TTA Standard

정보통신단체표준(국문표준)

TTAx.xx-xx.xxxx/R1 제정일: 20xx년 xx월 xx일

스마트디바이스와 로봇 간의 인터페이스 및 서비스 - 제2부 : 로봇 인터페이스

Interface between Smart-device and Robot - Part 2 : Robot Interface

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	김동한	경희대학교	교수	-	
표준 초안 작성자	김동한	경희대학교	교수	-	
	홍성표	피플앤드 테크놀러지	부대표	-	
	전상원	파인로보틱스	이사	-	
	조영조	ETRI	책임	-	
	서준호	KAR	팀장	-	
	성기엽	KAR	대리	-	
	홍승택	KAR	전임	-	
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 스마트디바이스와 로봇 간의 표준 인터페이스 규격을 정의하기 위하여 작성하였다.

2 주요 내용 요약

시장에서는 점차 가전제품과 로봇기술을 융합한 모델, 전화기에 홈모니터링 등의 서비스를 지원하는 모델 등 다양한 아이디어의 제품과 서비스를 결합하여 새로운 시스템이 만들어 지고 있다. 따라서 이 표준은 점차 커지고 있는 스마트디바이스와 로봇기술을 융합한 새로운 시장 활성화를 위해서 스마트디바이스와 로봇 간의 표준 인터페이스 규격을 정의하고자 한다.

3 인용 표준과의 비교

3.1 인용 표준과의 관련성

해당사항 없음

Preface

1 Purpose

This Standard has been enacted for specifying the interface between smart devices, such as smartphone, and robots.

2 Summary

This Standard describes the specification of the interface between smart devices, such as smartphone, and robots. The expected effect of this standard is to vitalize the smart robot industry and also expected to encourage interactions among smart devices and robots.

3 Relationship to Reference Standards

None

목 차

스마트디바이스와 로봇 간의 인터페이스 및 서비스 - 제2부 : 로봇 인터페이스 (Interface between Smart-device and Robot - Part 2 : Robot Interface)

1 적용 범위

시장에서는 점차 가전제품과 로봇기술을 융합한 모델, 전화기에 홈모니터링 등의 서비스를 지원하는 모델 등 다양한 아이디어의 제품과 서비스를 결합하여 새로운 시스템이 만들어 지고 있다. 따라서 본 표준에서는 점차 커지고 있는 스마트디바이스와 로봇기술을 융합한 새로운 시장 활성화를 위해서 스마트디바이스와 로봇 간의 표준 인터페이스 규격을 정의하고자 한다.

2 인용 표준

KOROS 1063:2008, 서비스로봇의 콘텐츠 용어 및 정의
KOROS 1030:2007, 지능형로봇 가상 모델링 기본요소 - 정의 및 용어
KOROS 1031:2007, 지능형로봇 가상평가를 위한 모델링 및 후처리요소 정의 및 용어
KOROS 1110-1:2016, 스마트디바이스와 로봇 간의 인터페이스 및 서비스 - 제1부 : 용어 및 정의
ISO/IEC 19501, Unified Modeling Language Specification

3 용어 정의

3.1 로봇 인터페이스 (Robot Interface)

스마트 디바이스와 로봇간의 연결과 명령어 송신 등의 로봇 제어와, 로봇에서 발생하는 각종 상태 값 수신 등을 연계하는 역할을 수행하며, 스마트 디바이스 내부에서 작동하는 로봇 제어 인터페이스를 의미함

3.2 스마트 디바이스 로봇 (Smart Device Robot)

스마트 폰과 태블릿(패드)의 CPU와 각종 센서 등의 스마트 디바이스의 리소스를 활용하여 로봇의 각종 여러 동작 및 사용 시나리오를 구현하는 로봇. 본 표준에서 정의하는 로봇은 스마트 디바이스 로봇에 한정됨

3.3 로봇 ID (Robot ID)

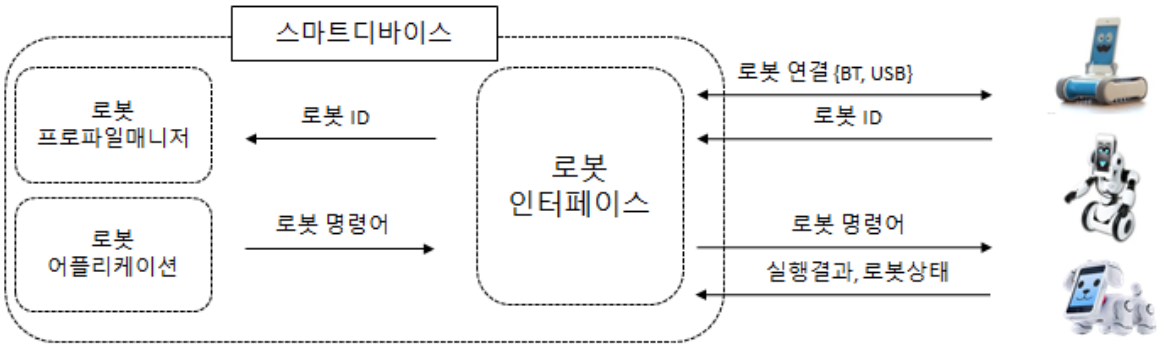
다수 존재하는 스마트 디바이스 로봇의 유일 구분자(Unique Identification)으로서, 로봇 인터페이스에서 어떤 로봇이 연결되었고, 제어 대상인지를 구분하는 ID로 사용됨

