HOVIS Genie/App으로 배우는

Android Robot Programming

기초부터 재미있게 배우는
로봇 프로그래밍의 원리

동부로봇 자료실

(매뉴얼 및 어플리케이션을
다운로드 받으십시오.)

www.dongburobot.com

Dongbu Robot

Contents

01 안드로이드 로봇 Hovis Genie 소개

Android Robot Program-ming

02 안드로이드 및 로봇 개발환경 구축하기

2.1 JDK 설치

2.2 안드로이드 SDK설치

2.3 이클리스 설치 및 환경설정

03 안드로이드 로봇(지니) 프로그래밍

3.1 HelloGenie 프로젝트 생성

3.2 HelloGenie 프로젝트의 실행

3.3 HelloGenie 프로젝트의 기본 구성

(1) AndroidManifest.xml

(2) MainActivity.java와 activity_main.xml

01 안드로이드 로봇 HOVIS Genie 소개

1.1 HOVIS 제작 배경

기존에 개발되었던 서비스로봇은 안내소등에서 볼 수 있는 크기가 크고, PC 용 터치 스크린이 장착되어 유저들이 서서 작동을 할 수 있게 시범적으로 만든 로봇이 거의 전부였습니다. End User 즉 최종 소비자가 직접 구매한다기 보다는 기관이나 정부의 사업으로 진행되는 경우가 대부분이였습니다.

이러한 형태의 로봇은 서비스 내용이 극히 제한되고, 공급자, 즉 개발회사에서 직접 프로그래밍 할 수 밖에 없는 구조를 가지고 있습니다. 이러한 제한을 극복하고 로봇 서비스의 확장을 위해서 Hovis 라는 안드로이드 로봇을 제작하게 되었습니다.

HOVIS 는 Home Service 의 약자로 홈서비스에 실제로 활용되는 로봇을 제작하기 위해 만든 로봇입니다. 실제 집안에서 사용될 수 있기 위해서는 집안에서 발생할 수 있는 모든 형태의 서비스를 아우르는 소프트웨어가 필요합니다.

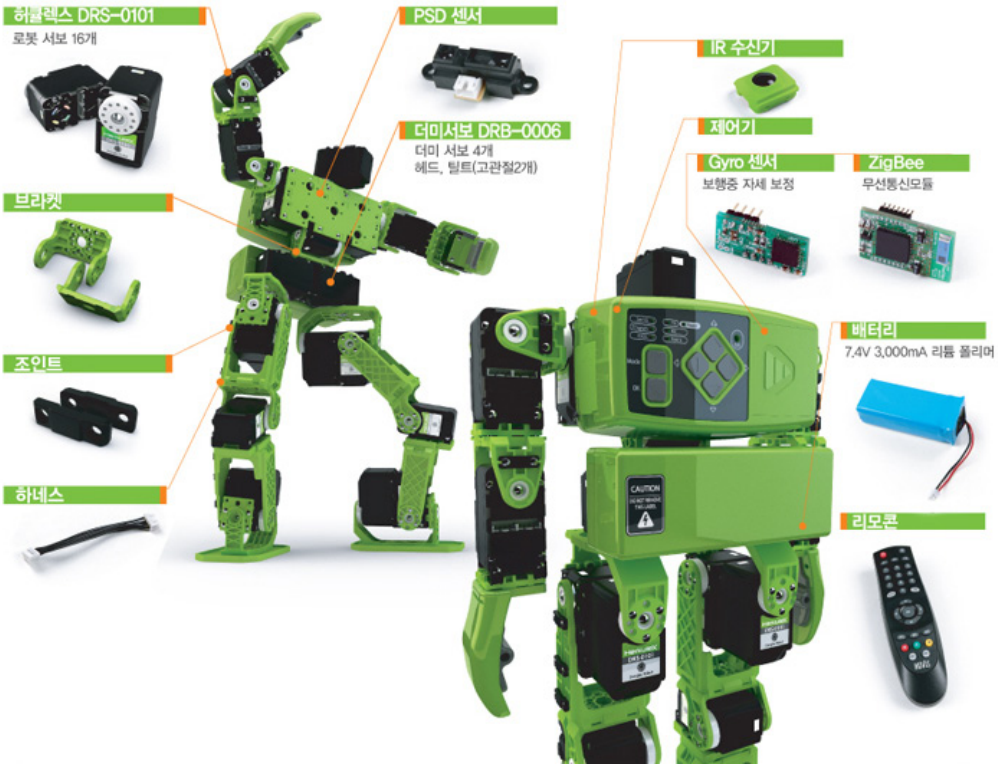
그러한 서비스를 만들기위해서, 현재 스마트폰의 소프트웨어 공급 방식을 받아 들여, 로봇에도 그대로 적용한 것이 바로 Hovis Genie 입니다. 필요한 서비스는 현재 안드로이드 플랫폼을 이용하여 다수의 엔지니어에 의해서 공급될 수 있는 구조입니다.

(주)동부로봇은 안드로이드 단말기가 장착된 안드로이드 로봇을 공급하고, 그 로봇에서 안드로이드 로봇 서비스를 제작할 수 있는 소프트웨어, 즉 API 와 프로그래밍 방법을 제공하고 있습니다.

(1) Hovis Lite

Hovis Genie 이전에 Hovis Lite 에 대한 간단한 설명이 필요합니다.

1.2 HOVIS Genie의 특징



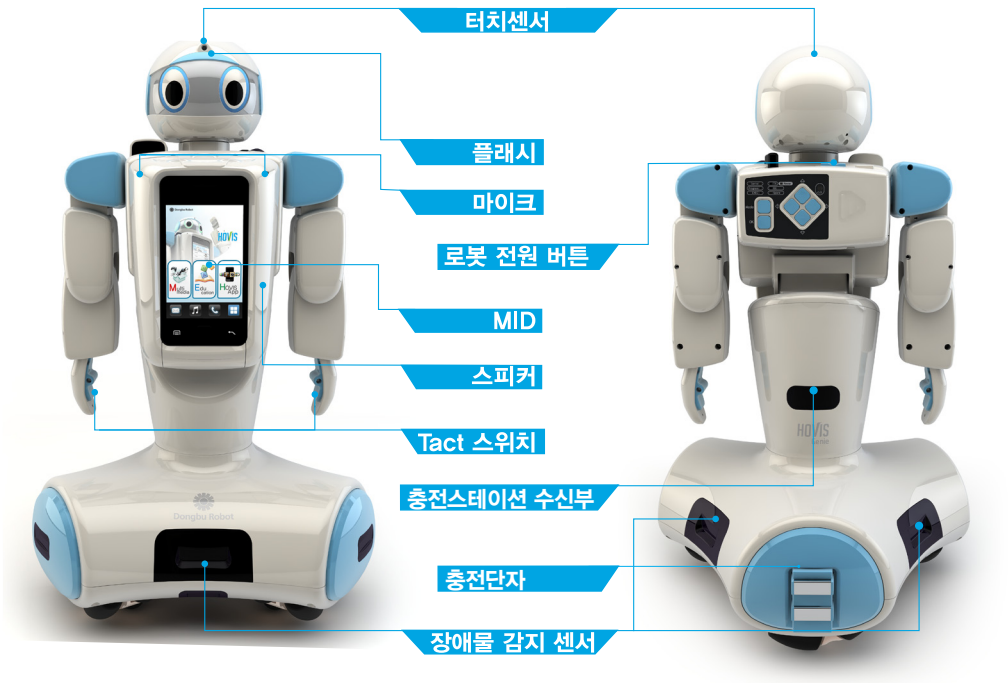
호비스 라이트는 마이크로프로세서인 ATMega 128 이 사용된 8bit 제어기 가 장착된 휴머노이드 로봇입니다. 서보모터, 갖가지 센서, 다양한 변신형 로봇을 통해서, 로봇 자체 교육을 목적으로 제작된 로봇입니다.

위 8bit 제어기를 DRC 라고 부르며, 호비스 라이트의 두뇌 역할을 합니다. DRC와 모터, 센서를 제어하기 위한 소프트웨어 툴을 제공하는데, 그것이 바로 DR-Visual Logic 입니다. C 언어처럼 변수선언과 함수 구현, 로직 구성이 가능한 그래픽 개발 랭귀지 입니다. 또한 16축이 사용되는 호비스 휴머노이드의 복잡성을 감안하여 쉽게 모션을 제작할 수 있도록 DR-SIM 이라는 모션 제작 툴을 제공하고 있습니다.

위 호비스라이트의 다리를 옴니휠로 바꾸고, 커버를 씌운 후 안드로이드 단말기를 장착한 것이 호비스 지니입니다. 따라서 호비스 지니는 안드로이드 단말기를 통해서 DRC 와 각종 센서를 제어하게 됩니다.

(2) Hovis Genie

호비스 지니는 기본적으로 홈서비스를 목적으로 탄생한 로봇입니다. 안드로이드 단말기를 채용하여 각종 서비스 어플리케이션을 앱 형태로 실행시키면 목적에 맞는 서비스가 구현됩니다.



호비스 지니에는 머리부분에 터치센서, 플래시, 눈 LED 를 통해서 감정의 표현을 얼굴로도 할 수 있고, 팔쪽에는 Tact 스위치를 장착하여 손 터치를 통한 시나리오도 가능합니다. 지니의 편리함과 안정감은 하단부의 옴니휠 바퀴에 있습니다. 사방 360도 어느 방향으로도 우회전 없이 직접적으로 갈 수 있고, 자동충전에 적합하며, 교육용으로도 활용이 가능합니다. 옴니휠 주변에는 거리센서, 바닥감지 센서등이 있어서 로봇이 장애물을 피하는데 유용하게 쓰일 수 있습니다.

호비스 지니는 조립형과 완성형 두가지 형태로 제공합니다. 조립형은 안드로이드를 통한 로봇 교육에 활용하기 위한 목적으로 호비스 라이트처럼 처음부터 하드웨어를 교육생이 직접 조립한 후 소프트웨어를 조작해보라는 의도입니다. 완성형은 아파트등 홈서비스에 직접적으로 들어가기 위한 목적으로 일반 사용자가 프로그래밍 없이 안에 있는 어플을 활용하여 사용하게 하자는 목적입니다. 하지만, 완성형도 안드로이드 교육에 활용이 가능합니다. 조립형의 조립매뉴얼은 (주)동부로봇 홈페이지 자료실과 www.hovis.co.kr/guide 사이트에서 다운로드 받아 사용이 가능합니다.

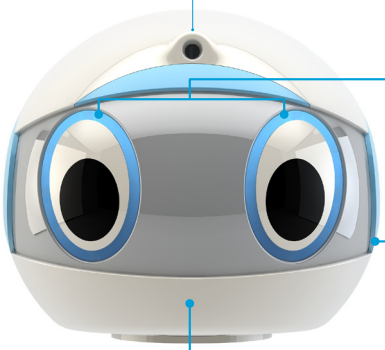
1.3 HOVIS Genie 의 하드웨어적인 구성

호비스 지니는 기본적으로 로봇 본체와 MID 로 크게 구분됩니다. MID는 탈부착이 가능하고, 탈착시에는 일반 안드로이드 단말기로 활용이 가능합니다.

(1) 로봇 본체

하드웨어에는 로봇의 기본 구성이라고 할 수 있는 모터, 브라켓, 거리센서, IR통신기 등이 있고, 외부적으로 기구물이 감싸고 있는 형태입니다. 자동충전이 가능한 충전스테이션과 어댑터, 리모콘 등 악세사리가 기본 제공됩니다.

지니의 입력파트는 MID 의 터치와 음성인식, 머리부분의 터치센서, 손부분의 Tact 스위치, 바퀴부분의 거리센서와 바닥감지 센서 등을 통하여 로봇에 외부값이 입력이 됩니다. 출력파트는 MID 의 화면, 음성, 음악 과 머리부분의 눈 LED, 모터를 기본으로 한 로봇 동작 등으로 로봇의 출력과 표현을 담당합니다. 이 두가지 입출력을 바탕으로 로봇 서비스가 이뤄지며, 프로그래밍도 가능하게 됩니다.



로봇 전원 버튼

머리를 쓰다듬으면 지니가 느끼고 반응해요.

자율모드에서 이동 시 머리를 터치하면
지니가 일정시간 동안 동작을 멈춰요.

눈(LED)

충전 중일 때 충전 상태에 따라 눈 가장자리 색이 변해요.

- 충전완료 시 : 파랑
- 충전 중 : 빨강
  (충전될수록 빨간 선 길어짐)

귀(LED)

- 평상 시(자율/정지 모드) 양쪽 귀가 흰색으로 켜져 있어요.
  (깜박거리지 않음)
- 음성인식모드일 경우 인식 가능 상태일 때 깜박거려요.

입(LED)

지니가 말을 하거나 감정이 변할 때
1~3개의 흰색 LED가 깜박거려요.

(3) 호비스 지니 스펙

호비스 지니는 어른 무릎 높이 정도의 크기입니다. 자동충전이 가능하고, MID 를 통한 카메라 원격제어가 가능합니다. 세부 스펙은 아래와 같습니다.

	항목	세부 사양
로봇	모델명	HOVIS Genie
	크기	255(W) X 236.5(D) X 408.5(H)
	무게	3.4kg
	모터	11개
	센서	장애물 감지 센서, 낭떠러지 감지 센서, IR 센서, 터치 센서, Tactile 센서, 위치감지 센서
	이동방식	Omni-directional Drive / HerkuleX 0102 3ea
	네트워크	IR Receiver, ZigBee[option]
	외장 스피커	External 2 way stereo Speaker(1W 2Ch)
	배터리 용량	7.4V / 3,000mAh / 리튬 폴리머
MID	디스플레이	3.5"TFT(480X320)
	화면 입력 방식	정전식 Full Touch
	메인 프로세서	MID : S5PC110, 1GHz
	카메라	Front Rear Camera
	오디오	1ch MIC(MID 내장)
	내장 메모리	256MB
	외장 메모리	Micro SD 4G
	네트워크	Wi-Fi
기타	리모콘	IR 방식 / ZigBee Remocon[option]

1.4 HOVIS Genie 의 소프트웨어적 특징

호비스 지니는 개발 측면에서 두가지의 소프트웨어적인 특징이 있습니다. 로봇의 제어기와 센서 제어 보드에 들어가는 펌웨어와 MID 어프리케이션 개발을 위한 응용 소프트웨어 입니다. 펌웨어는 C 언어로 작성하는 AVR 프로그래밍이고, MID 응용 소프트웨어는 Java 를 활용하는 안드로이드 프로그래밍입니다. (주) 동부로봇에서 안드로이드 엔지니어를 통한 로봇 서비스용 API 를 제공합니다.

(1) MPSU (DRC-005T), OPSU (DRC-004TO), MID Side Board – 펌웨어

위 보드는 로봇의 등과 내부에 장착되어 로봇의 모터와 센서를 제어하게 됩니다. 주 명령은 MID 로 부터 받으며, MPSU와 OPSU에는 관련된 펌웨어 프로그램이 임베디드 되어있습니다.
MPSU 는 DRC 라고 불리며, 호비스 라이트의 DRC 와 동일합니다. MID에서 오는 명령을 받아 모터와 머리 LED, MID Side Board 등을 제어하는 역할을 하며 머리의 터치 센서, 어깨의 거리 센서 기능도 수행합니다.
OPSU 는 호비스 지니 배 안쪽에 있는 보드로서, 바닥 감지센서, 거리센서와 충전 스테이션의 IR 센서 등의 값을 제공합니다.
MID Side Board는 호비스의 가슴과 MID 밑에 장착된 보드로, 지니의 스피커, MID 충전, MID 와 MPSU의 통신 레벨 변환 등의 기능을 수행합니다.
MPSU 와 OPSU 는 ATmega128 마이크로프로세서 가 들어가 있습니다. AVR 프로그래밍에 의해 펌웨어를 생성시킵니다. 호비스 라이트에서는 MPSU, 즉 DRC AVR firmware 프로그래밍 방법과 교재를 제공하고 있습니다. 하지만 호비스 지니는 MID 에서 서비스 구현을 목적으로 하기 때문에 제조사에서 임베디드 시킨 펌웨어를 그대로 사용하게 됩니다. 따라서 다른 엔지니어나 유저가 펌웨어 변경은 금지하고 있습니다.

(2) MID – Android

MID는 Multimedia Internet Device 의 약자로, 안드로이드 OS 이전에도 많은 스마트 미니 단말기가 있었습니다. (주)동부로봇 MID 는 안드로이드 OS 장착하고, 기본 안드로이드 마켓에서 통용되는 어플리케이션 다운로드 신규 서비스를 받을 수 있게 만든 단말기입니다. 기존 단말기와 다른 점은 호비스 지니 로봇 본체를 조작할 수 있는 내부 로봇 서비스 프로그램이 들어 있다는 것입니다. 그러한 내용은 본 책 2장부터 설명할 것입니다.
MID 는 시스템소프트웨어(펌웨어)와 어플리케이션으로 구분될 수 있습니다. 안드로이드 OS가 리눅스 기반의 시스템으로 단말기 자체 펌웨어를 구성합니다. (주) 동부로봇에서는 시스템소프트웨어 프로그래밍에 대한 교재는 추후에 편찬을 결정할 예정입니다.
호비스 지니 소프트웨어의 궁극적인 목적은 MID 의 펌웨어가 아니라, MID 의 안드로이드 어플리케이션 개발입니다. JAVA 언어로 개발되는 안드로이드 어플리케이션은 이미 안드로이드 마켓 등에서 수천만개가 판매되고 있습니다. 똑같은 방식으로 호비스지니 로봇의 안드로이드 어플리케이션을 개발하여 MID 에 다운로드하면 단말 서비스가 아니라 로봇 서비스를 실행할 수 있습니다.

1.5 HOVIS Genie 어플리케이션 실행하기

호비스 지니의 MID 를 켜면 안드로이드 단말기 내부 프로그램이 아닌 호비스 관련 프로그램이 바로 메인화면에 나타납니다. 기본 안드로이드 어플은 뒤로 보내고, 로봇관련 대메뉴로 재 구성한 호비스지니용 런처(시작) 프로그램입니다.

(1) 메인화면

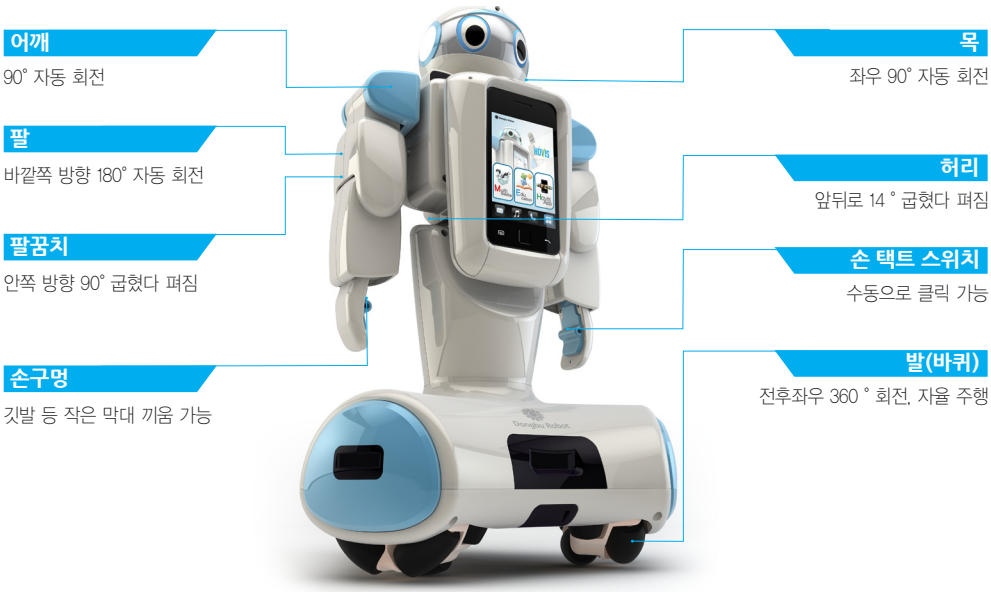
로봇을 켜면 화면에 로봇 심장 그림과 소리가 들리면서 곧 바로 메인화면으로 이동합니다.
메인화면은 상단에 대메뉴 3개와 하단에 바로가기 버튼과 홈과 로봇 변환 소프트버튼으로 구성되어있습니다.

