프레임워크는 물론 컴포넌트 및 개발도구들도 다양한 사업 현장에 맞도록 수정 변경이 요구될 수 있다. 수정된 사항이 기 개발된 OPRoS와 연동을 보장하기 위한 사항들을 표준으로 하고자 한다.
개방형 로봇 소프트웨어 플랫폼의 주요한 요소인 컴포넌트, 로봇 및 서버 간 통신 프로토콜, 원격 서비스를 위한 인터페이스들에 적용되는 내용들을 각 표준에서 기술하고 있다.

Ⅰ-3.2 개방형 로봇 소프트웨어 플랫폼 평가 시리즈

소프트웨어의 시험(절차 및 코드의 실행 등)과 관련된 일련의 기본 시험 문서들에 대해 설명하며, 각 기본 문서의 목적, 개요, 그리고 목차를 정의한다. 본 표준에서 기술된 문서들은 동적 시험에 초점을 두고 있지만, 몇몇 문서들은 다른 시험 활동에도 적용 가능하다.
개방형 로봇 소프트웨어 플랫폼 평가 표준은 로봇 소프트웨어를 검증하기 위한 시험 문서 작성 방법, 시험 사례 생성 방법, 개방형 로봇 소프트웨어 컴포넌트 시험 방법, 그리고 로봇 장치 추상화를 위한 공통 로봇 인터페이스 시험 방법을 포함한다. 로봇에 사용되는 S/W (컴포넌트, 프레임워크, 통합개발환경 등에 포함되어 있는 S/W이며 H/W와 연동되는 S/W도 포함)들의 시험에 대한 절차에 관련되어 기술되어 있으며, 시험을 하기 위해서는 시험계획서, 시험설계명세서, 시험케이스명세서, 시험절차 명세서를 순서대로 만들고 시험을 수행할 때에는 시험상황기록, 시험사고(incident), 시험요약보고서를 작성하여 제출하여야 한다. 특히 시험수행 결과로써 최소한 시험요약보고서는 반드시 제출하여야 함을 각 표준에서 기술하고 있다.

부 록 Ⅰ-4

참고 문헌

해당 사항 없음

부 록 Ⅰ-5

영문표준 해설서

해당 사항 없음

부 록 Ⅰ-6

표준의 이력

판수	채택일	표준번호	내용	담당 위원회
제1판	2016.12.xx	제정 TTAx.xx-xx.xxxx	-	지능형로봇 프로젝트그룹 (PG413)