4 로봇 인터페이스의 역할

4.1 로봇 인터페이스의 구조

로봇 어플리케이션을 이용하여 로봇을 조정하기 위해서는, 먼저 스마트 디바이스와 로봇의 연결이 이루어져야한다. 로봇 연결이 이루어진 이후에는 로봇 서비스 플랫폼의 다른 컴포넌트들의 동작이 가능하다. 예를 들면, 로봇 프로파일 매니저는 연결된 로봇 리스트들에 대한 정보를 로봇 인터페이스로부터 제공받게 된다. 또한, 로봇 어플리케이션 앱도 사용자에게 보여줄 로봇들에 대한 리스트를 로봇 인터페이스로부터 제공받게 된다. 이러한 특징으로 보면, 로봇 어플리케이션이 로봇을 조정하기 위해서 첫 번째로 필요한 컴포넌트가 로봇 인터페이스라고 할 수 있다.

따라서 로봇인터페이스의 역할은 로봇 어플리케이션 및 로봇 서비스 플랫폼에 포함된 다른 컴포넌트로부터 수신된 로봇 명령어를 실제 로봇으로 전달하고, 명령어 실행의 결과 및 로봇의 상태를 다시 전달하는 것이라고 정의 할 수 있다. 아래의 (그림 4-1)은 이러한 로봇 인터페이스의 역할에 대해 설명하고 있다. 본 표준에서의 로봇 인터페이스는 다수의 로봇과 서로 N:N 관계로 연결 및 제어를 할 수 있다.



(그림 4-1) 스마트 디바이스 로봇의 구조

로봇 인터페이스는 다른 컴포넌트들에게 연결된 로봇의 ID를 제공한다. 로봇 ID를 제공받은 컴포넌트들이 해당 로봇 ID와 함께 로봇에서 실행되어질 명령어를 로봇 인터페이스에게 전송하게 되면, 로봇 인터페이스는 해당 명령어를 실제 로봇에게 전송하게 된다.

4.2 로봇 인터페이스의 기능

로봇 인터페이스는 최초 실행 시에, 기등록되어져 있는 로봇들에 대한 정보들을 메모리에 유지하

게 된다. 로봇 인터페이스가 실행 중일 때, 새롭게 등록되어지는 로봇이 있다면 연결 기능을 먼저 실행하고 이후 등록절차를 진행하게 된다. 등록절차가 완료되면, 로봇 명령어를 연결된 로봇에 전달하게 된다.

4.3 기등록된 로봇 정보 로딩

로봇 인터페이스는 등록된 로봇에 대한 정보를 유지하고 있다. 이러한 정보는 로봇 서비스 플랫폼에 포함된 어떤 컴포넌트의 실행 여부와는 상관없이 유지되어지고, 스마트 디바이스가 재부팅되더라도 로봇 인터페이스에 의해서 유지되어야 한다. 로봇 인터페이스가 실행되어질 때 등록된 로봇에 대한 정보를 메모리로 로딩 한다.

4.3.1 신규 등록(연결) 로봇 정보 추가 관리

별도의 응용 로봇 어플리케이션에서 제공하는 로봇 셋팅 기능에 의해 새로운 로봇의 연결 및 등록 기능을 수행한다. 블루투스의 경우 연결 기능은 스마트 디바이스와 특정 로봇의 블루투스 페어링(paring) 및 본딩(bonding) 기능을 의미한다. 연결 기능이 완료되게 되면 해당 로봇을 등록하는 것이 가능하게 된다. 로봇을 등록한다는 것은 로봇 인터페이스에서 등록된 로봇으로 관리하겠다는 것을 의미하며, 등록된 로봇들의 정보는 로봇 서비스 플랫폼의 다른 컴포넌트에 제공될 수 있다.

4.3.2 로봇 명령어 전달

로봇 어플리케이션에서 전달되어지는 로봇 명령어는 로봇 명령어 중계기에 의해서 특정 로봇이 이해할 수 있는 로봇 원시 명령어로 변환되어지거나 로봇 어플리케이션에서 바로 원시명령어로 변환되어 로봇 인터페이스에 전달된다. 로봇 인터페이스는 이러한 로봇 원시 명령어를 해당 로봇에게 전달하게 된다.

4.3.3 로봇 명령어 실행결과와 상태값 수신

로봇 명령어의 종류에 따라 실행 결과를 리턴 받을 수 있다. 또한 로봇 명령어와는 상관없이, 로봇에서 특정 조건에 의해서 로봇의 상태를 로봇 인터페이스에 전달할 수 있다. 이렇게 수신된 로봇 명령어 실행결과나 로봇 상태 값은 로봇 어플리케이션에 전달된다.