대메뉴에는 카메라나 사진, 소리관련된 멀티미디어 메뉴와, 아침기상정보, 원격제어등 로봇과 관련된 호비스앱 메뉴, 어린이들이 볼 수 있는 동화,동요 등 콘텐츠를 모아놓은 교육 메뉴로 구성되어 있습니다. 멀티미디어는 단말기 자체의 기본 어플리케이션중에 로봇에 필요한 멀티미디어적인 것을 모아놓은 것이고, 호비스앱은 로봇과 관련된, 즉, 로봇의 동작이나, LED, 터치, 음성 등 로봇 요소를 활용하여 개발된 로봇 기능활용용 어플리케이션을 모아놓았습니다. 그리고, 호비스지니는 기본적으로 홈로봇 컨셉입니다. 홈에서 아이들의 교육에 활용될 수 있는 콘텐츠인 동화 동요 등은 플래쉬나 간단한 어플로 대량 제작하여 배포하고 있습니다. 두 번째 호비스앱과 세번째 교육 콘텐츠 앱은 (주)동부로봇 MID 용 앱스토어에서 재 다운로드가 가능하고, 다른 어플리케이션과 콘텐츠도 접속하여 다운로드 받을 수 있습니다.

그리고 기존에 받은 어플리케이션도 업데이트 파일이 있으면 재 다운로드가 가능합니다. 아직은 외부 공개용 앱스토어는 아니지만, 동부로봇용 간단한 앱스토어는 구축되어 있습니다.

(2) 자율행동

흔히 사람들은 로봇하면 TV 나 영화에 나오는 아톰류나 터미네이터 같은 로봇을 상상하게 됩니다. 그러나 그러한 로봇은 실용성 측면에서 많이 떨어질 수 밖에 없고, 현실적으로 구현하는 것도 불가능합니다. 따라서 가장 실용적으로 쓰이는 로봇을 뽑자면 바로 산업용 로봇이라 볼 수 있습니다. 산업용 로봇은 눈도 없고 다리도 없습니다. 오직 팔만 있고, 정확한 위치에 정확한 행동으로 물건을 만들어냅니다. 일종의 자동화 기계라고 보는 것이 맞습니다.
호비스 지니는 그런 자동화 산업 로봇보다는 TV 영화의 아톰류같은 휴머노이드 로봇입니다. 당연히 실용성은 떨어지지만, 홈서비스를 통해 그 실용성을 높이기 위해 만든 로봇이기 합니다.
아톰류의 로봇은 사람과 거의 비슷하게 행동하고 움직입니다. 그것은 만화이기 때문에 가능한 것입니다. 현실적으로 만든 로봇중에 사람과 가장 유사한 로봇은 아직 없다고 봐야합니다. 로봇이 사람같다는 의미는 사람이 하는 행동을 따라한다는 것인데, 그 중 가장 뛰어난 기능이 자율적으로 행동하는 로봇일 것입니다.
자율로봇 기능 로봇의 모터, 터치센서, 바닥감지 센서 등을 이용하여 로봇이 일정시간이나 어떤 반응에 자율적으로 움직일 수 있는 모든 데이타를 미리 넣어놓고 반응하게 만드는 것입니다. 하지만 그 기능은 제한 될 수 밖에 없습니다.



1 장애물을 만났을 때 행동

주행 중 장애물을 만나면 제자리에서 몸을 돌려 다른 곳으로 이동합니다.

2 머리를 터치했을 때 행동

1 쓰다듬으면 기분이 좋다는 반응을 보입니다.

2 머리를 때리면 기분이 안 좋다는 반응을 보입니다.

3 배고플 때 행동

배터리가 20% 남아있으면 자동충전을 실시합니다.

4 시간대별 행동

1 아침에는 오늘 날씨를 물어보며 좋은 하루를 보내라고 말합니다.

2 점심에는 즐거운 점심시간이라고 말하며 점심은 먹었는지 물어봅니다.

3 저녁에는 오늘 날씨는 어땠는지 오늘 무엇을 하며 지냈는지 물어봅니다.
또한 저녁 9시가 되면 자야할 시간이라고 말하고 10시가 되면 잠잘 준비하자고 말하며 양치질이나 세수를 하는 동작을 합니다.

5 자율행동

심심해하기, 사랑 표현하기, 애교 부리기, 시간 묻기, 혼자 놀기, 기지개 펴기, 머리긁기, 체조하기, 지휘하기 등 여러 가지 행동과 말을 합니다.

6 손 택트 스위치를 클릭했을 때 행동

1 **왼쪽 손 클릭 시 –** '안녕'하고 인사를 하거나 뽀뽀하며 네가 좋다는 긍정적인 감정표현을 합니다.

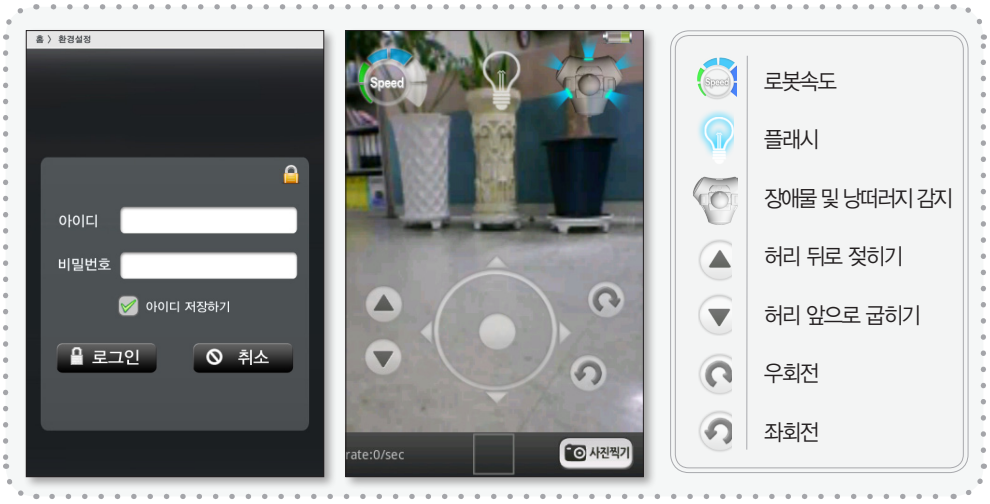
2 **오른쪽 손 클릭 시 –** 손목시계를 보는 시늉을 하며 현재 시간과 요일을 이야기합니다.

3 **양 손 함께 클릭 시 –** 양 손바닥 사이에 가볍고 탄력있는 공을 끼워 주면 공을 머리 위로 올린 후 공을 던져 줍니다 .

장애물을 만났을 때는 하단 전방 거리센서가 작동하고, 머리를 만졌을 때 행동은 머리쪽 터치센서가 반응하는 것입니다. 밧데리가 떨어지면 자동을 충전기로 이동합니다. 또한 시간대별로 행동 패턴을 다르게 구성한다든지, 무작위로 춤을 추게 할 수 도 있습니다.
자율행동 또한 로봇 프로그래머에 의해 얼마나 세부적으로 제작하느냐에 따라서 로봇의 컨셉이 자율행동 컨셉이 정해질 것입니다.

(3) 원격제어

현시대에 로봇을 제어하는 방법 중에 가장 많이 활용되는 것이 로봇의 원격제어입니다. 집안에서 쓰는 청소기에 카메라를 이용하여 집안 원격 모니터링과 제어를 하기도 하고, 화성에 로봇을 보내서 원격제어로 화성탐사로봇을 조정하기도 합니다. 2004년에 NASA 가 화성에 보낸 오퍼튜너티는 7년 넘게 화성에서 탐사 임무를 완수했습니다. 탐사방식은 오퍼튜너티에 달려있는 카메라 영상을 지구에서 받아서 지구에서 다시 명령을 내리는 방식입니다. 전파 전달 속도는 빛의 속도와 같으므로, 화성까지 명령이 전달되는 시간은 4분 27초라고 합니다. 화성탐사로봇이든, 청소기 로봇이든 카메라를 이용한 원격제어 방식은 모두 동일하다고 보면 됩니다.



마찬가지로 호비스지니도 MID 에 장착된 전면부 카메라를 통해서 원격제어가 가능합니다. 소비자가 가지고 있는 스마트폰에서 호비스지니 제어 어플리케이션을 설치하고, ID 로 로그인하면, 중계서버를 거쳐서 호비스 지니 MID 와 연결됩니다. MID 화면은 원격제어중이라고 표기됩니다. 스마트폰에서 이동관련 UI 를 통해서 호비스 이동을 시킬 수 있고, MID 에 저장된 로봇 모션을 실행 시킬 수 있습니다. 또한 제어 어플에서 문자를 보내는 기능이 있는데, 문자를 써서 MID 에 보내면 MID 에서는 TTS로 음성이 출력됩니다. 외부에서 보면 로봇이 말하는 것처럼 보입니다.
원격제어 어플리케이션은 다른 어플리케이션 과 다르게 두 개의 어플리케이션이 필요합니다. 하나의 MID 에 설치되는 Recieve 어플과 스마트폰에 설치되는 제어 어플입니다. 현재 거의 모든 어플리케이션은 MID 에 설치되어 로봇 내부 시스템과 연동하는 프로그래밍을 해줘야합니다. 하지만, 스마트폰에 설치되는 원격제어 어플리케이션은 일반 스마트폰 어플 개발과 동일하게 개발하면 됩니다. 따라서 일반 엔지니어가 가장 빨리 제작할 수 있고, 손쉽게 자신의 스마트폰에서 조작해 볼 수 있습니다.
제조사 제공 제어 어플리케이션 아닌, 일반 엔지니어들이 개발하는 제어 어플리케이션은 훨씬 더 다양한 형태로 나올 수 있습니다. 모션제어와 함께 음악과 음성을 동시 출력한다든지, 스마트폰 자이로 센서를 이용하여 로봇을 이동시킨다든지, 하는 다양한 서비스가 가능한 제어 어플리케이션이 나올 수 있을 것입니다.

(4) 아침기상정보/동요동화 등 정보콘텐츠 로봇콘텐츠화

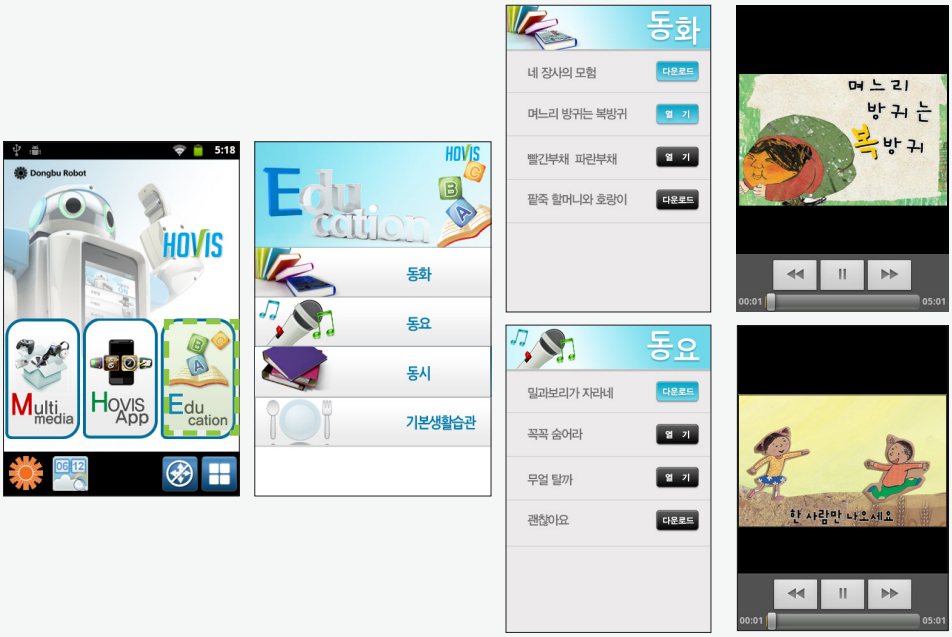
로봇은 움직이는 단말기라고 볼 수 있습니다. 로봇을 통해서 얻어지려는 서비스는 궁극적으로 이 부분입니다. 유저가 활용하고자 하는 정보를 로봇으로 가공된 어플리케이션을 대량으로 유입 가능한 서비스 생태계를 만들어야만, 유저들은 로봇 서비스를 골라 취할 수있고, 더욱 간편한 서비스가 등장하게될 것입니다.

대량 서비스 생산을 위해서는 현재 스마트폰에서 사용되어지는 유용한 어플을 로봇과 결합하여 대량으로 유입하는 방식입니다. 아침기상정보가 그 사례입니다. 스마트폰 어플중에 알람, 일정관리, 날씨을 합하고, 로봇의 움직이는 기능을 결합한 서비스 입니다. 개별적인 어플로 보면 스마트폰과 다를바 없지만, 예를 들어 아침에 로봇이 알람을 울리면서 방안을 시끄럽게 돌아다닙니다. 그런 후 오늘의 날씨를 얘기하고, 유저가 기록해놓은 일정을 차례대로 말을 합니다. 일종의 집안의 비서 역할을 할 수 있는 것입니다.

가장 많은 콘텐츠가 있다고 볼 수 있는 교육 콘텐츠 중에 동화 동요 등 어린이들에게 활용될 수 있는 어플들은 로봇에 약간의 동작만 가미해주면 재미있는 입체 어플로 재 탄생하게됩니다. 어린이들에게는 로봇자체도 흥미꺼리이기 때문에 동화 동요에 더 집중할 수 있습니다.

정보단말기 콘텐츠와 로봇콘텐츠는 동작의 유무에 따라 구분될 수 있을 정도로 그 차이는 미미합니다. 하지만 결합된 형태의 정보는 유저들에게 더 많은 효과를 가져다 줄 수 있습니다. 이러한 아이디어들이 제조사, 개발사가 아닌, 일반 엔지니어들이 참여하여 만든다면, 더 다채롭고 다양한 콘텐츠가 만들어질 것으로 예상됩니다.

[아침기상정보 콘텐츠]



[동화, 동요 콘텐츠]

(5) 소프트웨어 제공 취지 및 활용

호비스 지니 본체 + 안드로이드 Genie API + 모듈별 제어 예제 및 교재 를 통하여 호비스 지니 어플리케이션을 개발할 수 있는 환경을 제공할 것입니다. 이 로봇은 기본적으로 홈서비스용 타켓으로 만들어 졌지만, 안드로이드 어플을 통하여 다양하고 많은 수의 어플리케이션의 재생산과 보급을 위해서 일반 엔지니어에 대한 교육이 필요하며, 그 교육생으로 하여금 호비스 지니용 어플리케이션이 많이 만들어질 수 있게 본 교구재를 활용할 수 있게 환경을 제공할 예정입니다.

호비스 지니 로봇 서비스 소프트웨어는 크게, 자율행동, 원격제어, 정보가공 으로 나눠질 수 있습니다. 이 세가지 기본 예제는 (주)동부로봇에서 제공하고, 이 소프트웨어를 샘플로 해서, 다양한 소프트웨어들이 다시 가공되어 만들어지길 기대하면서 안드로이드 지니에 대한 소개를 마칩니다.

02 안드로이드 및 로봇 개발 환경 구축하기

이번 장에서는 안드로이드 및 로봇 개발환경 구축에 대해서 설명합니다.

안드로이드 및 로봇 개발환경 구축은 크게 다음과 같이 네 단계로 이루어집니다.

- JDK의 설치
- 안드로이드 SDK 설치
- 이클립스 설치 및 환경설정
- MID 개발보드의 연결 및 디버깅

2.1 JDK 설치

JDK(Java Developmenet Kit)은 자바개발도구입니다. 안드로이드는 기본적으로 자바기반으로 작성되었습니다. 따라서 자바개발도구의 설치가 필요합니다. JDK의 설치는 안드로이드 개발환경구축에 영향을 미치므로 선행되어야 합니다.

01

자바 사이트로 이동하여 Java Download 아이콘을 클릭

http://www.oracle.com/technetwork/java/javase/downloads/index.html

02

Java SE Development Kit 항목에서 라이센스 동의 후, 자신의 OS에 알맞은 JDK 다운로드
(예 : Windows 64bit는 Windows x64, Windows 32bit는 Windows x86)

Java SE Development Kit 7u5

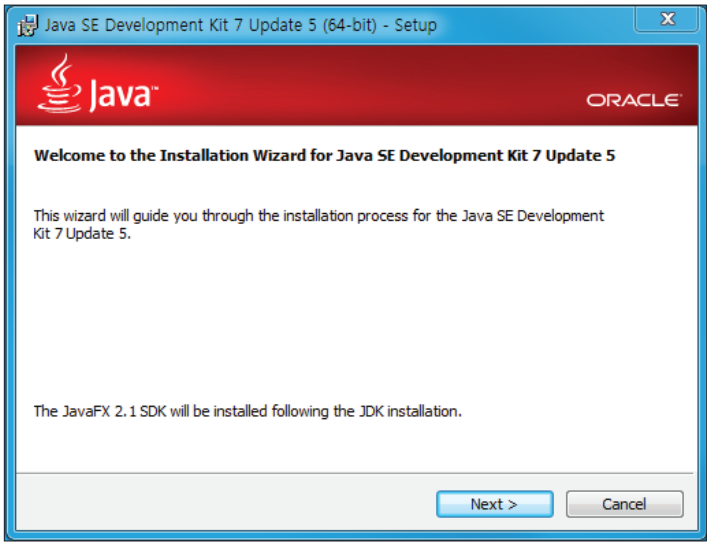
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	64.1 MB	jdk-7u5-linux-i586.rpm
Linux x86	79.1 MB	jdk-7u5-linux-i586.tar.gz
Linux x64	64.93 MB	jdk-7u5-linux-x64.rpm
Linux x64	77.67 MB	jdk-7u5-linux-x64.tar.gz
Macosx-x64	97.3 MB	jdk-7u5-macosx-x64.dmg
Solaris x86	137.41 MB	jdk-7u5-solaris-i586.tar.Z
Solaris x86	82.01 MB	jdk-7u5-solaris-i586.tar.gz
Solaris SPARC	140.43 MB	jdk-7u5-solaris-sparc.tar.Z
Solaris SPARC	86.72 MB	jdk-7u5-solaris-sparc.tar.gz
Solaris SPARC 64-bit	16.45 MB	jdk-7u5-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	12.55 MB	jdk-7u5-solaris-sparcv9.tar.gz
Solaris x64	14.92 MB	jdk-7u5-solaris-x64.tar.Z
Solaris x64	9.54 MB	jdk-7u5-solaris-x64.tar.gz
Windows x86	87.95 MB	jdk-7u5-windows-i586.exe
Windows x64	92.36 MB	jdk-7u5-windows-x64.exe

03

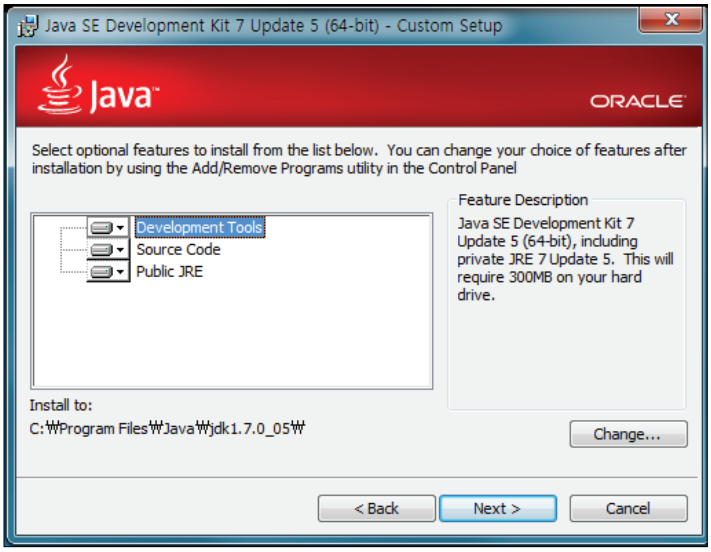
다운받은 설치파일을 실행하여 JDK 설치



04

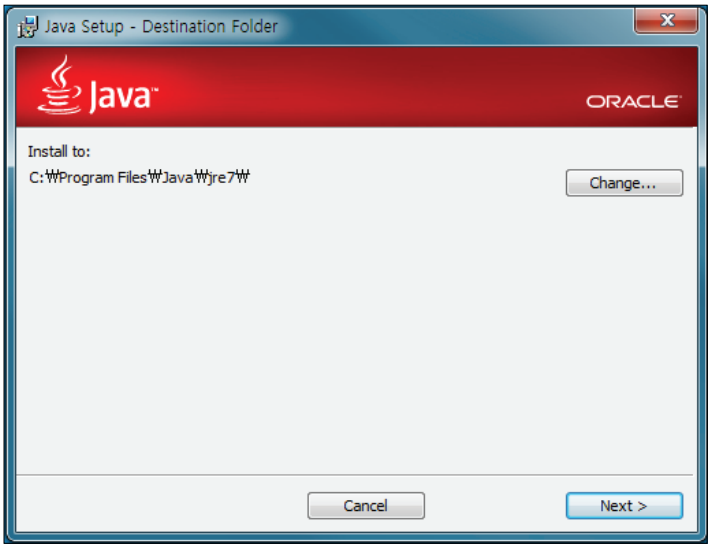
JDK 설치경로 지정

(기본 지정된 경로 사용 권장하며 한글이 포함 된 경로 지정 하지 말 것)



05

JRE(Java Runtime Environment)설치



06

설치완료

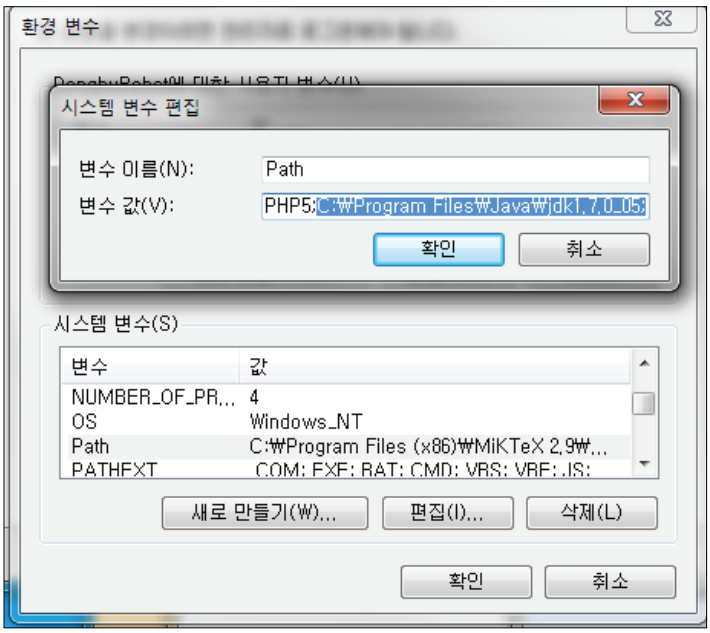
(이후 JavaFX SDK는 설치하지 않아도 됨)

07

환경 변수 설정 제어판 〉 시스템 〉 고급 시스템 설정 〉 환경변수(N) 클릭

08

4번에서 지정했던 경로 추가하기 단, 여기서는 C:₩Program Files₩Java₩jdk1.7.0_05₩



2.2 안드로이드 SDK 설치

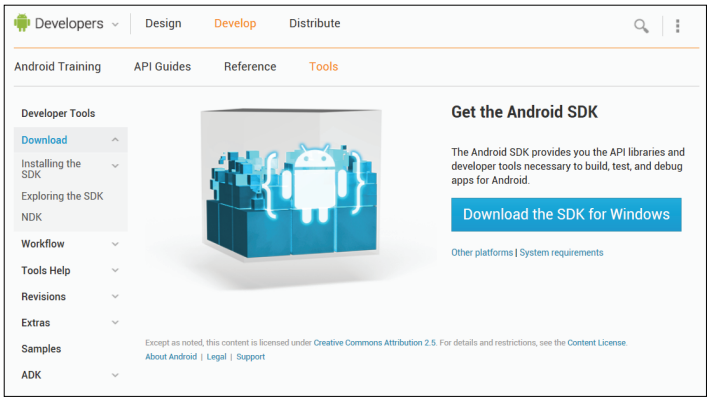
안드로이드 SDK(Software Development Kit)는 개발자가 안드로이드 기반으로 어플리케이션을 제작하기 위해 필요한 여러 라이브러리의 집합입니다. 안드로이드 SDK는 안드로이드의 공식 개발자 사이트인 http://developer.android.com/ 에서 다운로드 할 수 있습니다.

안드로이드 공식개발자 사이트인 http://developer.android.com/ 에서는 안드로이드의 최신정보, 초보자를 위한 튜토리얼, 및 API(Application Programing Interface) 사용법등의 많은 정보를 제공하고 있습니다. 안드로이드의 발전은 매우 빠르므로, 책은 최신 안드로이드 정보를 담지 않고 있는 경우가 대부분입니다. 게다가, 이 사이트는 시중의 안드로이드 기본서보다 더 많은 예제를 제공하고 있습니다.
여러분이 좋은 안드로이드 개발자로 성장하기 위해서는 이 사이트를 잘 이용하는 것이 필수입니다. 본서에서는 안드로이드를 통한 로봇 프로그래밍을 중점적으로 다루고 있습니다. 그러므로, 안드로이드 개발에 필요한 부수적인 정보를 획득하기 위하여 이 사이트를 자주 방문하기를 권장합니다.

지금부터 Android SDK설치합니다. 안드로이드 개발자 사이트를 참고하여 아래와 같이 SDK를 설치합니다.

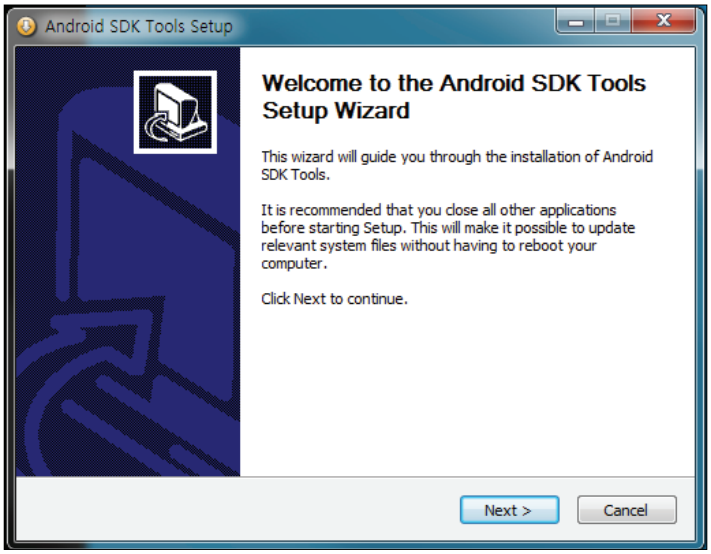
01

안드로이드 개발자 사이트 이동 후 SDK 다운로드(http://developer.android.com/sdk/index.html)



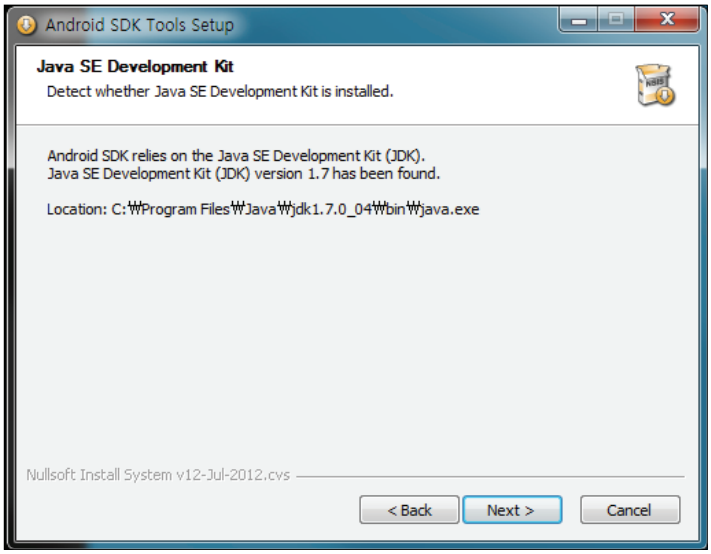
02

다운로드 받은 안드로이드 SDK 설치파일 실행 후 Next 클릭



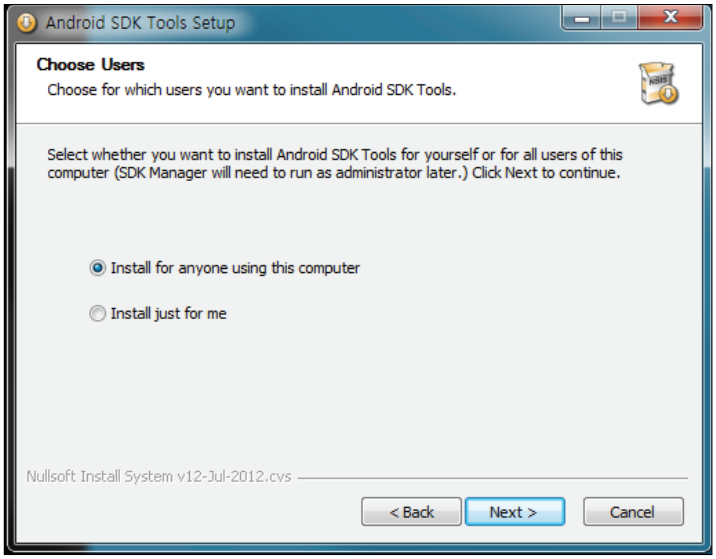
03

JDK 설치여부를 자동으로 검사한다. 설치가 되었다면 Next 클릭
(JDK를 설치하지 않았다면, JDK 우선 설치)



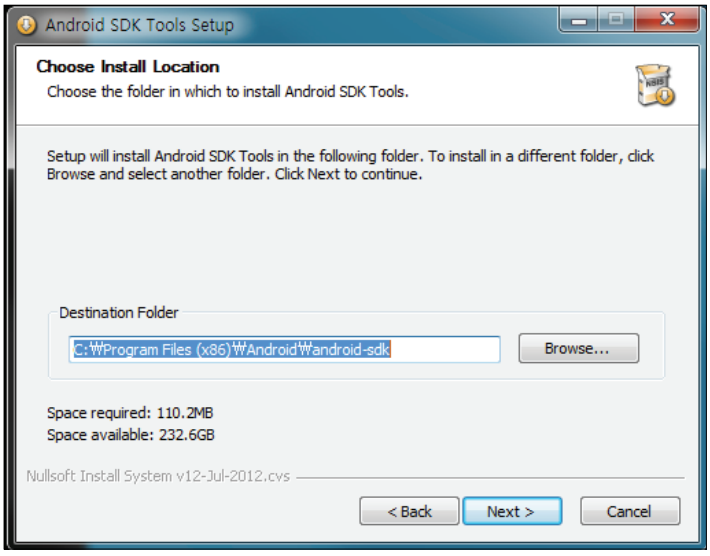
04

유저 사용 옵션을 선택하고 Next 클릭



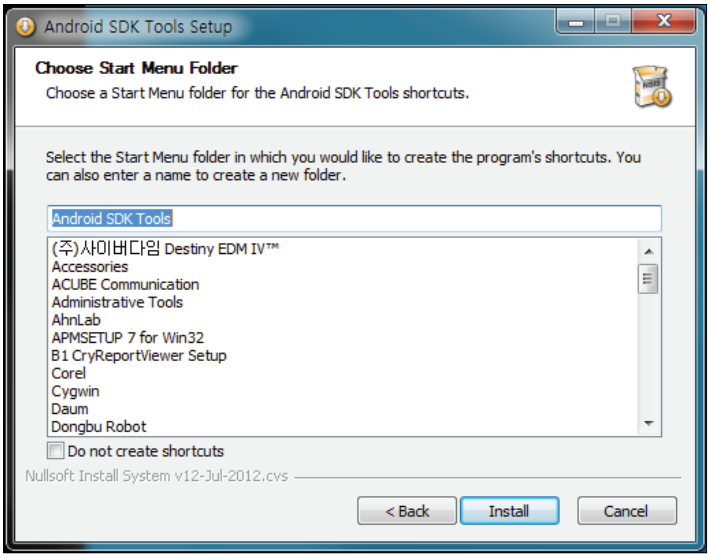
05

안드로이드 SDK를 설치할 경로 지정 후 Next 클릭
(한글 포함 된 경로를 지정하지 말 것, 기본 경로 사용 권장)



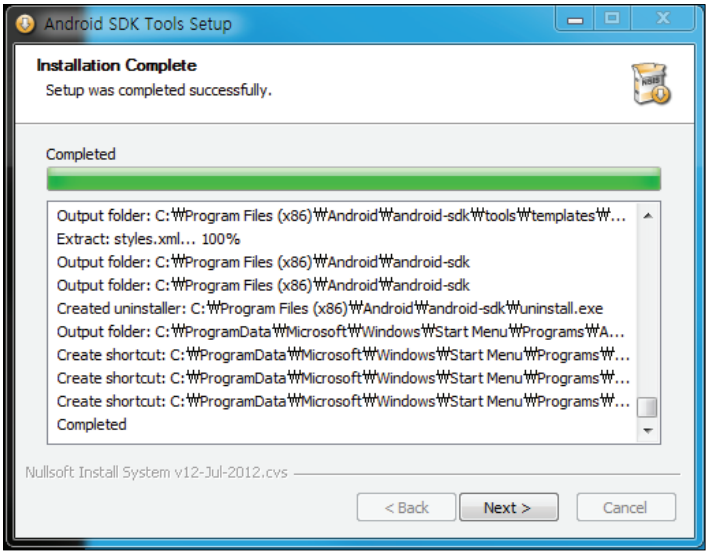
06

시작 메뉴에 등록 될 이름 선택 후, Next 클릭



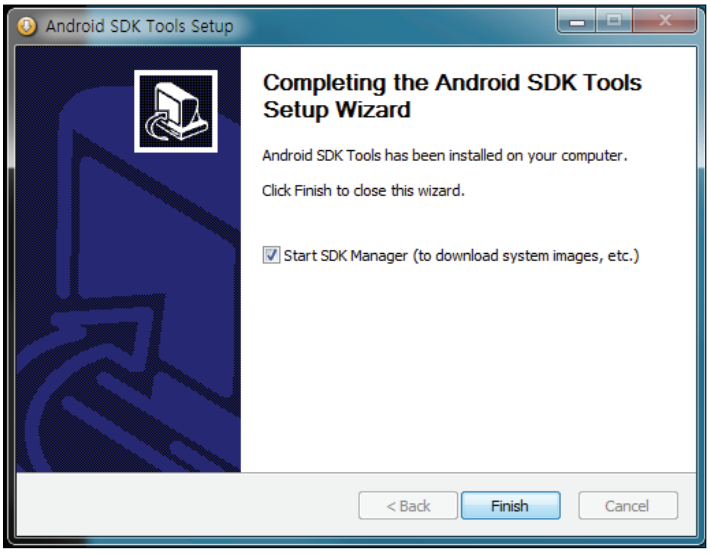
07

설치를 진행한 후, Next 클릭



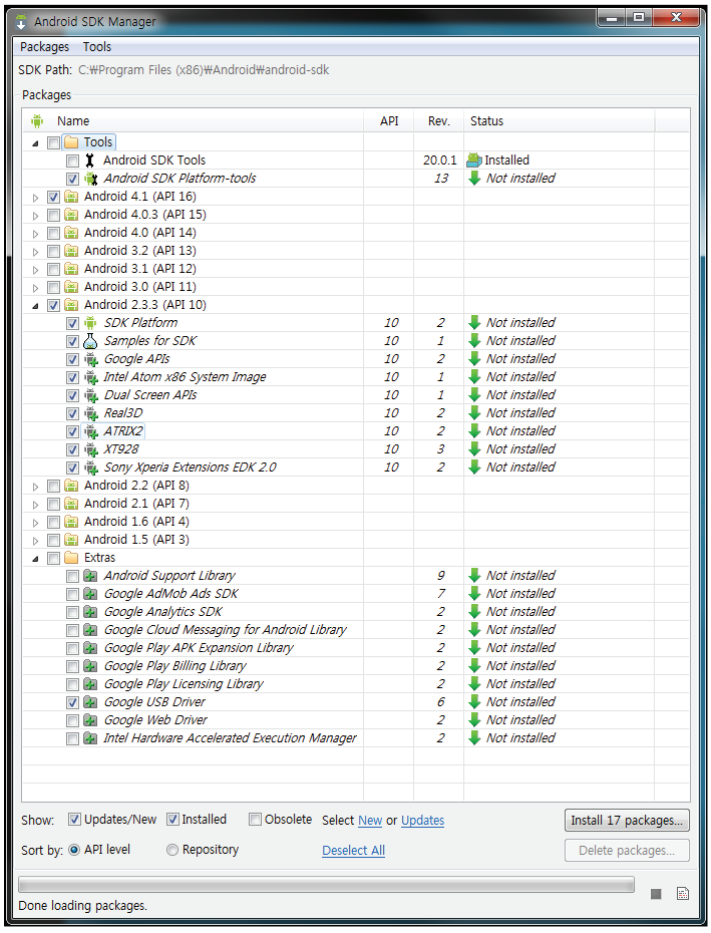
08

모든 설치가 완료 후, Finish 클릭하여 SDK Manager를 실행



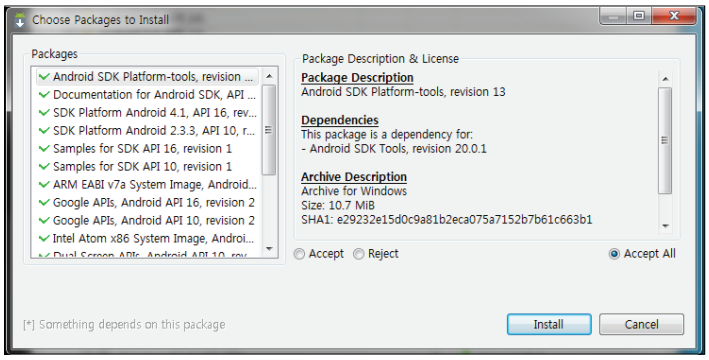
09

SDK 매니저가 실행되면, 기본권장사항이 체크되어 있음 Hovis Genie에서의 앱개발은 Android 2.3.3 (API 10)을 이용하므로,추가적으로 Android 2.3.3을 체크한 후, Install 버튼 클릭



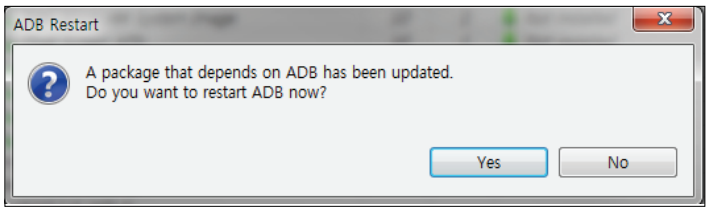
10

라이센스에 동의한 후, Install 클릭하여 설치 시작 (이 설치과정은 1~3시간 정도의 시간이 소요)



11

모두 설치가 완료되면 Yes를 클릭하여 ADB를 다시시작하고, SDK 매니저를 닫습니다.



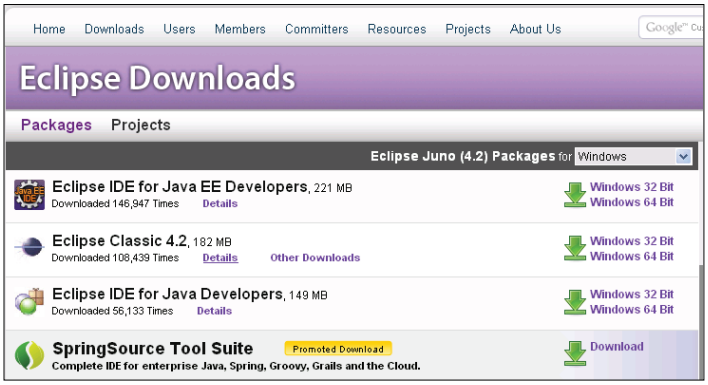
ADB (Android Debug Bridge)는 안드로이드 앱을 개발할 때 에뮬레이터 또는 실제 안드로이드 장치의 조작 및 디버깅 관련 인터페이스를 제공합니다.