5 로봇 인터페이스의 구성

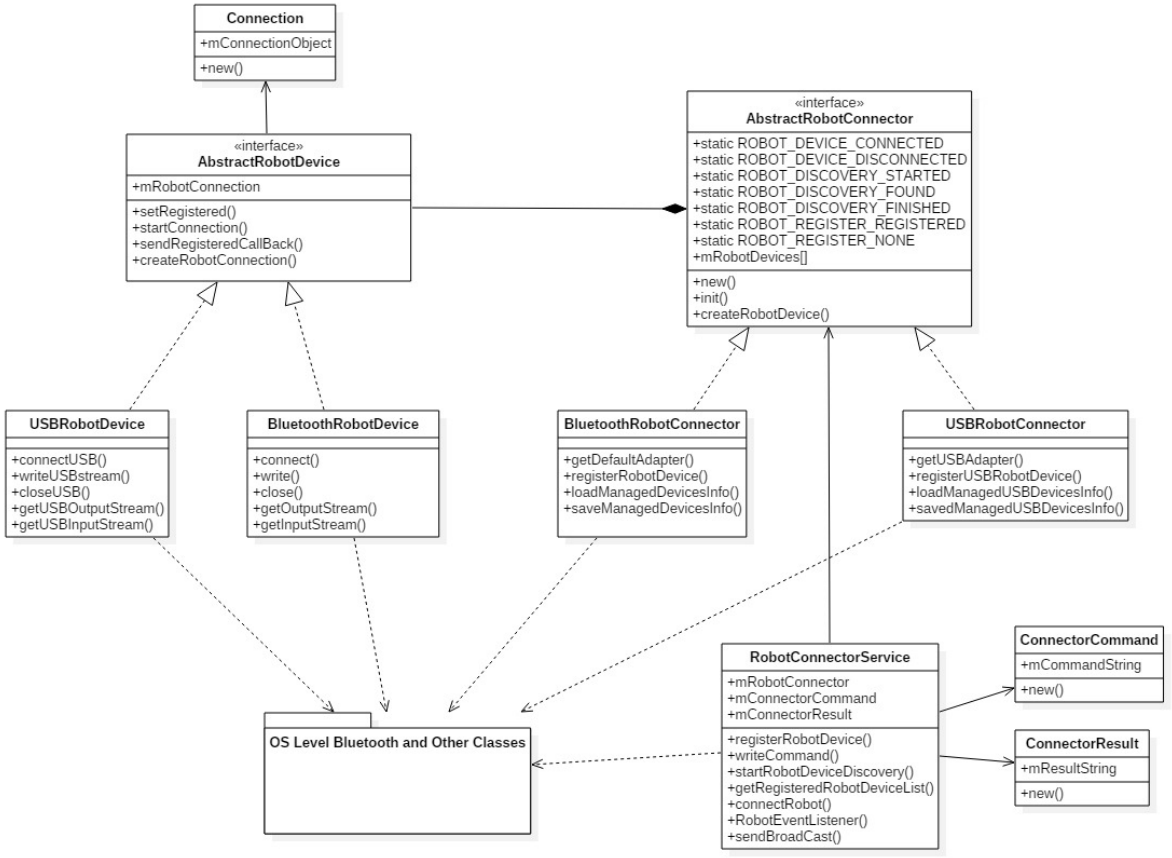
5.1 개요

로봇 인터페이스를 구성하고 있는 세부 모듈에 대해 클래스 다이어그램, 프로그램 흐름을 도식화하는 시퀀스 다이어그램, 외부 인터페이스 함수 등을 통하여 로봇 인터페이스

의 구조 및 동작흐름에 대해 설명한다.

5.2 클래스 다이어그램

로봇 인터페이스를 구성하는 클래스 다이어그램은 아래의 (그림 5-1)의 구조를 하고 있으며, 클래스의 멤버 구성을 설명한다.



(그림 5-1) 로봇 인터페이스의 클래스 다이어그램

<표 5-1> 클래스의 주요 멤버 변수 및 함수구성

클래스 명	클래스 멤버	멤버 설명
AbstractRobotDevice	추상로봇디바이스	소프트웨어적으로 형상된 스마트 디바이스 로봇 상위 추상 클래스
	주요멤버변수	mRobotConenction : 로봇 연결 객체 유지
	주요멤버함수	createRobotConnection() :로봇과의 연결 실행

USBRobotDevice	USB로봇디바이스	로봇추상클래스를 상속받아 구현한 USB 연결 타입의 스마트디바이스 로봇
	주요멤버변수	AbstractRobotDevice 상속
	주요멤버함수	connectUSB() :USB연결을 통한 로봇 연결
BluetoothRobotDevice	블루투스로봇디바이스	로봇추상클래스를 상속받아 구현한 블루투스 연결 타입의 스마트디바이스 로봇
	주요멤버변수	AbstractRobotDevice 상속
	주요멤버함수	Connect() :블루투스 연결을 통한 로봇 연결
AbstractRobotConnector	추상로봇연결기	스마트디바이스와 로봇의 모든 연결을 관리하는 상위 추상 클래스
	주요멤버변수	mRobotDevices[] :로봇추상클래스들의 Array
	주요멤버함수	createRobotDevice() :로봇디바이스 생성
BluetoothRobotConnector	블루투스로봇연결기	블루투스 타입의 로봇 연결기
	주요멤버변수	AbstractRobotConnector 상속
	주요멤버함수	loadManagedDevicesInfo() :기 연결된 로봇 디바이스 정보 로드
USBRobotConnector	USB로봇연결기	USB 타입의 로봇 연결기
	주요멤버변수	AbstractRobotConnector 상속
	주요멤버함수	loadManagedUSBDevicesInfo() :기 연결된 USB연결 타입의 로봇 디바이스 정보 로드
RobotConnectorService	로봇연결서비스	로봇연결기를 항상 구동하는 로봇 인터페이스의 메인 연결 프로세스
	주요멤버변수	mRobotConnector :로봇연결기 객체 유지
	주요멤버함수	writeCommand() :로봇명령어전달
		RobotEventListener() :로봇으로부터 전달받은 이벤트 콜백함수
ConnectorCommand	로봇명령어	로봇명령어 클래스
	주요멤버변수	mCommandString :로봇명령어 오브젝트
	주요멤버함수	new() :로봇명령어 클래스 생성
ConenctorResult	로봇명령수행결과	명령결과 리턴 받는 클래스
	주요멤버변수	mResultString :로봇명령어 결과 오브젝트
	주요멤버함수	new() :로봇명령어 수행결과 클래스 생성