2.3 이클립스 설치 및 환경 설정

안드로이드 개발을 위해서 다양한 툴을 사용할 수 있습니다. 그 중 가장 널리 사용되고 있는 툴은 이클립스입니다. 이클립스는 오픈소스기반의 통합개발환경으로서 소스 코드의 작성, 디버깅, 및 실행을 편리하게 할 수 있도록 다양한 기능을 제공하는 범용의 개발도구입니다. 사실 거의 대부분의 안드로이드 개발자가 이클립스를 사용한다고 해도 과언은 아닐 정도로 널리 쓰이고 있습니다. 그러므로 우리도 이클립스를 통해 안드로이드 개발환경 구축합니다.
이클립스의 설치는 다른 소프트웨어의 설치와는 조금 다릅니다. 모든 자바기반의 오픈소스 프로젝트 소프트웨어의 설치와 유사하게, 다운 받은 압축파일의 압축을 해제해줌으로써 설치를 완료할 수 있습니다.
이클립스의 설치를 마치면, 이클립스를 안드로이드 개발환경으로 구축하기 위해서 ADT Plugin을 설치합니다. ADT는 Android Development Tools의 약자로 이클립스상에서 안드로이드 앱의 제작, UI(User Interface)구성 및 앱의 디버깅등을 쉽고 빠르게 할 수 있도록 제공하는 도구를 말합니다. 즉, 이클립스상에서의 ADT Plugin의 설치는 범용개발도구로 구성된 이클립스를 안드로이드 앱 개발에 최적화 된 개발도구로 구축하는 작업입니다.
이제부터 안드로이드 로봇 앱개발을 위한 환경을 단계별로 구축합니다.

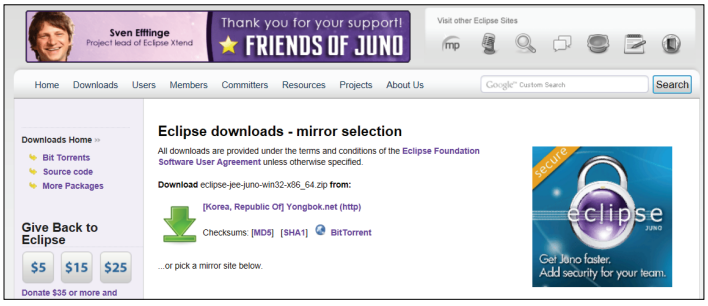
01

이클립스 사이트로 이동 후 OS에 맞는 Eclipse IDE for Java EE Developers 선택
(http://www.eclipse.org/downloads/)
*** 현재 이클립스 3.6.2(Helios) 또는 그 이후 버전을 다운로드 할 것**



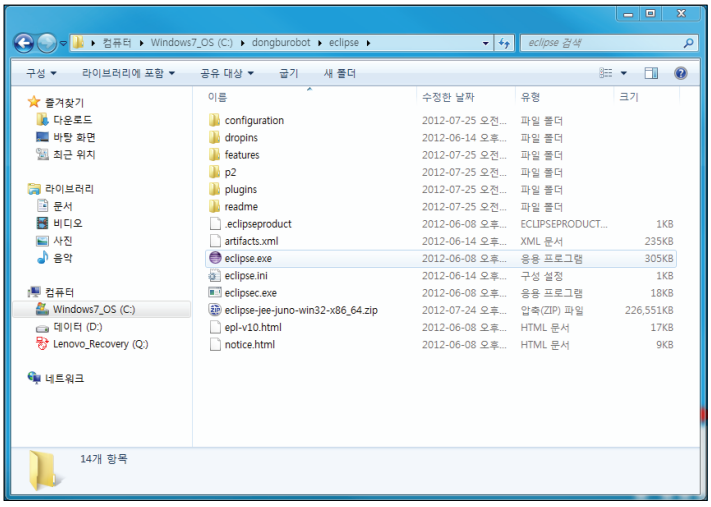
02

다음의 미러 사이트를 클릭하여 이클립스를 다운로드



03

다운로드 받은 이클립스 압축파일 풀기 (여기서는 C:₩dongburobot₩eclipse₩)
주의 : 압축을 풀어주는 경로에 한글 포함하지 말 것.

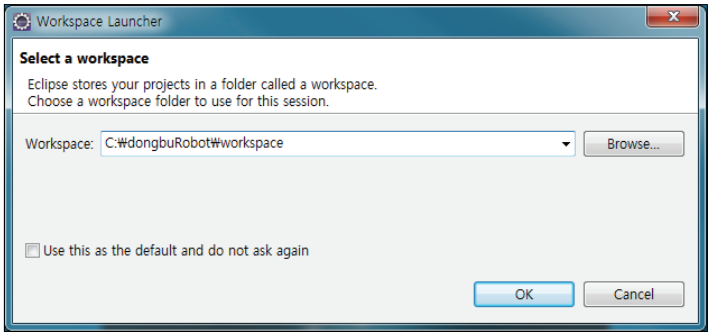


04

바탕화면에 바로가기 아이콘을 생성하고, 이클립스를 실행합니다.

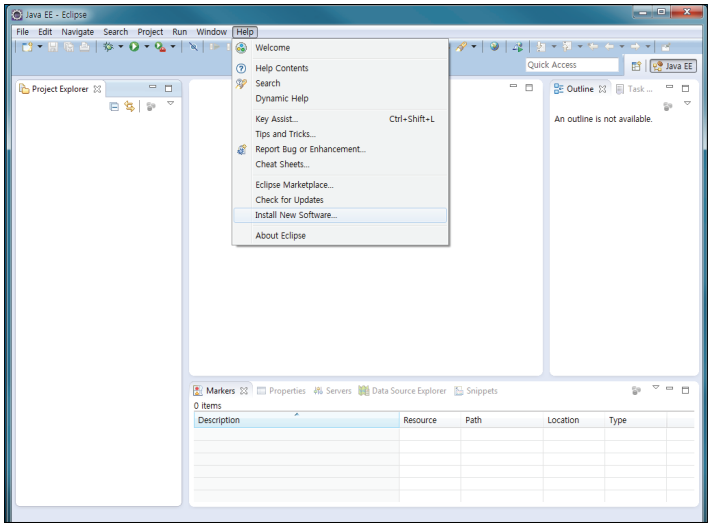
05

이클립스가 실행되면, Workspace를 지정하고 OK를 클릭합니다.(Workspace는 소스코드를 편집하고 저장하는 작업공간을 뜻합니다. 여기서는 C:₩dongburobot₩workspace₩)



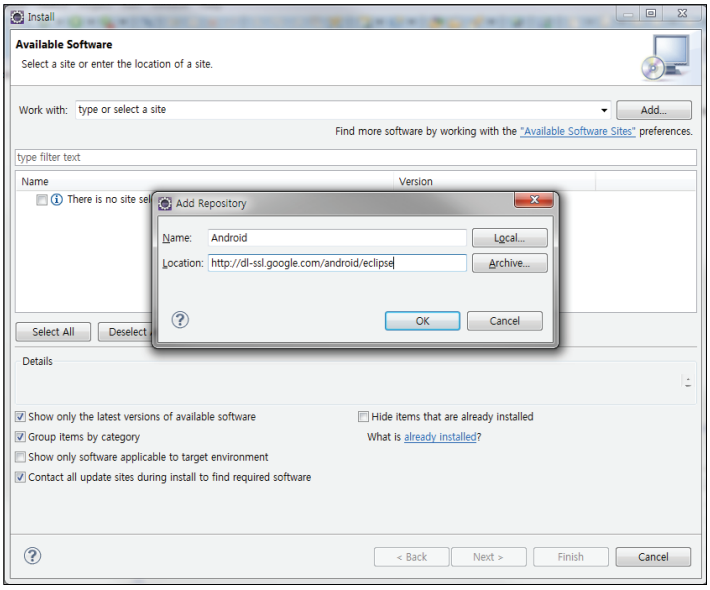
06

이클립스에서 Help 〉 Install New Software… 를 선택합니다.



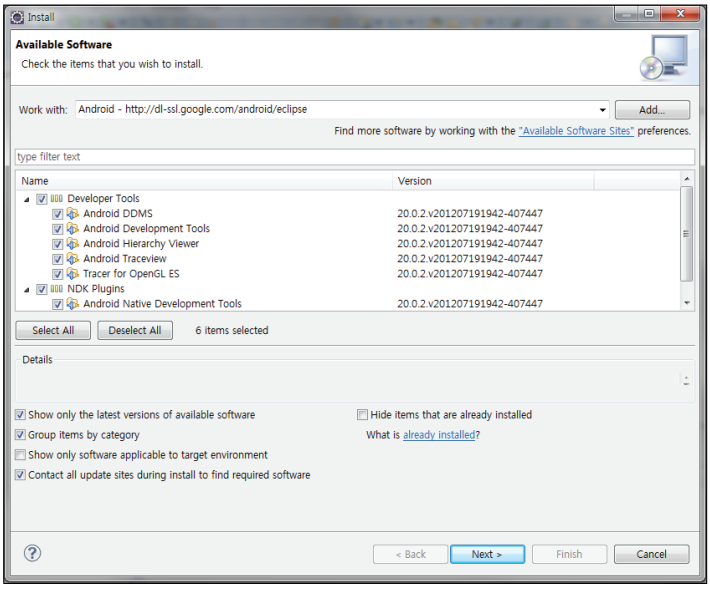
07

Add를 클릭하면 나타나는 Add Repository 윈도우에 다음과 같이 Name과 Location을 입력합니다.
Name : Android
Location : http://dl-ssl.google.com/android/eclipse/



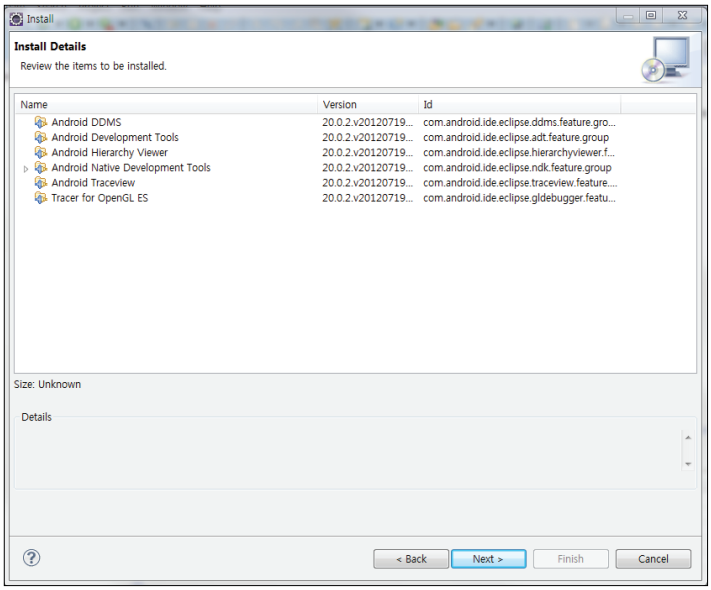
08

7에서 추가한 Repository를 선택하면 잠시후 다음과 같은 화면이 나타납니다. Select All을 선택하고, Next를 클릭합니다.



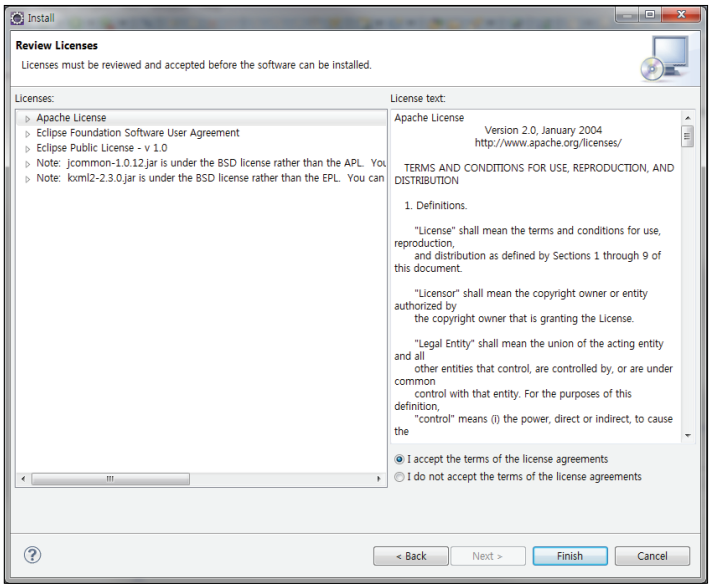
09

설치할 이클립스 Plugin이 목록을 확인하고 Next를 클릭합니다.



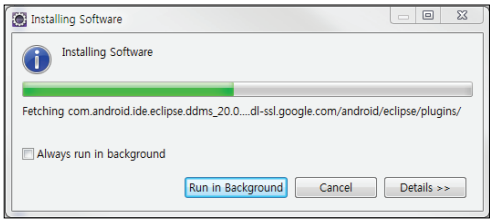
10

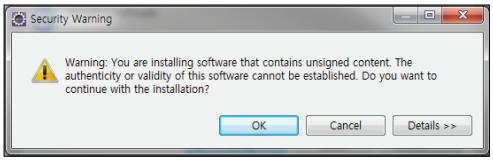
라이센스에 동의한 후 Finish를 클릭하여 설치를 시작합니다.



11

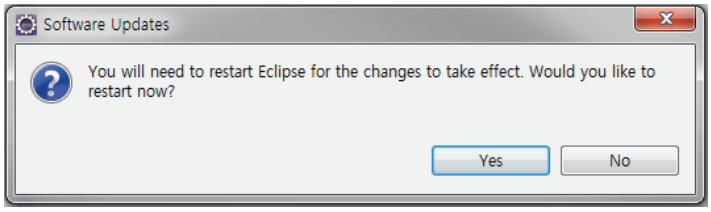
설치 도중 다음과 같은 보안 경고가 나타날 수 있습니다. OK를 클릭하여 계속 진행합니다.





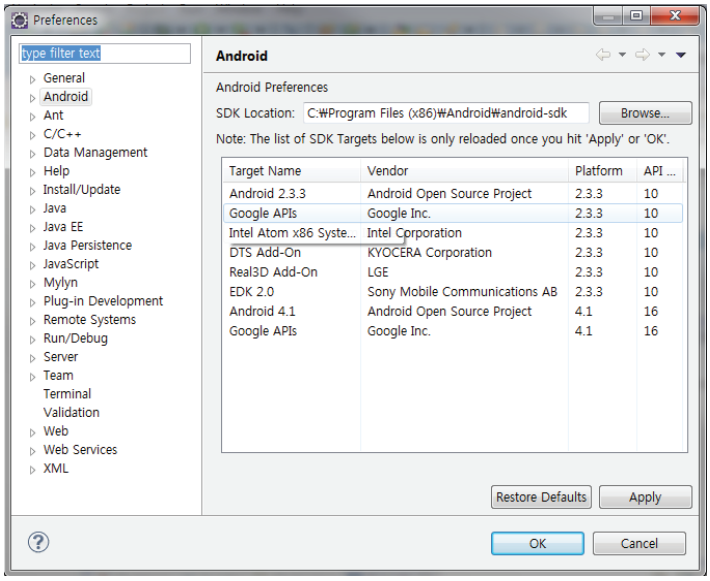
12

플러그인의 설치가 완료되면, 이클립스를 재시작합니다.



13

이클립스에서 Window 〉 Preferences 를 실행하여, Android 탭의 SDK Location을 지정합니다. SDK Location에는 2.2절에서 설치한 안드로이드 SDK의 경로를 지정합니다. Apply 버튼을 눌러 적용 후, OK를 눌러 설정을 마무리합니다.



03 안드로이드 로봇(지니) 프로그래밍

Chapter 2에서는 안드로이드 프로그래밍을 위한 개발환경을 구축하였습니다. 이번 장에서는 본격적으로 안드로이드 기반의 프로그래밍을 시작합니다.

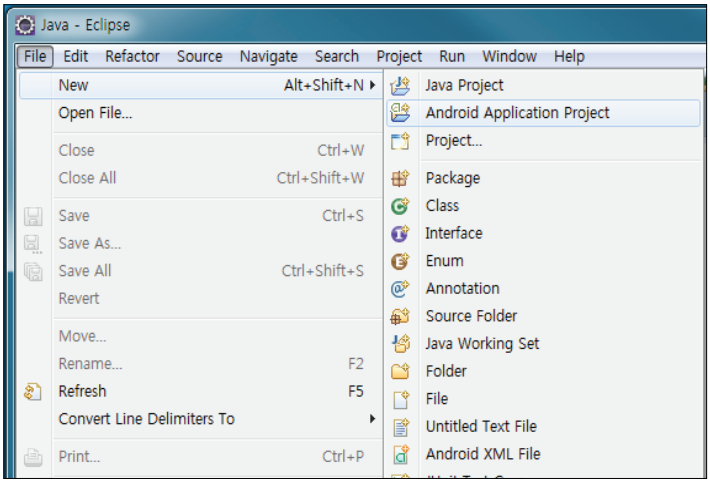
이번 장의 목표는 안드로이드 플랫폼에서 동작하는 첫 번째 안드로이드 App의 제작입니다. 목표를 달성하기 위해서 안드로이드에 대한 이해가 선행되어야 합니다. 우리는 첫 번째 App을 만드는 과정을 통해서 안드로이드의 기본적인 구조와 특징을 살펴볼 것입니다. 그러나, 안드로이드는 방대합니다. 이 장에서 모든 내용을 다루기는 한계가 있으므로 안드로이드 기본서 또는 안드로이드 개발자 사이트(http://developer.android.com/index.html) 를 참조하시기 바랍니다.

비록, 안드로이드의 모든 내용을 다루지는 않지만, 여러분은 이 책에 담긴 예제를 통해 안드로이드의 전반적인 이해에 도달하리라 생각합니다. 그러면 우리의 목표인 안드로이드 기반 로봇 프로그래밍을 위하여 첫 번째 App 인 HelloGenie를 시작합니다.

3.1 Hello Genie 프로젝트 생성

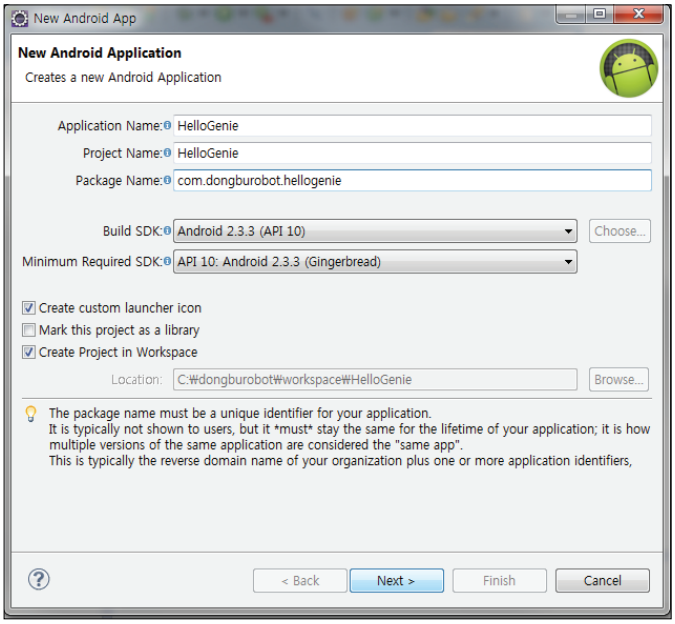
01

이클립스를 실행하고, File〉New〉Android Application Project를 선택합니다.