Connection	로봇연결객체	스마트디바이스와 로봇간의 연결 객체 클래스
	주요멤버변수	mConnectionObject :로봇연결 Session 오브젝트
	주요멤버함수	new() :로봇연결 객체 클래스 생성

로봇 인터페이스의 주요한 기능인 로봇 연결 기능 및 로봇 명령어 전달과 관련한 주요한 클래스에 대한 설명은 다음과 같다.

5.2.1 RobotConnectorService

로봇 인터페이스의 메인 프로세스이다. 로봇 인터페이스가 시작되어지는 시점에 기등록되어진 모든 로봇들에 대하여 RobotDevice 인스턴스를 생성하여 메모리에 유지한다. 그 이후 로봇 인터페이스가 실행 중일 때는, 새로운 로봇에 대한 연결 을 실행한다. 만약 새로운 로봇의 등록 요청이 있을 때는 새로운 RobotDevice 인스턴스를 생성하고, 기존 로봇 연결과 동일한 동작 프로세스를 실행한다.

5.2.2 AbstractRobotConnector

로봇 인터페이스의 추상화 상위 클래스 이다. 모든 연결 프로토콜에 대한 상위 클래스로서, 물리적인 연결 방식(블루투스, USB, Wi-FI, TCP/IP Socket등)에 따라서, 해당 연결 프로토콜에 대한 상속 클래스(Extended Class)가 존재할 수 있다. 추상화된 기능 함수에 대한 구현부는 상속 클래스에서 구현한다. 예를 들어 BluetoothRobotConnector나 UsbRobotConnector와 같은 상속 클래스가 존재할 수 있다.

5.2.3 BluetoothRobotConnector

추상 클래스인 AbstractRobotConnector의 구체화된 클래스 중의 하나이다. 블루투스를 통한 로봇 인터페이스의 기능을 제공하기 위해, 블루투스와 관련된 추상 클래스 함수에 대한 실제 구현을 블루투스 프로토콜의 규격에 맞추어 제공해야 한다.

5.2.4 UsbRobotConnector

추상 클래스인 AbstractRobotConnector의 구체화된 클래스 중의 하나이다. USB를 통한 로봇 인터페이스의 기능을 제공하기 위해, USB와 관련된 추상 클래스 함수에 대한 실제 구현을 USB 프로토콜의 규격에 맞추어 제공해야 한다.

5.2.5 Abstract RobotDevice

물리적인 HW 로봇을 형상화한 추상화된 클래스이다. 로봇의 공통적인 속성에 대한 추상

화된 인스턴스 변수, 함수등에 대한 정의가 포함된다.

5.2.6 BluetoothRobotDevice

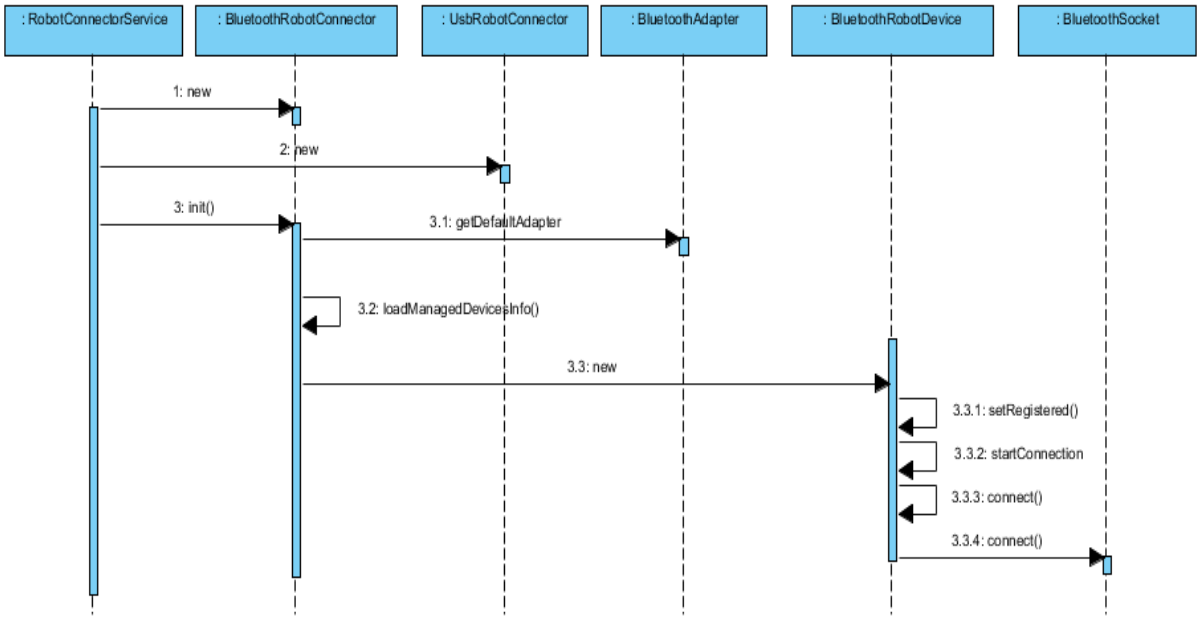
RobotDevice를 상속받는 구체화된 클래스로, 블루투스 연결 속성을 가진 로봇 디바이스를 형상화한다.