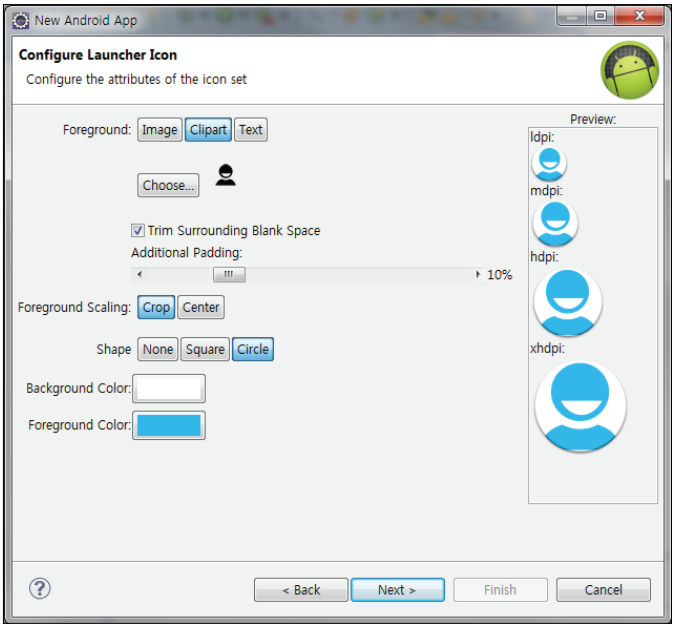
02

App, 프로젝트 및 패키지 이름을 지정합니다. 이 때, Build SDK 항목은 반드시 Android 2.3.3 (API 10)으로 지정합니다. 그 이유는 Hovis Genie에 부착되는 MID의 안드로이드 버전이 2.3.3이기 때문입니다. 모든 이름을 지정하면 Next를 클릭하여 다음으로 넘어갑니다.



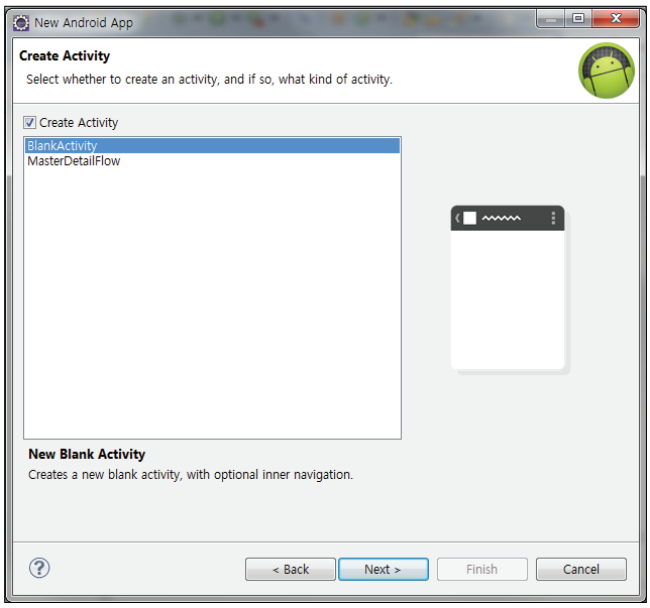
03

런처 아이콘을 설정합니다. 런처 아이콘이란 스마트폰에서 App을 나타내기 위한 네모난 아이콘입니다. Next를 눌러 다음으로 이동합니다.



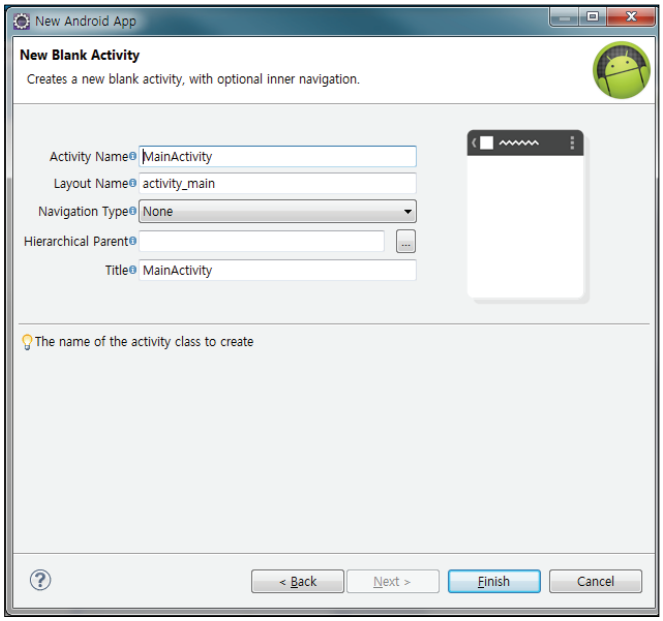
04

Activity를 생성합니다. Activity란 안드로이드 폰에서 하나의 화면을 뜻합니다. 예를 들어, 안드로이드 폰으로 문자 메시지를 보다가 전화가 와서 화면이 전환된다면, 하나의 Activity가 다른 Activity로 전환된 것입니다. 이런 Activity의 자세한 내용은 다시 다루도록 하겠습니다. BlankActivity를 선택하고 Next를 눌러 다음으로 이동합니다. (MasterDetailFlow는 Tablet에서 사용할 수 있도록 새롭게 추가된 기능입니다. 본서에서는 다루지 않습니다.)



05

생성된 BlankActivity의 이름을 지정합니다. 여기서는 기본적으로 지정된 이름을 사용합니다. Finish를 눌러 프로젝트 생성을 완료합니다.



3.2 Hello Genie 프로젝트의 실행

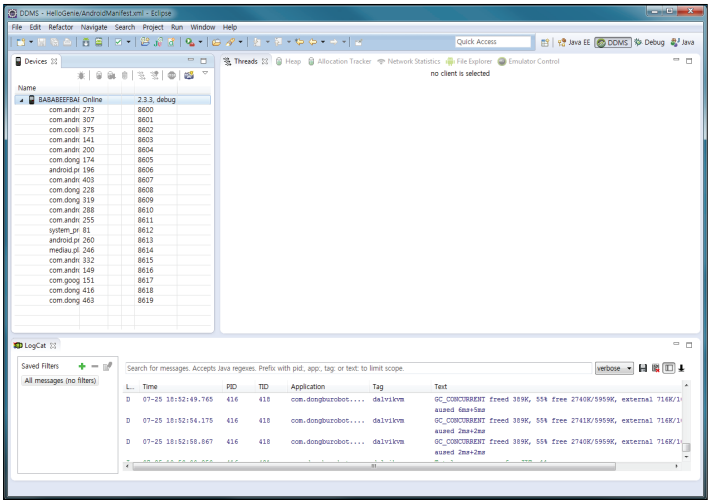
HelloGenie 프로젝트를 실행해보겠습니다. 이클립스에서 생성한 HelloGenie 프로젝트를 실행하는 방법에는 두가지가 있습니다. 실제 안드로이드 장치와 가상의 에뮬레이터 장치에서 프로젝트를 실행할 수 있습니다. 여기에서는 Hovis Genie를 제어하는 안드로이드 장치인 MID를 이용합니다.

01

USB 잭을 이용하여 MID와 PC 연결

02

이클립스 DDMS를 이용하여 장치의 연결을 확인하고 장치 선택



03

이클립스 툴바에서 ▶ 버튼을 눌러 프로젝트를 실행하고 MID에서 실행 된 화면을 확인합니다.

[TIP] 만약 MID가 검색되지 않는다면?

[설정 〉 어플리케이션 〉 개발 〉 USB 디버깅] USB 디버깅을 해제하고 다시 체크할 것

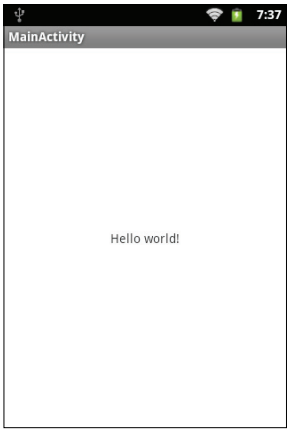
[TIP] 만약 실행이 되지 않는다면?

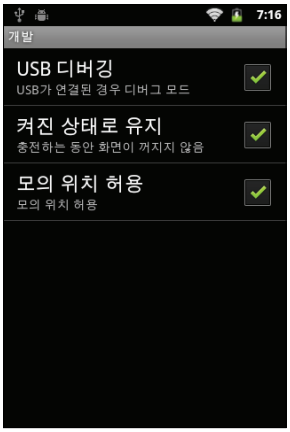
버튼을 클릭시 xml 소스코드가 선택되어 있다면 실행이 되지 않습니다. 이 경우 *.xml.out 이라는 파일이 생성되게 되는데, 이 파일을 지우고 /src의 아무 java 파일을 열고 실행 버튼을 눌러 실행하세요.

[TIP] 다음과 같은 에러 메시지가 나오다면?

Parser exception for /HelloGenie/ AndroidManifest.xml: 문서에서 루트 요소 다음에 오는 마크업은 올바른 형식이어야 합니다.

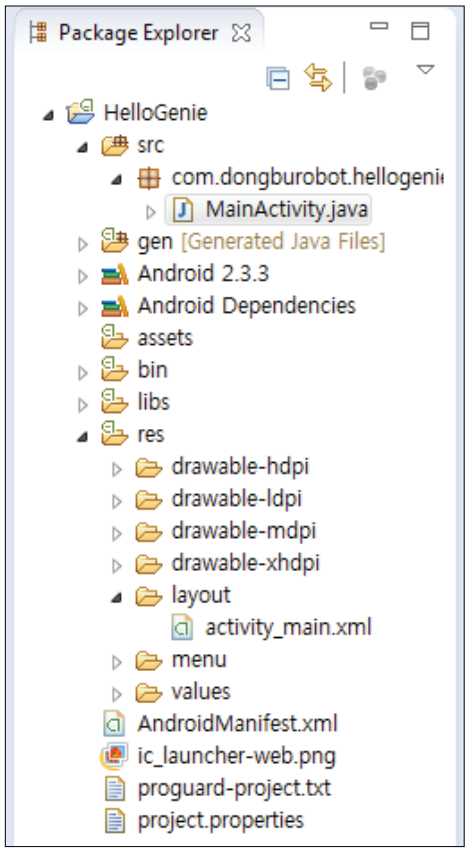
이클립스에 추가한 ADT Plugin의 버그로 보입니다. 다음 절의 소스코드 설명을 참조하여 소스를 수정하세요.





3.3 Hello Genie 프로젝트의 기본구성

첫 번째 App인 HelloGenie 프로젝트를 생성하여 실제 안드로이드 장치인 MID에서 실행해보았습니다. 이어서 생성된 HelloGenie의 프로젝트 파일들을 확인합니다. HelloGenie 프로젝트의 기본 구성은 이클립스의 왼쪽에 있는 Package Explorer창에서 확인할 수 있습니다. 각 구성은 다음과 같습니다.



/src
App의 소스를 포함하고 있는 디렉토리입니다. 현재는 프로젝트를 생성할 때 지정한MainActivity가 기본적으로 포함되어 있습니다.

/gen
이 디렉토리에 있는 파일들은 자동생성됩니다. 리소스 및 환경에 대한 정보를 가지는 파일들이 있습니다.

/bin
프로젝트를 빌드하면 생성되는 파일들이 있는 디렉토리입니다.

/libs
프로젝트에 포함되는 외부 라이브러리가 있는 디렉토리입니다. 향후 로봇서비스를 제공하는 라이브러리를 포함할 때 이 디렉토리를 이용할 것입니다.

/res
프로젝트의 리소스를 포함하고 있습니다.

/res/drawable
프로젝트에 사용되는 이미지 파일들을 포함합니다.

/res/layout
이 디렉토리는 화면의 UI(User Interface)를 묘사하는 파일을 포함합니다. 하나의 Activity는 하나의 화면을 나타냅니다. 하나의 Activity는 이 디렉토리에 포함된 하나의 xml파일을 사용합니다.

/res/menu
이 디렉토리는 안드로이드 폰에서 메뉴 버튼을 눌렀을 때 나오는 화면에 대해 묘사하는 xml파일을 포함합니다.

/res/values
기타 리소스(문자열 또는 스타일)을 지정하는 xml파일들을 포함합니다.

AndroidManifest.xml
이 파일은 프로젝트의 컴포넌트 구성과 기본적인 속성에 대해 정의합니다.
하나의 프로젝트는 하나의 AndroidMnaifest.xml 파일이 있습니다.

HelloGenie 프로젝트의 기본 구성은 많은 디렉토리를 포함하고 있습니다. 이 중 /src, /res, 그리고 AndroidManifest.xml 파일은 App 구동 이해에 필요한 부분입니다.

(1) AndroidMainfest.xml

우선 AndroidMainfest.xml파일을 살펴보겠습니다. 하나의 App은 하나의 Android Mainfest.xml을 포함합니다. AndroidMainfest.xml 파일은 App의 컴포넌트 구성을 정의하고, App의 권한과 속성을 지정해주는 역할을 합니다. 즉, App의 핵심적인 설정을 담고 있는 파일입니다. 다음은 AndroidMainfest.xml 파일입니다.

```
1: 〈manifest xmlns:android="http://schemas.android.com/apk/res/android"
2:     package="com.dongburobot.hellogenie"
3:     android:versionCode="1"
4:     android:versionName="1.0" 〉;
5:
6:     〈uses-sdk
7:         android:minSdkVersion="10"
8:         android:targetSdkVersion="10" /〉;
9:
10:    〈application
11:        android:icon="@drawable/ic_launcher"
12:        android:label="@string/app_name"
13:        android:theme="@style/AppTheme" 〉;
14:        〈activity
15:            android:name=".MainActivity"
16:            android:label="@string/title_activity_main" 〉;
17:            〈intent-filter〉
18:                〈action android:name="android.intent.action.MAIN" /〉
19:                〈category android:name="android.intent.category.LAUNCHER" /〉
20:            〈/intent-filter〉
21:        〈/activity〉
22:    〈/application〉
23:
24: 〈/manifest〉
```

Listing1. AndroidManifest.xml

AndroidMainfest.xml의 최상위 구조 〈manifest〉 태그입니다. 이 태그는 App의 기본정보를 포함합니다. 2번째 줄에 Package 속성은 프로젝트를 생성할 때 지정한 패키지 이름을 가집니다. 3~4번째 줄에 android:versionCode, android:versionName는 해당 App의 버전정보를 지정합니다. 이 속성들은 향후 App을 업그레이드하여 재배포시 수정하는 부분입니다.

〈manifest〉의 하위 태그로는 〈uses-sdk〉와 〈application〉이 있습니다. 〈uses-sdk〉 태그는 App의 동작환경을 지정하는 역할을 합니다. 7번째 줄에 android:minSdkVersion 속성은 이 App이 동작할 수 있는 최소한의 안드로이드 버전을 지정하는 부분입니다. 8번째줄의 android:targetSdkVersion 속성은 App이 사용하는 API Level을 뜻합니다. API Level 10은 안드로이드 2.3.3 진저브레드 버전입니다. 현재 로봇에 탑재될 안드로이드 장비인 MID의 안드로이드의 버전이 2.3.3이므로 이 속성을 10으로 지정합니다.

AndroidManifest.xml의 최상위 구조 〈manifest〉 태그입니다. 이 태그는 App의 기본정보를 포함합니다. 2번째 줄에 Package 속성은 프로젝트를 생성할 때 지정한 패키지 이름을 가집니다. 3~4번째 줄에 android:versionCode, android:versionName는 해당 App의 버전정보를 지정합니다. 이 속성들은 향후 App을 업그레이드하여 재배포시 수정하는 부분입니다.

〈application〉 태그는 App의 컴포넌트 구성을 지정합니다. 안드로이드에는 크게 네 가지의 App 핵심 컴포넌트가 있습니다. 이는 Activity, Service, Broadcast receiver, 그리고 Content Provider입니다 (컴포넌트 상세 설명 웹사이트 참조). 현재 HelloGenie 프로젝트는 단순히 하나의 Activity로 구성되어 있기 때문에 하나의 〈activity〉 태그를 포함하고 있습니다.

11번째 줄에 기술 된 〈application〉 태그의 android:icon 속성은 안드로이드 장치에서 보여 줄 아이콘을 설정합니다. 속성값 "@drawable/ic_launcher"는 안드로이드 장치의 해상도에 따라 /res/drawable-hdpi/, … , /res/drawable-mdpi 디렉토리에 ic_laucher를 이름으로 가지는 그림파일을 가리킵니다 (@은 이미 있는 리소스를 참조할 때 사용하는 키워드). 12번째 줄의 android:label 속성은 App의 이름을 지정하며, "@string/app_name"은 /res/values/strings.xml에 지정된 app_name의 문자열을 참조합니다. 13번째 줄은 App의 style을 지정하며, "@style/AppTheme"은 /res/values/styles.xml에 지정된 AppTheme을 참조합니다.

14~21번째 줄에서는 프로젝트에 포함된 Activity를 설정합니다. 15번째 줄에 android:name 속성은 이 Activity가 구현된 소스코드의 위치를 지정하고 있습니다. 이 Activity가 구현된 소스코드의 위치는 com.dongburobot.hellogenie.MainActivity입니다. 그러나 〈manifest〉 태그에서 지정된 package이름은 생략이 가능하여 속성값을 .MainActivity로 지정해도 무방합니다. 16번째 줄은 이 Activity가 외부로 보여지는 이름을 지정합니다.

17~20번째 줄에는 〈intent-filter〉가 명시되어 있습니다. 인텐트 필터는 이 Activity에 전달되는 인텐트를 걸러내는 역할을 합니다. 여기서 인텐트란 어떤 핵심 컴포넌트를 원하는 대로 동작시키기 위해 필요한 정보를 가지는 객체이며, 다수 혹은 특정 핵심 컴포넌트를 활성화 시키는 역할을 합니다.
구체적으로 런처(홈 스크린 화면)에서 HelloGenie App을 실행할 때 android.inent.action.MAIN의 Action과 android.intent.category.LAUNCHER의 Category 속성을 가지는 인텐트를 HelloGenie에 전달합니다. 전달된 인텐트는 인텐트 필터에 의해 두 가지 필터 속성을 가지고 있는 .MainActivity를 실행합니다.

AndroidManifest.xml 파일은 App의 구성을 묘사하는 중요한 파일이며, 하나의 App은 하나의 AndroidManifest.xml을 반드시 가지고 있습니다. AndroidManifest.xml을 통해서 본 HelloGenie App은 〈uses-sdk〉 태그를 통해 안드로이드 2.3.3에서 실행되며, 〈application〉 태그를 통해 하나의 Activity를 포함하고 있다는 것을 알 수 있습니다. 현재 유일한 〈activity〉인 MainActivity는 인텐트 필터에 의해, 런처(홈 스크린 화면)에서 유저가 아이콘을 클릭 시 발생하는 인텐트를 수신하여 실행됩니다. 그러면 이제 MainActivity를 살펴보겠습니다.

(2) MainActivity.java와 activity_main.xml

하나의 Activity란 하나의 화면을 뜻합니다. 유저는 이러한 화면(Activity)을 통해서 스마트폰과 조작합니다. 쉽게 생각하면, 하나의 App은 다수의 화면으로 이루어져있으므로, 여러 개의 Activity의 느슨한 결합이라고 생각하셔도 무방합니다.

하나의 App에서 유저가 그 App을 실행할 때 가장 처음으로 나타나는 하나의 Activity가 있습니다. 이는 AndroidManifest.xml에 정의되어 있는데, 우리의 HelloGenie 프로젝트에선, 이것이 MainActivity입니다.

MainActivity에도 유저와의 상호작용을 위해서는 유저 인터페이스 정의부와 처리부가 존재합니다. 여기서 유저 인터페이스 정의부가 activity_main.xml에 해당하고, 처리부가 MainActivity.java에 해당합니다. 우선, MainActivity.java 파일을 살펴보겠습니다.

```
1: package com.dongburobot.hellogenie;
2:
3: import android.os.Bundle;
4: import android.app.Activity;
5: import android.view.Menu;
6:
```

```
 7: public class MainActivity extends Activity {
 8:
 9:     @Override
10:     public void onCreate(Bundle savedInstanceState) {
11:         super.onCreate(savedInstanceState);
12:         setContentView(R.layout.activity_main);
13:     }
14:
15:     @Override
16:     public boolean onCreateOptionsMenu(Menu menu) {
17:         getMenuInflater().inflate(R.menu.activity_main, menu);
18:         return true;
19:     }
20: }
```

Listing2. MainActivity.java

모든 안드로이드 Activity는 android.app.Acitivity 클래스를 상속받아서 생성됩니다. 우리의 MainActivity 클래스도 Activitiy 클래스를 상속받아 생성되었습니다. Activity 클래스를 상속받으면 Activity의 라이프사이클에 따라 여러가지 메서드를 오버라이드 할 수 있습니다. 참고로, Activity 라이프 사이클은 매우 중요한 내용으로 반드시 숙지하셔야 되는 부분입니다. 자세한 내용은 http://developer.android.com/guide/components/activities.html 를 참조하고, 상세 내역은 차후에 다시 설명하도록 하겠습니다.

이 중 10번째 줄의 onCreate() 메서드는 MainActivity가 처음 생성될 때 수행되는 메서드입니다. onCreate()의 12번 째 줄에는 다음과 같은 라인이 있습니다.

```
setContentView(R.layout.activity_main);
```

이 라인의 setContentView()는 activity_main.xml 에 정의되어 있는 유저 인터페이스를 현재의 MainActivity에 설정하는 역할을 합니다. setContentView()에 인자인 R.Layout.activity_main은 res/layout/activity_main.xml을 나타냅니다. 다음은 activitiy_main.xml입니다.

```
1: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2:     xmlns:tools="http://schemas.android.com/tools"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent" >
5:
6:     <TextView
7:         android:layout_width="wrap_content"
8:         android:layout_height="wrap_content"
9:         android:layout_centerHorizontal="true"
10:        android:layout_centerVertical="true"
11:        android:text="@string/hello_world" />
12: </RelativeLayout>
```

Listing3. activity_main.xml

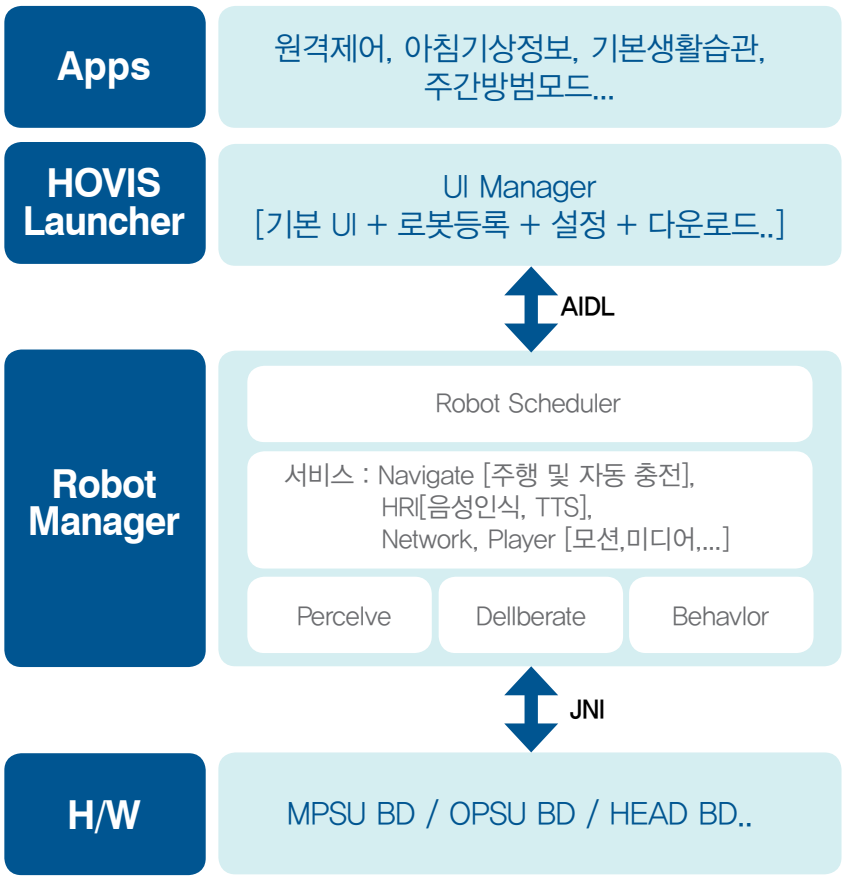
activity_main.xml은 RelativeLayout과 하위 앨리먼트인 TextView로 구성되어 있습니다. Relative Layout은 하나 이상의 뷰를 뷰 간의 상대적인 위치를 통해 배치하는 레이아웃입니다. TextView는 화면에 텍스트를 출력하는 뷰입니다. Activitiy_main.xml에서 정의 된 유저 인터페이스가 실제화면에서 다음 그림과 같이 표현됩니다.

다만, 여기에서는 상세한 설명을 생략합니다. 자세한 내용은 Activity의 라이프 사이클과 같이 차후에 예제에서 다루도록 하겠습니다. XML 레이아웃 파일에 대한 자세한 내용은 http://developer.android.com/guide/topics/ui/declaring-layout.html 를 참조할 것을 권합니다.

04 HOVIS Genie 로봇 프로그래밍 시작하기

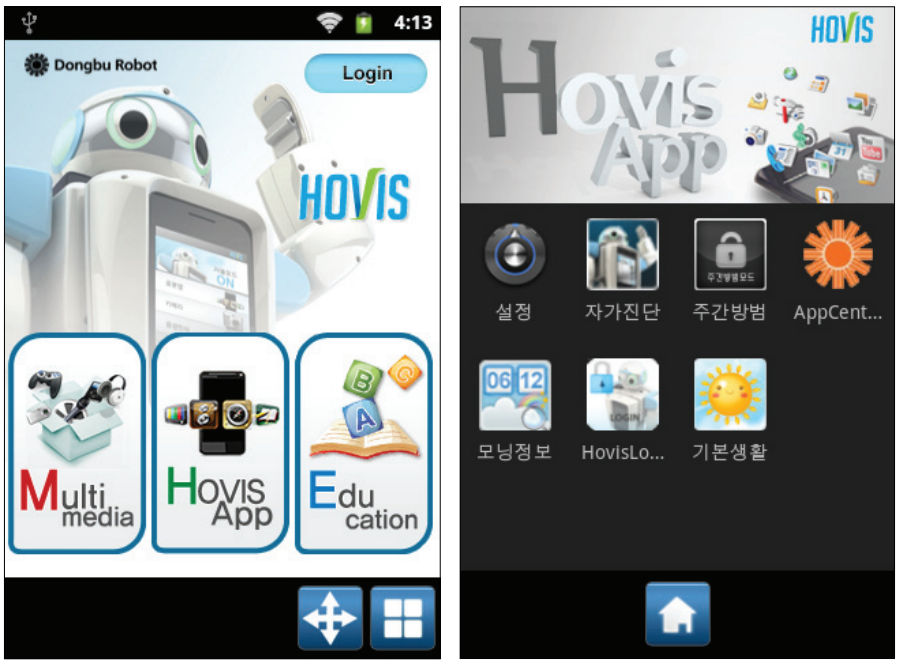
Chapter 2에서는 안드로이드 프로그래밍 환경을 구성하였고, Chapter 3에서는 첫번째 안드로이드 App을 실행해보면서 전반적인 안드로이드 프로젝트에 대한 구성을 살펴보았습니다. 이번 장에서는 본격적으로 Hovis Genie 로봇 프로그래밍을 위해서 첫 발을 들여놓도록 하겠습니다.

우선 Hovis Genie 로봇 프로그래밍을 시작하기 앞서, Hovis Genie에서 동작하는 S/W 구조의 이해가 필요합니다. 이를 통해, 로봇이 동작 원리를 이해하고 서비스를 이용함으로써 로봇 App의 제작이 가능하기 때문입니다. [그림 4-1]은 안드로이드 시스템 상에서의 Hovis Genie의 전체적인 S/W의 구조를 나타냅니다.



[그림 4-1] Hovis Genie S/W

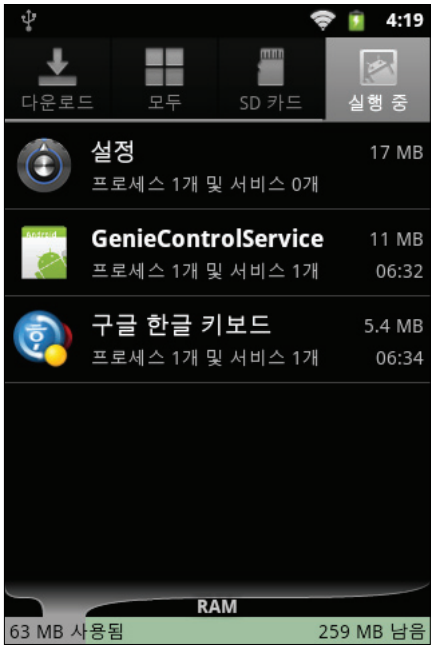
안드로이드 시스템상에서의 Hovis Genie의 S/W 구성은 응용프로그램인 Apps, HOVIS Launcher, 그리고 Robot Manager로 구분됩니다. 이 중 상위 응용프로그램 레벨인 Apps와 Hovis Launcher는 각각 로봇 App과 홈 스크린 화면을 가리키며 [그림 4-2]와 같이 눈으로 쉽게 확인할 수 있습니다.



[그림 4-2] Hovis Launcher와 로봇 App

반면에, Robot Manager는 시스템 내부에서 실행이 되므로 외부로 드러나지 않지만, 중요한 역할을 담당하는 Robot Scheduler와 서비스가 있습니다. 첫째로, Robot Scheduler는 로봇의 이벤트 처리를 담당하며, 로봇의 상태에 따라 자율모드, 충전모드 등의 주요 모드에 따라 로봇의 행동을 결정하고 수행하는 부분입니다. 둘째로, 서비스는 로봇의 센서 및 모터 등의 하드웨어 전반을 추상화하여 이를 손쉽게 사용할 수 있도록 로봇 서비스를 제공하는 부분입니다.

특히, 서비스는 우리가 제작할 로봇 App이 로봇을 쉽게 제어할 수 있도록 API(Application Programing Interface)를 제공합니다. Robot Manager 상위에 있는 Hovis Launcher(홈 스크린 화면)와 로봇 App은 Robot Manager의 서비스를 사용하여 로봇과 관련된 작업을 수행합니다.



[그림 4-3] Robot Manager 레벨의 서비스가 GenieControlService

이 서비스는 안드로이드에서 [그림 4-3]과 같이 GenieControlService라는 이름으로 실행됩니다. GenieControlService가 [그림 4-1]의 Robot Manager레벨을 구현하고 있으며, Robot Scheduler와 서비스를 포함하고 있습니다. 중요 모듈을 포함하는 GenieControl-Service는 로봇의 제어를 위한 필수이므로 로봇이 운용 시에는 항상 백그라운드에서 동작하는 특징이 있습니다. 그리고 예기치 못한 시스템 오류에 강인하여 문제가 발생해도 자동으로 복구되는 특징도 있습니다.

요약하자면, 로봇 App에서 로봇 H/W 자원을 사용하기 위해서는 반드시 GenieControl-Service가 필요합니다. 이번 장에서는 우리가 생성하는 App에서 로봇 서비스인 Genie-ControlService로 접근하는 방법을 주로 알아보겠습니다. 이번 장의 예제는 Chapter5의 서비스를 이용한 로봇 프로그래밍 예제로 이어집니다. 다음 장에 선행되는 예제이므로 정확한 숙지가 필요합니다. 그럼 시작합니다.

4.1 GenieApiDemo 프로젝트 생성

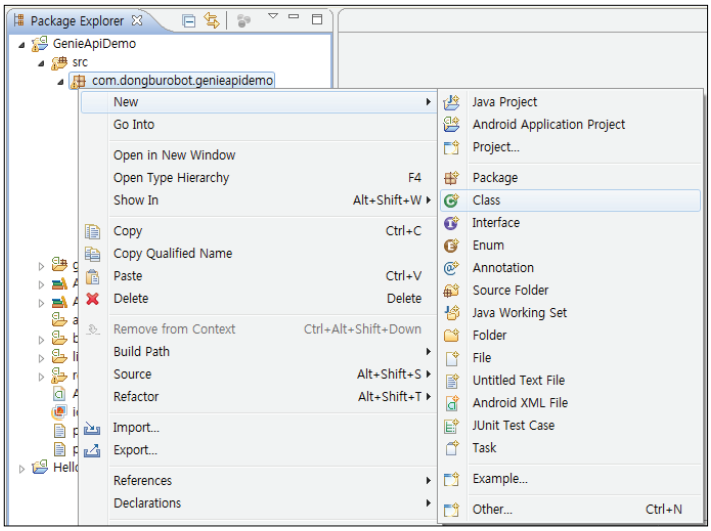
이번 장의 예제를 위해 새로운 프로젝트를 생성하겠습니다. 이클립스에서 File 〉 New 〉 Android Application Project 창을 실행하여 다음과 같이 입력하여 프로젝트를 생성합니다. (상세내용은 Chapter 3 참조)

- Application Name : GenieApiDemo
- Project Name : GenieApiDemo
- Package Name : com.dongburobot.genieapidemo
- Build SDK : Android 2.3.3 (API 10)
- Minimum Required SDK : API 10: Android 2.3.3 (Gingerbread)

GenieApiDemo 프로젝트가 생성되면, GenieApiDemoApplication.java와 BaseActivity.java를 추가합니다. 그리고 GenieControlService를 사용하기 위해 HovisService.jar 파일을 libs 폴더에 추가합니다.

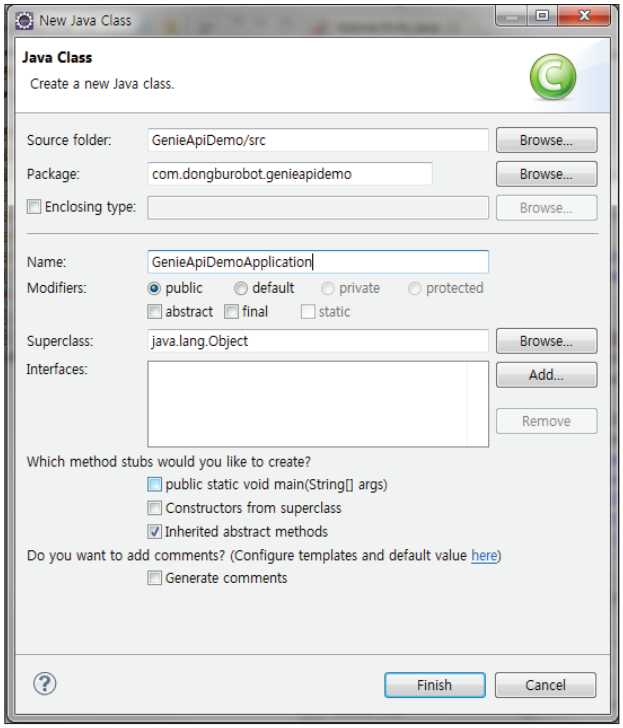
01

src/com.dongburobot.genieapi에서 우클릭 후 New 〉 Class 클릭



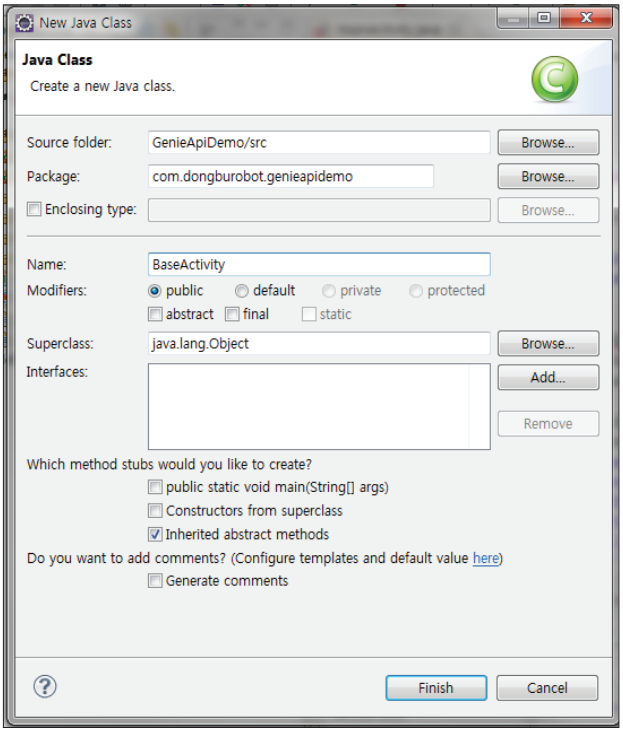
02

GenieApiDemoApplication.java를 추가합니다.



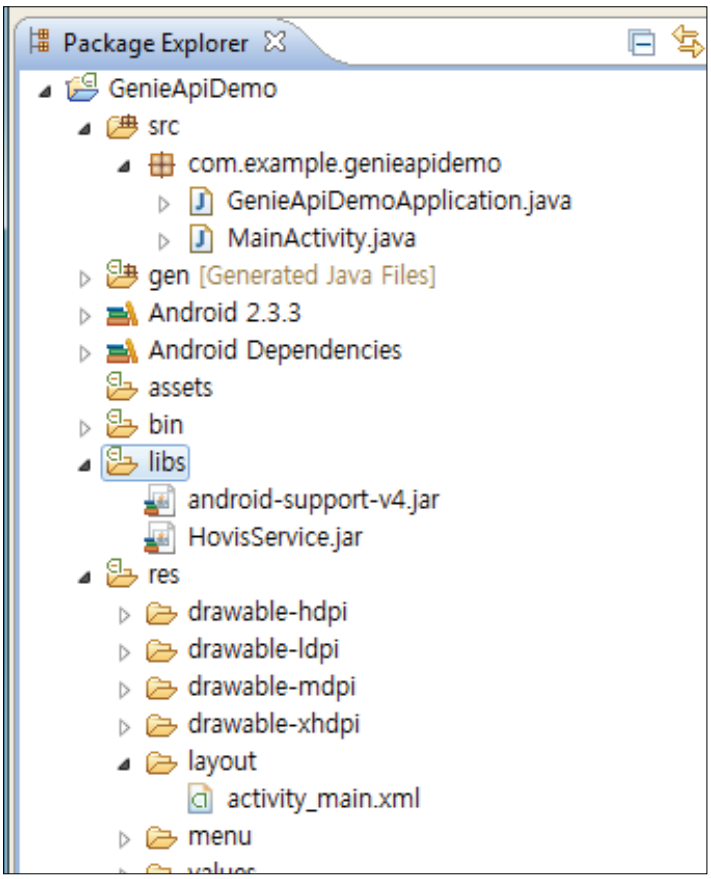
03

똑같은 방법으로 BaseActivity.java를 추가합니다.



04

HovisService.jar 파일을 드래그하여 libs 폴더에 추가합니다. HovisService.jar 파일은 동부로봇 홈페이지에서 다운로드합니다.
(http://www.dongburobot.com/jsp/cms/view.jsp?code=100122)



여기서 libs폴더의 HovisService.jar은 GenieControlService에 포함되어 있는 로봇 관련 API의 선언부를 하나의 파일로 묶어 배포된 파일입니다. 이 파일이 프로젝트에 제대로 포함되지 않으면 프로젝트가 제대로 빌드되지 않으니 주의하세요. 그럼 본격적으로 GenieControlService와 바인딩을 통해 로봇 App 개발을 시작합니다.

※주의 – 이클립스에 설치 된 ADT(Android Development Tools) 버전에 따라, HovisService.jar 파일을 추가하는 방법이 다를 수 있습니다. 최신버전의 이클립스와 ADT Plugin을 사용할 것을 권장합니다.

4.2 GenieApiDemoApplication 클래스

GenieApiDemoApplication 클래스는 Application 클래스를 상속받아 작성합니다. Application 클래스는 android.app.Application에 정의되어 있는 클래스로 안드로이드 App을 추상적으로 표현하는 클래스입니다. Application 클래스는 App이 실행되는 동안 전반적인 상태를 유지가 필요한 경우 사용됩니다. 다시 말하면, App의 전반에 걸쳐 사용할 수 있는 전역변수를 선언할 때 유용합니다. 우리의 GenieApiDemo 프로젝트에서 GenieApiDemoApplication 클래스의 역할은 App이 실행되고 있는 동안 GenieControlService에서 제공하는 서비스(HovisGenieService)에 바인드하여 서비스를 사용할 수 있도록 전역변수를 선언하는 용도로 사용됩니다.