5.2.7 UsbRobotDevice

RobotDevice를 상속받는 구체화된 클래스로, USB 연결 속성을 가진 로봇 디바이스를 형상화 한다.

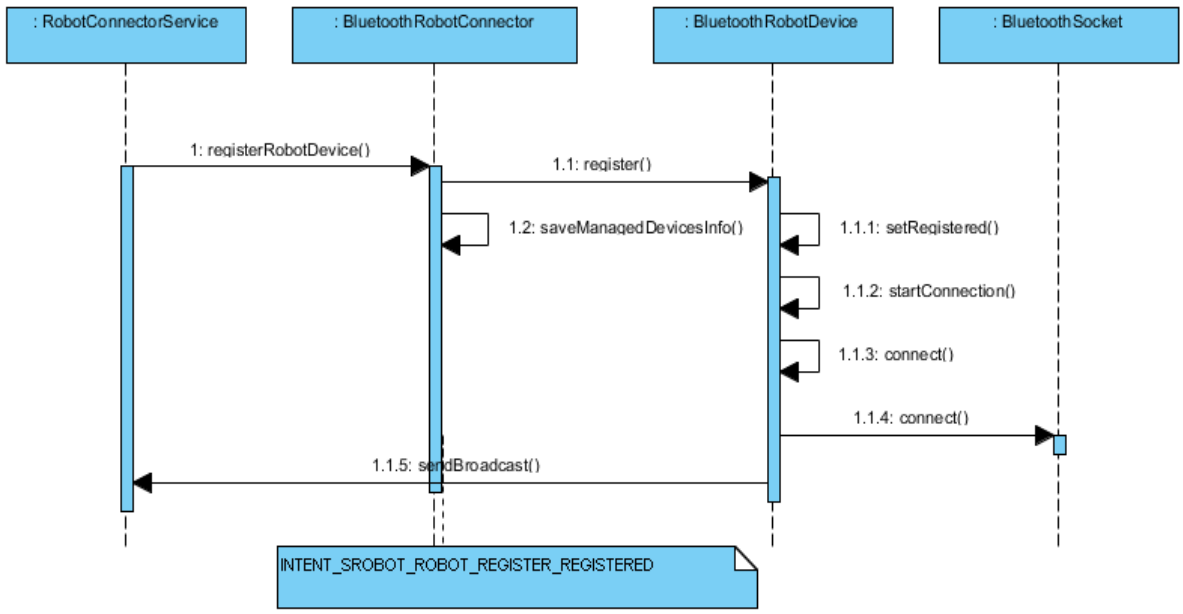
5.2.85 시퀀스 다이어그램

로봇 인터페이스가 구동되면, 등록된 로봇들에 대한 정보를 메모리에 유지하며, 로봇과 스마트 디바이스간의 연결을 실행한다. 아래의 (그림 5-2)에는 이러한 과정들을 설명한다.



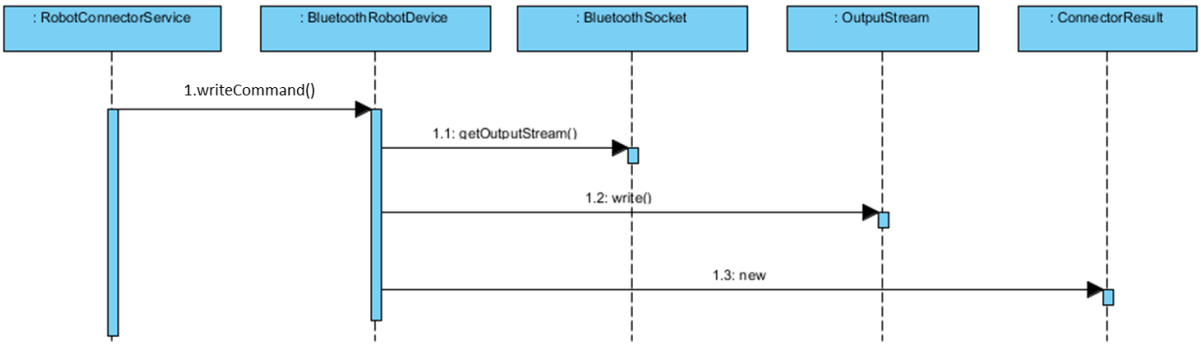
(그림 5-2) 로봇 인터페이스의 최초 구동 시, 프로그램 순서도

로봇 인터페이스가 실행 중에 새로운 로봇의 등록 요청이 들어오면, RobotDevice를 형상화하는 클래스를 인스턴스화하여 메모리에 유지한다. 아래의 (그림 5-3)에는 이러한 과정들을 설명한다.
신규 로봇이 등록될 때에는 등록되는 순간에 로봇과 스마트디바이스 간의 연결을 동시에 실행한다. 로봇의 연결까지 완료되면, 연결 완료되었다는 메시지를 로봇인터페이스에 전달해야 한다.



(그림 5-3) 특정 로봇 등록 시, 프로그램 순서도

로봇 연결 및 등록이 완료되면, 로봇 어플리케이션은 특정 로봇에게 로봇 명령어를 전달함으로써, 해당 로봇을 동작시킬 수 있다. 로봇 어플리케이션에서 로봇 서비스 프레임워크로 전달되어진 명령어는 해당 로봇이 이해할 수 있는 원시 명령어로 변환되어 전달되어야 한다. 이 원시 명령어가 로봇 인터페이스에 의해서 해당 로봇에 전달된다. 아래의 (그림 5-4)에는 이러한 과정들을 설명한다.



(그림 5-4) 로봇 명령어 전달 시, 프로그램 순서도

6 로봇 인터페이스의 주요 함수

로봇 인터페이스가 로봇 서비스 플랫폼을 구성하는 다른 컴포넌트들과 통신하기 위한 메세지 종류와 외부에 노출되어진 주요 함수들에 대해서 설명한다.

6.1 로봇 연결 결과 메시지

로봇 디바이스와의 연결 및 연결 해제가 일어날 때, 이를 로봇 어플리케이션에게 알려주어야 한다. 이를 위한 로봇 연결 결과 브로드캐스트 메시지를 다음과 같이 정의한다.

- srobot.intent.action.ROBOT_DEVICE_CONNECTED
  로봇 장치가 스마트 디바이스 기기에 연결될 때 발생하는 메시지로서, 연결된 로봇 디바이스핸들 값을 브로드캐스트 메시지의 인자 변수 Extra값으로 같이 전송한다.

- srobot.intent.action.ROBOT_DEVICE_DISCONNECTED
  로봇 장치가 스마트 디바이스 기기로부터 탈착될 때 발생하는 메시지로서, 탈착된 로봇 디바이스 핸들 값을 브로드캐스트 메시지의 인자 변수 Extra값으로 같이 전송한다.

새로운 로봇을 등록하기 위한 과정에서 각 과정의 진행 상태를 브로트 캐스트 메시지로 로봇 어플리케이션에게 알려줄 필요가 있다. 이를 위한 로봇 연결 과정 브로드캐스트 메시지를 다음과 같이 정의한다.

- srobot.intent.action.SROBOT_DISCOVERY_STARTED
  로봇 HW 디바이스 연결을 위한 특정 프로토콜(블루투스, USB등)의 연결 가능한 로봇 디바이스 검색이 시작되었음을 알린다.

- srobot.intent.action.SROBOT_DISCOVERY_FOUND
  특정 프로토콜에 의해 연결 가능한 로봇 디바이스가 하나씩 검색될 때 마다 알린다.

- srobot.intent.action.SROBOT_ROBOT_REGISTER_REGISTERED
  새로운 스마트디바이스 로봇이 등록되었음을 알린다.

- srobot.intent.action.SROBOT_ROBOT_REGISTER_NONE
  등록된 스마트 디바이스 로봇이 등록 해제되었음을 알린다. (예를 들어, 사용자가 명시적으로 안드로이드 설정에서 "unpair"하면 발생함)

6.2 인터페이스 함수

로봇 인터페이스는 다른 컴포넌트들이 사용할 수 있는 함수들을 외부 모듈에 제공해야 한다. 이를 위해 공통 인터페이스 함수를 설명한다.

- boolean startRobotDeviceDiscovery()
  지금 연결 가능한 로봇 디바이스들을 연결하기 위해 찾기 과정을 실시한다.

- List<RobotDeviceInfo> getRegisteredRobotDeviceList()
  기존에 이미 한번이라도 연결되어 연결정보가 등록되어 있는 로봇 디바이스의 인스턴스 리스트를 반환한다.

- boolean connectRobot(int handle)
  스마트 디바이스 로봇을 연결한다. 이때 파라미터 인자 값은 연결하고자 하는 로봇의 인스턴스 값이다. 연결이 성공하면 TRUE, 실패하면 FALSE를 반환한다.

- writeCommand(in ConnectorCommand cmd)
  로봇 디바이스에 로봇 명령어를 실행하기 위하여 명령어를 전달한다. 파라미터 인자 값은 로봇에 전달되는 원시 명령어의 Index값 등으로 로봇 디바이스마다 각각 고유의 값을 가진다.

- Listener Interface RobotEventListener(int handle, int evengroup, int status)
  로봇 디바이스의 상태 값은 주기적으로 스마트 디바이스에 전달된다. 전달된 로봇의 상태값은 로봇 인터페이스를 통해서 외부 모듈에 전달된다. 파라미터 인자 값은 로봇 인스턴스 핸들, 이벤트종류 구분, 이벤트 상태 값이다.