GenieApiDemoApplication 클래스의 소스코드는 다음과 같습니다.

```
 1: package com.dongburobot.genieapidemo;
 2:
 3: import android.app.ActivityManager;
 4: import android.app.ActivityManager.RunningServiceInfo;
 5: import android.app.Application;
 6: import android.content.ComponentName;
 7: import android.content.Intent;
 8: import android.content.ServiceConnection;
 9: import android.os.IBinder;
10: import android.util.Log;
11:
12: import com.dongburobot.HovisGenieServiceInterfaces.IHovisGenieService;
13:
14: public class GenieApiDemoApplication extends Application {
15:
16:     // HovisGenieService 서비스 바인딩
17:     public IHovisGenieService mBinder = null;
18:
19:     private boolean mIsBinded;
20:     private Intent mIntent;
```

```
21:
22:    public ServiceConnection mConnection = new ServiceConnection() {
23:
24:        public void onServiceConnected(ComponentName name, IBinder service) {
25:            mBinder = IHovisGenieService.Stub.asInterface(service);
26:        }
27:
28:        public void onServiceDisconnected(ComponentName name) {
29:            mBinder = null;
30:        }
31:    };
32:
33:    protected boolean isServiceRunning() {
34:        ActivityManager manager = (ActivityManager)getSystemService(ACTIVITY_SERVICE);
35:
36:        for (RunningServiceInfo serviceInfo: manager.getRunningServices(Integer.MAX_VALUE)) {
37:            if (serviceInfo.service.getClassName().equals("com.dongburobot.geniecontrol.HovisGenieService")) {
38:                return true;
39:            }
40:        }
40:          }
41:         return false;
42:
43:     }
44:
45:     protected boolean connectService() {
46:         if ( isServiceRunning() ) {
47:             mIntent = new Intent(IHovisGenieService.class.getName());
48:             mIsBinded = bindService(mIntent, mConnection, BIND_AUTO_CREATE);
49:             return mIsBinded;
50:         }
51:
52:         return false;
53:     }
54:
55:     protected void disconnectService() {
56:         if (mIsBinded) {
```

```
57:             if (mConnection != null) {
58:                 unbindService(mConnection);
59:                 mConnection = null;
60:             }
61:         }
62:
63:         mIntent = null;
64:     }
65:
66:     // Application 시작시와 종료시 서비스 바인딩
67:     @Override
68:     public void onCreate() {
69:         super.onCreate();
70:
71:         if ( !connectService() )
72:             Log.i("service", "not connected");
73:     }
74:
75:     @Override
76:     public void onTerminate() {
77:         super.onTerminate();
78:
79:         disconnectService();
80:     }
81: }
```

[Listing. 4-1] GenieApiDemoApplication.java

지금부터 GenieApiDemoApplication 클래스의 작성을 시작합니다.
GenieApiDemoApplication.java파일을 열고 다음과 같이 GenieApiDemoApplication 클래스가 Application 클래스를 상속하도록 수정합니다.

public class GenieApiDemoApplication extends Application { … }

'extends Application' 을 추가하면 GenieApiDemoApplication 빨간색 밑줄이 추가 됩니다. 이때 밑줄 위에서 Ctrl+Shift+O를 누르면 포함해야 할 패키지와 Override 되어야 할 메소드가 자동으로 추가됩니다.

TIP. **이클립스에서 추가되지 않은 패키지 추가는 'Ctrl+Shift+O'입니다.**
추가적으로, 소스 작성시 코드 자동 완성 기능은 'Ctrl+Space' 입니다.

재정의 할 메소드는 onCreate()와 onTerminate()입니다. 여기서 onCreate()는 App이 처음 시작될 때 호출되는 메소드입니다. 우리는 GenieApiDemo App이 처음 시작될 때 GenieContorlService에서 제공하는 HovisGenieService에 연결하는 메소드인 connect Service()를 실행하므로 이를 추가합니다. 반대로, App이 종료될 때는 onTerminate()가 호출됩니다. App이 종료될 때에는 서비스의 연결을 종료하므로 disconnectService() 메소드를 추가합니다.

```
...
import android.app.Application;
...
public class GenieApiDemoApplication extends Application {
...
@Override
    public void onCreate() {
        super.onCreate();
        a
        if ( !connectService() )
            Log.i("service", "not connected");
    }

    @Override
    public void onTerminate() {
        super.onTerminate();

        disconnectService();
    }
}
```

connectService()와 disconnectService()는 서비스를 연결하고 종료하는 역할을 합니다. 이 메소드들을 정의하기 앞서 서비스 인터페이스 객체를 선언합니다.

```
import com.dongburobot.HovisGenieServiceInterfaces.IHovisGenieService;

public class GenieApiDemoApplication extends Application {
public IHovisGenieService mBinder = null; // 서비스 인터페이스 객체
...
    ...
    ...
}
```

mBinder는 IHovisGenieService의 객체로서, 앞으로 모든 로봇 서비스의 이용은 mBinder를 통해 이루어지기 때문에 중요한 객체입니다. 여기서 IHovisGenieService 클래스는 GenieControlService 내부의 서비스인 HovisGenieService에 접근하기 위한 인터페이스입니다. IHovisGenieService는 Hovis Service.jar에 정의되어 있습니다. HovisService.jar 파일에 정의된 IHovisGenie Service 클래스의 패키지 네임스페이스는 com.dongburobot.HovisGenieService Interfaces.IHovisGenieService입니다. 따라서, IHovisGenieService를 사용하기 위해서 이를 import합니다.

세부적으로 IHovisGenieService는 AIDL(Android Interface Definition Language) 로 작성되어 IPC(Inter Process Communication)을 담당합니다. 여기서는 서비스에 바인딩 하여 로봇 서비스를 사용하는 것에 초점을 맞추므로 자세한 내용을 생략합니다. IPC관련 AIDL에 관심이 있으신 분은 안드로이드 Developer 홈페이지를 참조하십시오.
– http://developer.android.com/guide/components/aidl.html

이어서 [Listing. 4-1]을 참조하여, connectService(), disconnectService(), 그리고 is-ServiceRunning()을 추가합니다. 그리고 19~20줄에 있는 변수 및 객체도 아울러 추가합니다.

마지막으로 GenieApiDemoApplication 클래스는 안드로이드 기본 Application 클래스를 상속받아 작성하였으므로, AndroidManifest.xml에 이를 명시합니다. 다음과 같이 11번째 줄에 android:name=".GenieApiDemoApplication"을 추가합니다.

```
1: <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2:     package="com.dongburobot.genieapidemo"
3:     android:versionCode="1"
4:     android:versionName="1.0" >
5:
6:     <uses-sdk
7:         android:minSdkVersion="10"
8:         android:targetSdkVersion="10" />
9:
10:    <application
11:        android:name=".GenieApiDemoApplication"
12:        android:icon="@drawable/ic_launcher"
13:        android:label="@string/appa_name"
14:        android:theme="@style/AppTheme" >
15:        <activity
16:            android:name=".MainActivity"
17:            android:label="@string/title_activity_main" >
18:            <intent-filter>
19:                <action android:name="android.intent.action.MAIN" />
20:                <category android:name="android.intent.category.LAUNCHER" />
21:            </intent-filter>
22:        </activity>
23:    </application>
24:
25: </manifest>
```

[Listing 4-2] AndroidManifest.xml

(1) 로봇 서비스 연결

우리의 App과 로봇서비스는 App이 처음 실행될 때 연결합니다. App이 시작되면 GenieApi DemoApplication의 onCreate()를 호출됩니다. 그리고 onCreate() 내부에서 connectService()를 호출하여 서비스에 바인딩합니다. connectService()에 의해 연결된 로봇 서비스는 App이 종료될 때 까지 그 연결이 지속됩니다.

connectService()는 그 내부에서 isServiceRunning()을 통해 서비스가 백그라운드에서 동작하는지 확인하고, bindService()를 통해 로봇 서비스에 바인딩합니다. 그리고 바인딩 결과에 따라 true 또는 false로 반환합니다.

```
@Override
public void onCreate() {
    super.onCreate();

    if ( !connectService() )
        Log.i("service", "not connected");
}

protected boolean connectService() {
        if ( isServiceRunning() ) {
            mIntent = new Intent(IHovisGenieService.class.getName());
            mIsBinded = bindService(mIntent, mConnection, BIND_AUTO_CREATE);
            return mIsBinded;
        }

        return false;
    }
}
```

우선 connectService()가 수행되면 33번째 줄의 isServiceRunning()를 호출하여 현재 MID에서 로봇 서비스(GenieServiceControl)이 수행되고 있는 지 확인합니다.
isServiceRunning()는 현재 로봇 서비스가 백그라운드에서 동작할 때 true를 반환하고, 그렇지 않으면 false를 반환합니다.
만약 현재 백그라운드에서 로봇 서비스가 동작하고 있으면, bindService()를 호출하여 실제로 로봇 서비스에 바인딩합니다. binsService()는 android.content에 정의되어 있으며, 원형은 다음과 같습니다.

```
public abstract boolean bindService(Intent service, ServiceConnection conn, int flags)
```

파라미터

service	서비스를 연결할 대상을 정의합니다. 인텐트는 서비스에 정의 된 인텐트 필터와 매치되는 컴포넌트 네임 등을 지정합니다.
conn	서비스가 시작되거나 중단될 때 정보를 받는 객체를 지정합니다.
flags	서비스 바인딩 옵션, BIND_AUTO_CREATE, BIND_DEBUG_UNBIND, BIND_NOT_FOREGROUND, BIND_ABOVE_CLIENT, BIND_ALLOW_OOM_MANAGEMENT, or BIND_WAIVE_PRIORITY.

반환값

서비스 바인딩에 성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

```
mIsBinded = bindService(mIntent, mConnection, BIND_AUTO_CREATE);
```

bindService()의 첫번째 인자는 인텐트입니다. 인텐트는 하나의 App 컴포넌트가 다른 App 컴포넌트의 실행을 위해 보내는 객체입니다. 여기서는, 우리의 App이 GenieControlService에게 bindService()를 통해 인텐트를 보냄으로써, GenieControlService에서 제공하는 로봇 서비스들을 사용하겠다는 메시지를 보내는 것입니다. 이 인텐트는 connectService()안에 다음과 같이 선언되었습니다.

```
mIntent = new Intent(IHovisGenieService.class.getName());
```

GenieContorlService의 AndroidManifest.xml 중에서.

```
<intent-filter>
<action android:name="com.dongburobot.HovisGenieServiceInterfaces.IHovisGenieService"/>
</intent-filter>
```

Intent에 생성자에 인자인 IHovisGenieService.class.getName()은 로봇 서비스 인터페이스의 클래스명인 com.dongburobot.HovisGenieServiceInterfaces.IHovisGenieService를 반환합니다. 그리고 mIntent는 이 클래스명을 가진 Intent의 객체가 할당됩니다. mIntent가 로봇 인터페이스의 클래스명을 가지는 이유는 로봇 서비스에 정의 된 인텐트 필터가 다음과 같기 때문입니다.

bindService()의 두번째 인자는 서비스가 시작하거나 끝날 때, 이에 대한 정보를 수신하는 ServiceConnection 클래스의 객체입니다. ServiceConnection의 객체인 mConnection는 22~31번째 줄에서 선언하였습니다.

```
public ServiceConnection mConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName name, IBinder service) {
            mBinder = IHovisGenieService.Stub.asInterface(service);
        }

        public void onServiceDisconnected(ComponentName name) {
            mBinder = null;
        }
    };
```

onServiceConnected()는 서비스가 바인딩 되었을 때 콜백(시스템 자동적으로 메서드를 호출)되며 mBinder에 연결된 서비스의 인터페이스를 할당합니다. 반면에, onServiceDisconnected()는 서비스가 단절되었을 때 콜백되며, 서비스를 더 이상 유효하지 않으므로 mBinder는 null로 설정합니다. 즉, mConnection는 로봇 서비스의 연결상태에 따라 mBinder를 관리하는 역할을 합니다.

bindService()의 마지막 세번째 인자는 서비스 바인딩 옵션입니다. BIND_AUTO_CREATE는 바인딩이 되어 있는 한 서비스를 자동으로 생성해줍니다. 기타 여러 가지 옵션이 있지만 거의 사용되지 않습니다.

bindService()를 호출하면 서비스 연결 상태에 따라 ServiceConnection 클래스의 객체 mConnection에 정의 된 두 개의 재정의 된 메소드에 따라 IHovisGenieService의 객체 mBinder에 인터페이스르 할당합니다. 그리고 connectService()가 모두 수행되고 나면 App이 서비스를 사용할 준비를 마치게 됩니다.

(2) 로봇 서비스 연결해제

서비스의 해제는 GenieApiDemoApplication 클래스의 onTerminate() 내부의 disconnectService() 호출을 통해 이루어집니다. onTerminate()는 App이 종료될 때 자동으로 호출되므로, App 종료시 서비스도 함께 종료됩니다.

```
@Override
public void onTerminate() {
    super.onTerminate();

    disconnectService();
}
```

```
protected void disconnectService() {
    if (mIsBinded) {
        if (mConnection != null) {
            unbindService(mConnection);
            mConnection = null;
        }
    }

    mIntent = null;
}
```

disconnectService()가 호출되면 mIsBinded를 통해 현재 서비스가 바인드 된 상태인지 확인합니다. 서비스가 바인딩 된 상태라면 mConnection이 null이 아닌지 확인하고 unbindService()를 호출합니다. unbindService의 원형은 다음과 같습니다.

```
public abstract void unbindService(ServiceConnection conn)
```

파라미터

conn	서비스 연결시bindService()에 적용된ServiceConnection의 객체

```
unbindService(mConnection);
```

unbindService()는 현재 연결된 서비스를 해제합니다. unbindService()의 인자는 서비스 연결 시 bindService()에 두번째 인자로 사용하였던 ServiceConnection의 객체입니다.

unbindService()를 호출하면 서비스의 연결이 해제 됨과 동시에 인자로 전달된 mConnection에 재정의 된 onServiceDisconnected()를 호출합니다. 그리고 mBinder에 null을 설정하여 서비스 연결해제를 완료합니다.

```
public ServiceConnection mConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName name, IBinder service) {
            mBinder = IHovisGenieService.Stub.asInterface(service);
        }

        public void onServiceDisconnected(ComponentName name) {
            mBinder = null;
        }
    };
```

서비스 연결해제를 정확히 해줌으로서 로봇 시스템의 이상동작을 방지하는 효과도 있습니다. 서비스 연결을 하면 반드시 연결해제를 해주는 습관은 그래서 중요합니다. 파일 입출력 시 파일을 열면 닫아주는 프로그래밍 습관이 중요하듯이, 로봇 서비스 연결시에도 연결 후 연결해제를 반드시 해주는 습관을 가집시다.

4.3 로봇 시스템을 위한 BaseActivity만들기

안드로이드 장치에서 하나의 화면은 하나의 Activity로 구성됩니다. App 제작시 하나의 화면을 만든다는 것은 하나의 Activity를 만드는 것입니다. 그러므로 보통의 스마트폰 App 에서는 하나의 화면은 android.app.Activity에 정의 된 Activity 클래스를 상속받아 제작합니다.

로봇 App도 하나의 화면을 위해 Activity 클래스를 상속받아 만들 수 있습니다. 그러나 여타 스마트폰 App과 달리 다음과 같은 두 가지를 Activity 제작 전에 고려하여야 합니다.

고려사항

1. **로봇은 한대이므로 제어의 동기화가 필요**
2. **App 화면의 직접적인 On/Off 제어가 필요**

첫째, 로봇은 한대이므로 제어의 동기화가 필요합니다. 제어의 동기화가 보장되지 않으면 로봇의 정확한 동작을 보장할 수 있습니다. 그러나 안드로이드의 Activity 관리는 제어 동기화를 보장하지 않을 수 있는 잠정적인 위험이 있습니다. 그 위험은 하나의 화면이 사라져도 그 화면을 표시하는 Activity가 종료되지 않기 때문입니다.

예를 들어, 로봇전진Activity가 있고 이 Activity가 로봇 전진 명령을 내리고 있는 가운데, 로봇이 배터리 전원이 부족해서 충전모드로 진입하여 화면에는 충전Activity가 나왔다고 가정합니다. 이 때, 로봇전진Activity가 종료되지 않고 스택에 쌓여서 계속 로봇전진 명령을 내린다면, 충전Activity에서 수행해 주여야 할 자동충전 주행에 영향을 미칠 수가 있습니다.
물론, 이런 상황은 극히 드물게 발생할 수 있습니다. 또한, 개발자가 이러한 병목상황을 사전에 예상하고 그에 대처할 수 있도록 세심하게 코드를 작성할 수도 있습니다. 그러나 가장 좋은 방법은 이러한 상황이 아예 발생하지 않도록 사전에 차단하는 것입니다.

우리가 작성 할 로봇 App에서는 하나의 Activity가 화면에서 사라지면 그 Actvitity가 완전히 종료되도록 합니다. Activity가 화면에서 사라질 때 반드시 호출되는 메소드는 onPause()입니다. 그러므로, onPause()에서 Activity가 완전히 종료되도록 보장하여야 합니다.

```
 1: package com.dongburobot.genieapidemo;
 2:
 3: import android.app.Activity;
 4: import android.os.Bundle;
 5: import android.view.Window;
 6: import android.view.WindowManager;
 7:
 8: public class BaseActivity extends Activity {
 9:
10:     protected GenieApiDemoApplication myRobotApp;
11:
12:     @Override
13:     protected void onCreate(Bundle savedInstanceState) {
14:         super.onCreate(savedInstanceState);
15:
16:         // 화면 설정
17:         this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
18:                 WindowManager.LayoutParams.FLAG_FULLSCREEN);
19:         this.getWindow().addFlags(WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED
20:                 | WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD
21:                 | WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON
22:                 | WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON);
23:         this.requestWindowFeature(Window.FEATURE_NO_TITLE);
24:
```

```
25:         myRobotApp = (GenieApiDemoApplication)this.getApplicationContext();
26:     }
27:
28:     @Override
29:     protected void onPause() {
30:         super.onPause();
31:
32:         this.finish();
33:     }
34: }
```

[Listing 4-2] BaseActivity.java

BaseActivity의 13번째 줄 onCreate()는 Activity가 생성되는 시점에 호출됩니다. onCreate()에서는 Activity가 시작할 때 필요한 부분을 작성합니다. 현재 필요한 것은 화면을 설정하는 코드입니다. 이는 고려사항 2에 해당합니다.

```
// 화면 설정 (고려사항 2)
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        this.getWindow().addFlags(WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED
                | WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD
                | WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON
                | WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON);
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);
```

17~23번째 줄의 코드는 화면을 설정하는 코드입니다. 17~18번째 코드는 현재 화면을 FULL SCREEN으로 사용하기 위해 설정합니다. 19~22번째 줄 코드는 화면이 락이 걸려도 화면을 보여주고, 안드로이드 키 입력화면을 보여주지 않으며, 스크린을 항상 키고, 킨 상태를 유지하는 flag를 사용하여 화면을 설정합니다. 23번째줄 코드는 화면의 타이틀바를 없앱니다. 기타 이 밖에도 화면을 설정할 수 있는 많은 flag가 있습니다. 자세한 내용은 안드로이드 개발자 사이트를 참조합니다.

29번째 줄 onPause()는 Activity가 화면이 사라지는 시점에 호출됩니다. 앞서 언급했듯이 우리의 로봇 App은 로봇 제어의 동기화를 위해서 스택에 Activity를 남기지 않습니다. 그러므로 32번 줄에 finish()를 이용하여 Activity가 완전히 종료되도록 합니다. 이는 고려사항 1에 해당합니다.

```
@Override
protected void onPause() {
    super.onPause();
    // 고려사항 1에 해당
    this.finish();
```

마지막으로 GenieApplicationDemoApplication의 객체가 10번째 줄에 선언되어 있습니다. myRobotApp은
getApplicationContext() 에 의해 반환 된 현재 프로세스의 전역 Application 객체를 할당합니다. onCreate() 내부 25번째 줄의 코드는 BaseActivity를 상속받아 작성되는 모든 Activity에서 GenieApiDemoApplication 클래스가 포함하는 객체와 변수들을 전역으로 사용이 가능합니다. 이로서 로봇 서비스를 모든 Activity에서 사용할 수 있는 기반을 설정합니다.

```
protected GenieApiDemoApplication myRobotApp;
...
myRobotApp = (GenieApiDemoApplication)this.getApplicationContext();
```

이로써 로봇 프로그래밍을 시작할 수 있는 기본적인 설정을 완료하였습니다. 이번 장에서는 로봇을 제어하기 위해서 로봇 서비스를 이용하는 방법에 대해서 살펴보았습니다. 다음 장에서는 로봇 서비스에서 제공해주는 함수를 살펴봅니다.