부 속 서 A

구현 Pseudo 코드 예시

로봇 인터페이스의 실제 구현 Pseudo 코드 중 블루투스로 연결되는 로봇 디바이스 부분에 대한 예시를 제시한다. 부속서에서는 스마트 디바이스를 Android 디바이스로 가정하고, Android에서 사용하는 프로그래밍 기법을 기준으로 설명하도록 한다. Pseudo 코드이므로 전체 구현부의 주요 부분만 설명한다.

A.1 블루투스 로봇 인터페이스 클래스 정의

```
//추상화 클래스인 AbstractRobotConnector 를 상속하여 BluetoothRobotConnector 를 구현
public class BluetoothRobotConnector extends AbstractRobotConnector {

    //로봇인터페이스를 호출하기 위한 서비스 인터페이스를 member 변수로 정의
    private RobotConnectorService mConnectorSvc = null;

    //Android Bluetooth 연결 Adaptor 를 member 변수로 정의
    private BluetoothAdapter mAdapter = null;

    //Robot Device 리스트를 member 변수로 정의
    private Map<Integer, RobotDevice> mDeviceMap = new HashMap<Integer,
RobotDevice>();

    //로봇인터페이스의 브로드캐스트 메시지들을 정의
    String INTENT_BLUETOOTH_DISCOVERY_STARTED =
"srobot.intent.action.SROBOT_BLUETOOTH_DISCOVERY_STARTED";
    String INTENT_BLUETOOTH_DISCOVERY_FOUND =
"srobot.intent.action.SROBOT_BLUETOOTH_DISCOVERY_FOUND";
    String INTENT_BLUETOOTH_DISCOVERY_FINISHED =
"srobot.intent.action.SROBOT_BLUETOOTH_DISCOVERY_FINISHED";
    String EXTRA_ROBOT_BLUETOOTH_DEVICE_INFO =
"srobot.intent.extra.ROBOT_BLUETOOTH_DEVICE_INFO";
    String INTENT_SROBOT_ROBOT_REGISTER_REGISTERED =
"srobot.intent.action.SROBOT_ROBOT_REGISTER_REGISTERED";
    String                    INTENT_SROBOT_ROBOT_REGISTER_NONE                    =
"srobot.intent.action.SROBOT_ROBOT_REGISTER_NONE";
```

A.2 블루투스 로봇 인터페이스 최초 활성화 init()

```
//로봇 인터페이스 최초 구동시 호출되는 init()  함수
public void init(RobotConnectorService svc) {
          mConnectorSvc = svc;

        //스마트 디바이스의 블루투스 연결에 이상이 없는지를 확인
        mAdapter   = BluetoothAdapter.getDefaultAdapter();
          if (DEBUG && mAdapter == null) {
               Log.e(TAG, "BluetoothAdapter::getDefaultAdapter failed");
               return;
        }

        //물리적인 블루투스 연결에 이상이 없다면, 기  등록된 로봇의 인스턴스를
메모리에 로드한다.
        //메모리에 로드시  BluetoothRobotDevice 의 객체를 생성하면서 물리적인 연결
connection 을  생성한다.
          if (mAdapter.isEnabled()) {
               loadManagedDevicesInfo();
        }
    }

void loadManagedDevicesInfo() {

            //등록된 로봇 리스트들의 값 ( 물리적인  저장소에서 읽어와서 배열로 처리)
            String  robotIds[] = {"SD_robot_01", " SD_robot_02", "  SD_robot_02"};
            String robotNames[] =  {"ROBOTIS DREAM", "iClebo", "ROBOMI"};

            //블루투스 연결을 지원하는 로봇의 인스턴스를 생성한다.
            for (int i = 0; i  < robotIds.length; i++) {

                //실제 인스턴스 생성 new() 함수 call
                BluetoothRobotDeviceInfo  info = new
BluetoothRobotDeviceInfo(robotIds[i], robotNames[i],  robotIds[i].hashCode());
                Intent btIntent =  new
Intent(INTENT_SROBOT_ROBOT_REGISTER_REGISTERED);
                  btIntent.putExtra(EXTRA_ROBOT_BLUETOOTH_DEVICE_INFO, info);

                //로봇의 인스턴스 생성시에 connection 까지  연결되고,
연결완료되었음을 브로드캐스팅한다.
                mConnectorSvc.sendBroadcast(btIntent);
```

```
                Log.d(TAG,    "sendBroadcast for registered device: " + info.getName()
+
                        "(handle: " + String.format("0x%04X",    info.getHandle()) +
")");
            }
}
```

A.3 블루투스 로봇 인스턴스 생성 시에 로봇 연결

```
//물리적인    HW 로봇을 형상화한 클래스, 블루투스 연결을 지원한다.
public    BluetoothRobotDevice(BluetoothDevice dev, boolean registered, boolean
paired)    {
        mHandle =    dev.getAddress().hashCode();
        mType = DEVICE_TYPE_BLUETOOTH;
        mDevice = dev;
        mName = dev.getName();
        mRobotId = dev.getName();
        mAddress = dev.getAddress();
        mPaired = paired;

        //인스턴스 생성시에 등록된 로봇이라면 인스턴스 생성시에 연결까지 실행한다.
       if (registered == true) {
           startConnection();
        }

    }

//로봇 연결은 연결 중 블로킹을 회피하기 위해    비동기 프로세스로 처리를 위해 별도
쓰레드로 처리한다..
private void startConnection() {
          setConnetingState(true);
          Thread thr = new Thread() {
              public void run() {
                  connect();
              }
          };
          thr.start();
    }

//물리적인 블루투스 연결은 RfcommSocket 연결로    진행한다.(스마트디바이스마다 상이할
수 있음)
```

```
synchronized public boolean connect() {
            setConnetingState(true);
            boolean result = false;
            if (mSocket == null) {
                try {
                  //RfcommSocket    블루투스 연결 방식의 경우
                    Method m =    mDevice.getClass().getMethod("createRfcommSocket",
new Class[]    {int.class});
                  //블루투스의 물리적인 연결을 위한 블루투스 소켓 객체 생성
                    mSocket = (BluetoothSocket) m.invoke(mDevice, 1);
                  if (mSocket == null) {
                      Log.i(TAG, "returned    socket is null");
                    }
                }
.....
          }

            if (mSocket != null) {
                if (mSocket.isConnected() == false) {
                    try {
                      //블루투스 소켓의 connection() 함수를    호출
                      mSocket.connect();
                      result = true;
.....
      }
```

A.4 로봇에 로봇 명령어를 전송

```
//물리적인 HW 로봇을 형상화한 로봇 디바이스      클래스의 함수로 writeCommand()를
정의한다.
synchronized public ConnectorResult writeCommand(ConnectorCommand cmdObj)    {
          if (!isConnected()) {
              ....
              }
              else {
                  startConnection();
                  Log.e(TAG,    "disconnected and failed but reconnecting to device");
              }
              ......
          }
```

```
        ConnectorResult result =    new ConnectorResult();
        if (isConnected()) {
            if (cmdObj != null) {

                //로봇명령어를  원시  로봇  명령어  (로봇       디바이스가  인식  가능한
명령어)로 변경한다.
                int cmd =    cmdObj.getCommand();
                byte[] rawCmd =    cmdObj.getRawCommand();

                try {

                    //물리적인 블루투스 연결 소켓의 Write() 함수를 call 한다.
                    OutputStream    outStream = mSocket.getOutputStream();
                      outStream.write(rawCmd);
                    .....
                    Thread.sleep(30);
                } catch    (IOException e) {
                    return result;
    }
```

A.5 로봇 상태 값을 리스너 이벤트를 통해 로봇 어플리케이션에 전송

```
//로봇의 센서 이벤트를 전송할 수 있는 리스너 클래스를 정의.
interface RobotSensorEventListener {
    /**
     * Called when sensor values    have changed
     */
    void onSensorChanged(RobotSensorEvent    event);

}

//SmartDeviceRobotManager    Class 정의
//로봇어플리케이션에서  스마트디바이스  기반  로봇의  로봇인터페이스등을  사용하기  위한
Context Manger Class.
public class SmartDeviceRobotManager {

    .........
    private Context mContext;
    private ISmartRobotService    mService;
    private ServiceConnection    mServiceConnection;
    private ServiceConnection    mServiceConnectionNotify;
```

```
    ........

    //로봇으로부터 전달되는 로봇 센서 이벤트를 수신하기 위해서 call 하는 함수
    public boolean    registerSensorEventListener
    (final RobotSensorEventListener    listener, int sensorType, int rate, int connID)
    throws RemoteException {

            //실제 로봇 디바이스에서 전송되는 로봇 센서 이벤트를 수신하기 위하여
stub() 함수를 통해 인스턴스화
            mSensorEventListener    = new IRobotSensorEventListener.Stub() {

            //RobotSensoerEventListner 의 onSensorChanged()를 통해서 수신되는
로봇 센서 값이 있을 때 마다 callback 된다.
            public void onSensorChanged(RobotSensorEvent event) throws
RemoteException {
                    listener.onSensorChanged(event);
            }


         ....
        };

       //백그라운드에서 수행되는 로봇 인터페이스가 다수의 로봇 어플리케이션에 센서
이벤트를    전달하기 위한 Registration
        return mService.registerSensorEventListener(mSensorEventListener,
sensorType, rate, connID);
    }
```

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 스마트디바이스와 로봇 간의 인터페이스 및 서비스 - 제1부 : 용어 및 정의

시장에서는 점차 가전제품과 로봇 기술을 융합한 모델, 전화기에 홈모니터링 등의 서비스를 지원하는 모델 등 다양한 아이디어의 제품과 서비스를 결합하여 새로운 시스템이 만들어 지고 있다. 따라서 이 표준에서는 커지고 있는 스마트 디바이스 시장에 로봇 기술을 융합한 스마트 디바이스와 로봇 간의 인터페이스 및 서비스에 관한 용어를 정의하고자 한다.

Ⅰ-3.2 스마트디바이스와 로봇 간의 인터페이스 및 서비스 - 제2부 : 로봇 인터페이스

시장에서는 점차 가전제품과 로봇기술을 융합한 모델, 전화기에 홈모니터링 등의 서비스를 지원하는 모델 등 다양한 아이디어의 제품과 서비스를 결합하여 새로운 시스템이 만들어 지고 있다. 따라서 이 표준에서는 점차 커지고 있는 스마트디바이스와 로봇기술을 융합한 새로운 시장 활성화를 위해서 스마트디바이스와 로봇 간의 표준 인터페이스 규격을 정의하였다.

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)