05 HOVIS Genie API (Application Programing Interface)

Hovis Genie를 제어하기 위한 서비스는 MID에서 백그라운드로 동작중인GenieContorlService에 의해 이루어집니다. 유저는 GenieControlService로 바인딩을 함으로써 로봇 서비스를 사용할 수 있습니다.

이전 장에서는 로봇 서비스를 사용하기 위해 서비스로의 바인딩하는 법에 대해서 알아보았습니다. 이번 장에서는 본격적인 로봇 프로그래밍에 앞서 Hovis Genie API에서 제공하는 함수를 살펴봅니다. Hovie Genie API에서 제공하는 기능은 다음과 같습니다.

구동부 함수

구동부 함수는 Hovis Genie의 옴니휠 구동부에 관련한 함수들의 집합입니다. 이 함수들은 구동부 바퀴의 각 종 설정 및 움직임 여부를 주로 설정하는데 사용됩니다. 실제 로봇 구동은 네비게이션 함수를 통해 이루어집니다.

네비게이션 함수

네비게이션 함수는 Hovis Genie를 실제로 움직이는 역할을 담당합니다. 이 함수들은 옴니휠의 구동 방식을 추상화하여 x, y, theta의 좌표로 이동할 수 있는 역할을 합니다.

센서 함수

센서 관련 함수는 Hovie Genie에 부착 된 각 종 센서(거리 감지 센서, 하단 감지 센서, 터치센서 등) 를 제어할 수 있도록 인터페이스를 제공합니다.

TTS(Text-to-Speech) 함수

TTS 함수는 문자열 입력을 로봇의 음성으로 변환하는 함수입니다.

사운드(효과음) 함수

사운드 함수는 로봇의 상황에 따라 여러 가지 효과음을 발생시키는 함수입니다.

멀티미디어(오디오 및 비디오) 함수

로봇에서 오디오 및 비디오 파일(MP3, MP4, AVI …)을 재생하거나 정지하기 위한 기능을 제공합니다.

모션 및 서보 모터 관련 함수

모션 및 서보 모터 관련 함수는 로봇 모션에 관련된 기능을 제공합니다.

5.1 구동부 함수

dmel_drive_servo_on

```
boolean dmel_drive_servo_on(ComponentName cn)
                        throws android.os.RemoteException
```
로봇 구동부 (DRS-0102) 서보 모터 3개를 “Servo ON” 한다
Parameters:
cn – Component 이름
Returns:
성공하면 true, 실패하면 false 반환
Throws:
android.os.RemoteException

dmel_drive_set_movable

```
void dmel_drive_set_movable(ComponentName cn,
                        boolean flag)
                          throws android.os.RemoteException
```
로봇 구동부의 구동 가능여부를 설정한다.
Parameters:
cn – Component 이름
flag – 구동 가능 설정 true, 불가 설정 false
Throws:
android.os.RemoteException

dmel_drive_get_movable

```
boolean dmel_drive_get_movable(ComponentName cn)
                          throws android.os.RemoteException
```
로봇 구동부의 구동여부를 반환한다.
Parameters:
cn – Component 이름
Returns:
로봇 구동부가 구동 가능시 true, 아니면 false 반환
Throws:
android.os.RemoteException

dmel_robot_set_horizontal_velocity

```
boolean dmel_robot_set_horizontal_velocity(ComponentName cn,
                                           double vel,
                                           double ang)
                                    throws android.os.RemoteException
```

로봇 구동부의 평행이동 속도를 설정한다.
Parameters:
cn – Component 이름
vel – 평행이동속도 (단위 : m/s)
ang – 평행에서 보정된 각도 (단위 : rad이며 값의 +는 CCW, –는 CW)
Returns:
Throws:
android.os.RemoteException

dmel_robot_set_rotation_velocity

```
boolean dmel_robot_set_rotation_velocity(ComponentName cn,
                                         double vel)
                                    throws android.os.RemoteException
```

로봇 구동부의 각속도를 설정한다.
Parameters:
cn – Component 이름
vel – 각속도 (rad/s)
Returns:
성공시 true, 실패시 false 반환
Throws:
android.os.RemoteException

dmel_robot_set_velocity

```
boolean dmel_robot_set_velocity(ComponentName cn,
                                double lin,
                                double ang,
                                double horizontal_ang,
                                boolean flag)
                                throws android.os.RemoteException
```

로봇의 구동부를 구동한다.
Parameters:
cn – Component 이름
lin – 선속도 (m/s)
ang – 각속도 (rad/s) +는 CCW, –는 CW
horizontal_ang – 지정된 값만큼 (+는 CCW, –는 CW) 보정된 각도로 평행이동
flag – true이면 평행이동하며 horizontal_ang값에 의해 이동, false면 회전이동 ang값에 의해 이동
fThrows:
android.os.RemoteException

dmel_param_set_max_lin_velocity

boolean dmel_param_set_max_lin_velocity(ComponentName cn,
double vel)
throws android.os.RemoteException
로봇 구동부의 최대 선속도를 설정한다.
Parameters:
cn – Component 이름
vel – 최대 선속도 (m/s)
Returns:
설정 성공시 true, 실패시 false
Throws:
android.os.RemoteExceptio

dmel_param_get_max_lin_velocity

double dmel_param_get_max_lin_velocity(ComponentName cn)
throws android.os.RemoteException
로봇 구동부의 최대 각속도를 반환한다.
Parameters:
cn – Component 이름
Returns:
최대 선속도 반환 (m/s)
Throws:
android.os.RemoteException

dmel_param_set_max_ang_velocity

boolean dmel_param_set_max_ang_velocity(ComponentName cn,
double vel)
throws android.os.RemoteException
로봇 구동부의 최대 각속도를 설정한다.
Parameters:
cn – Component 이름
vel – 최대 각속도 (rad/s)
Returns:
설정 성공시 true, 실패시 false
Throws:
android.os.RemoteException

dmel_param_get_max_ang_velocity

double dmel_param_get_max_ang_velocity(ComponentName cn)
throws android.os.RemoteException
로봇 구동부의 최대 각속도를 반환한다.
Parameters:
cn – Component 이름
Returns:
최대 각속도 반환 (rad/s)
Throws:
android.os.RemoteException
android.os.RemoteException

5.2 네비게이션 함수

dmel_robot_set_pose

boolean dmel_robot_set_pose(ComponentName cn,
double x,
double y,
double theta)
throws android.os.RemoteException
로봇의 현재 위치를 x, ,y, theta로 설정한다. 로봇의 현재 위치를 (0, 0, 0)으로 설정하면 현 위치가 기준점이 된다.
Parameters:
cn – Component 이름
x – (meter)
y – (meter)
theta – (Radian –PI ~ +PI)
Returns:
성공시 true, 실패시 false 반환
Throws:
android.os.RemoteException

dmel_robot_get_pose

double[] dmel_robot_get_pose(ComponentName cn)
throws android.os.RemoteException
시작점 기준으로 로봇의 현 위치를 반환한다.
Parameters:
cn – Component 이름
Returns:
시작점 기준 double[0] ~ double[2]에 각각 x, y, theta(Radian –PI ~ +PI) 값 반환
Throws:
android.os.RemoteException

dmel_navigate_move_pose

```
void dmel_navigate_move_pose(ComponentName cn,
                             double x,
                             double y,
                             double theta)
                             throws android.os.RemoteException
```

로봇의 시작점을 기준으로 (x, y, theta) 지점으로 이동한다.
Parameters:
cn – Component 이름
x – 시작점 기준 x (meter)
y – 시작점 기준 y (meter)
theta – 시작점에서의 로봇 Heading 기준 로봇 방향 theta (Radian –PI ~ +PI)
Throws:
android.os.RemoteException

dmel_navigate_prepare

```
void dmel_navigate_prepare(ComponentName cn)
                           throws android.os.RemoteException
```

로봇 네비게이션을 위한 초기화를 수행한다. 초기화시 로봇의 현 위치를 (0, 0, 0)으로 설정하며, 현재 Heading을 0도로 설정한다.
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_navigate_add_goal_pose

```
boolean dmel_navigate_add_goal_pose(ComponentName cn,
                                    double x,
                                    double y,
                                    double theta,
                                    boolean rflag,
                                    int time)
                                    throws android.os.RemoteException
```

로봇이 방문할 지점을 추가한다. 로봇에 설정된 시작점 (0, 0, 0)을 기준으로 방문할 지점을 결정한다. (dmel_navigate_prepare() 참조) 로봇은 설정된 방문 지점을 dmel_navigate_start()를 통해 방문을 시작한다.
Parameters:
cn – Component 이름
x – 시작점 기준 x (meter)
y – 시작점 기준 y (meter)
theta – 시작점에서의 로봇 Heading 기준 로봇 방향 theta (Radian –PI ~ +PI)
rflag – true시 로봇이 방문지점 도착시 theta도 방향을 바라봄
time – 로봇이 방문지점 도착시의 대기시간 (sec)
Returns:
Throws:
android.os.RemoteException

dmel_navigate_start

boolean dmel_navigate_start(ComponentName cn)
throws android.os.RemoteException
로봇 네비게이션 수행한다. 로봇 네비게이션은 미리 설정된 goal_pose를 한 지점 씩 방문한다.
Parameters:
cn – Component 이름
Returns:
성공시 true, 실패시 false 반환
Throws:
android.os.RemoteException

dmel_navigate_stop

boolean dmel_navigate_stop(ComponentName cn)
throws android.os.RemoteException
로봇 네비게이션 수행을 중단한다.
Parameters:
cn – Component 이름
Returns:
성공시 true, 실패시 false 반환
Throws:
android.os.RemoteException

5.3 센서 함수

dmel_robot_get_obs

double[] dmel_robot_get_obs(ComponentName cn)
throws android.os.RemoteException
전방 감지 PSD 센서 값을 반환하다. 전방부터 시계방향으로 0, 1, 2, 3, 4
Parameters:
cn – Component 이름
Returns:
전방 감지 센서 5개의 값을 double[0] ~ double[4]에 반환
Throws:
android.os.RemoteException

dmel_robot_get_cliff

double[] dmel_robot_get_cliff(ComponentName cn)
throws android.os.RemoteException
바닥 감지 PSD 센서 값을 반환한다. 전방부터 시계방향으로 0, 1, 2
Parameters:
cn – Component 이름
Returns:
바닥 감지 센서 3개의 값을 double[0] ~ double[2]에 반환
Throws:
android.os.RemoteException

dmel_hri_get_head_touch_info

boolean dmel_hri_get_head_touch_info(ComponentName cn)
throws android.os.RemoteException
머리 터치 여부를 반환한다.
Parameters:
cn – Component 이름
Returns:
머리 터치시 true, 아니면 false를 반환
Throws:
android.os.RemoteException

dmel_hri_get_hand_touch_info

boolean[] dmel_hri_get_hand_touch_info(ComponentName cn)
throws android.os.RemoteException
양손의 터치 여부를 반환한다.
Parameters:
cn – Component 이름
Returns:
터치시 true, 아니면 false를 반환. boolean[2] (boolean[0] : 왼손터치, boolean[1] : 오른손터치)
Throws:
android.os.RemoteException

5.4 TTS(Text-to-Speech) 함수

dmel_tts_speak

```
void dmel_tts_speak(ComponentName cn,
                    java.lang.String text)
                    throws android.os.RemoteException
```

TTS(Text-to-Speech)를 Play한다.
Parameters:
cn – Component 이름
text – 음성으로 변환할 문자열 String ex)"안녕하세요. 호비스지니입니다."
Throws:
android.os.RemoteException

dmel_tts_stop

```
void dmel_tts_stop(ComponentName cn)
                   throws android.os.RemoteException
```

TTS(Text-to-Speech)를 중단한다.
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

5.5 사운드(효과음) 함수

dmel_sound_play_intro

```
void dmel_sound_play_intro(ComponentName cn)
                           throws android.os.RemoteException
```

효과음을 재생한다. (Intro 효과음)
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_sound_play_chimes

```
void dmel_sound_play_chimes(ComponentName cn)
                        throws android.os.RemoteException
```

효과음을 재생한다. (Chimes 효과음)
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_sound_play_dingdong

```
void dmel_sound_play_dingdong(ComponentName cn)
                          throws android.os.RemoteException
```

효과음을 재생한다. (딩동 효과음)
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_sound_play_horn

```
void dmel_sound_play_horn(ComponentName cn)
                      throws android.os.RemoteException
```

효과음을 재생한다. (Horn 효과음)
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_sound_play_fail

```
void dmel_sound_play_fail(ComponentName cn)
                      throws android.os.RemoteException
```

효과음을 재생한다. (실패 효과음)
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

[음성인식 관련 함수]

dmel_voice_recognition_start

void dmel_voice_recognition_start(ComponentName cn)
throws android.os.RemoteException
음성인식시 음성 입력을 시작한다.
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_voice_recognition_stop

void dmel_voice_recognition_stop(ComponentName cn)
throws android.os.RemoteException
음성인식시 음성 입력을 끝낸다.
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_voice_recognition_set_mode

void dmel_voice_recognition_set_mode(ComponentName cn,
boolean is_auto)
throws android.os.RemoteException
음성인식을 할 수 있도록 모드를 전환한다
Parameters:
cn – Component 이름
is_auto – true면 자동으로 음성인식 시작과 끝을 반복
Throws:
android.os.RemoteException

dmel_voice_recognition_get_mode

boolean dmel_voice_recognition_get_mode(ComponentName cn)
throws android.os.RemoteException
음성인식 가능여부를 반환한다.
Parameters:
cn – Component 이름
Returns:
음성인식 가능하면 true, 아니면 false
Throws:
android.os.RemoteException

5.6 멀티미디어(오디오 및 비디오) 함수

dmel_audio_set_volume

```
void dmel_audio_set_volume(ComponentName cn,
                           float volume)
                           throws android.os.RemoteException
```

볼륨의 크기를 설정한다. 최소볼륨 : 0, 최대볼륨 : 1
Parameters:
cn – Component 이름
volume – 볼륨의 크기 설정 (0~1)
Throws:
android.os.RemoteException

dmel_audio_get_volume

```
float dmel_audio_get_volume(ComponentName cn)
                            throws android.os.RemoteException
```

현재 볼륨의 크기를 반환한다. 최소볼륨 : 0, 최대볼륨 : 1
Parameters:
cn – Component 이름
Returns:
볼륨의 크기를 반환 (0~1)
Throws:
android.os.RemoteException

dmel_audio_play_mp3

```
void dmel_audio_play_mp3(ComponentName cn,
                         java.lang.String path)
                         throws android.os.RemoteException
```

MP3 파일을 재생한다. (기본경로는 /sdcard/dongbu/ 로 지정되어 있음)
실제 경로가 /sdcard/dongbu/audio/test.mp3 일 경우
mBinder.dmel_audio_play_mp3(getComponentName() , “audio/test,mp3”);
Parameters:
cn – Component 이름
path – MP3파일 경로 (파일이름포함, 기본경로제외)
Throws:
android.os.RemoteException

dmel_audio_stop_mp3

void dmel_audio_stop_mp3(ComponentName cn)
throws android.os.RemoteException
MP3 재생을 중지한다.
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_video_play_mp4

void dmel_video_play_mp4(ComponentName cn,
java.lang.String path)
throws android.os.RemoteException
MP4 파일을 재생한다. (기본경로는 /sdcard/dongbu/ 로 지정되어 있음)
실제 경로가 /sdcard/dongbu/video/test.mp3 일 경우
mBinder.dmel_video_play_mp4(getComponentName() , “video/test,mp3”);
Parameters:
cn – Component 이름
path – MP4파일 경로 (파일이름포함, 기본경로제외)
Throws:
android.os.RemoteException

5.7 모션 및 서보 모터 관련 함수

dmel_motion_play

void dmel_motion_play(ComponentName cn,
java.lang.String motion)
throws android.os.RemoteException
로봇 모션 파일을 재생한다. (기본경로는 /sdcard/dongbu/motion/ 으로 지정되어 있음)
Parameters:
cn – Component 이름
motion – 모션파일이름 ex) “01n_shake_head.dmt”
Throws:
android.os.RemoteException

dmel_motion_play2

```
void dmel_motion_play2(ComponentName cn,
                       java.lang.String motion,
                       boolean repeat,
                       boolean sync)
                         throws android.os.RemoteException
```

로봇 모션 파일을 재생한다. 미디어 Sync 여부 및 반복 설정이 가능하다.
(경로는 /sdcard/dongbu/motion/ 으로 지정되어 있음)
Parameters:
cn – Component 이름
motion – 모션파일이름 ex) “01n_shake_head.dmt”
repeat – 정지 명령이 들어올때까지 계속해서 반복 실행됨.
sync – 미디어와 종료 맞춤. (repeat 보다 우선시 됨)
Throws:
android.os.RemoteException

dmel_motion_stop

```
void dmel_motion_stop(ComponentName cn)
                      throws android.os.RemoteException
```

현재 실행중인 모션을 중단한다
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_motion_servo_on

```
boolean dmel_motion_servo_on(ComponentName cn,
                             int id)
                             throws android.os.RemoteException
```

ID값에 해당하는 상체 모터에 “Servo ON”을 설정한다.
Parameters:
cn – Component 이름
id – id 모터의 ID
(0: 오른쪽 어깨 모터, 1: 오른팔 상박 모터, 2: 오른팔 하박 모터, 3: 왼쪽 어깨 모터,
4: 왼쪽 상박 모터, 5: 왼쪽 하박 모터, 18: 머리 모터, 19: 허리 모터)
Returns:
성공시 true, 실패시 false를 반환
Throws:
android.os.RemoteException

dmel_motion_servo_off

boolean dmel_motion_servo_off(ComponentName cn,
int id)
throws android.os.RemoteException
ID값에 해당하는 상체 모터에 “Servo OFF”를 설정한다.
Parameters:
cn – Component 이름
id – id 모터의
ID (0: 오른쪽 어깨 모터, 1: 오른팔 상박 모터, 2: 오른팔 하박 모터, 3: 왼쪽 어깨 모터,
4: 왼쪽 상박 모터, 5: 왼쪽 하박 모터, 18: 머리 모터, 19: 허리 모터)
Returns:
성공시 true, 실패시 false를 반환
Throws:
android.os.RemoteException

dmel_motion_set_servo

boolean dmel_motion_set_servo(ComponentName cn,
int on)
throws android.os.RemoteException
모든 상체 모터에 “Servo ON”을 설정한다.
Parameters:
cn – Component 이름
on – Servo ON : 1, Servo OFF : 0
Returns:
성공하면 true, 실패 시 false 반환
Throws:
android.os.RemoteException

dmel_motion_get_position

int dmel_motion_get_position(ComponentName cn,
int id)
throws android.os.RemoteException
ID값에 해당하는 상체 모터의 현재 위치값을 반환한다.
Parameters:
cn – Component 이름
id – 모터의 ID (0: 오른쪽 어깨 모터, 1: 오른팔 상박 모터, 2: 오른팔 하박 모터, 3: 왼쪽 어깨
모터, 4: 왼쪽 상박 모터, 5: 왼쪽 하박 모터, 18: 머리 모터, 19: 허리 모터)
Returns:
해당 모터의 현재 위치값 반환
Throws:
android.os.RemoteException

dmel_motion_set_calib_value

```
boolean dmel_motion_set_calib_value(ComponentName cn,
                                    int id,
                                    int val)
                                    throws android.os.RemoteException
```

ID값에 해당하는 상체 모터에 영점 조절값을 설정한다.
Parameters:
cn – Component 이름
id – 모터의 ID (0: 오른쪽 어깨 모터, 1: 오른팔 상박 모터, 2: 오른팔 하박 모터, 3: 왼쪽 어깨 모터, 4: 왼쪽 상박 모터, 5: 왼쪽 하박 모터, 18: 머리 모터, 19: 허리 모터)
val – 영점 조절 값 (–127 ~ 127)
Returns:
영점 조절이 성공하면 true, 실패하면 false 반환
Throws:
android.os.RemoteException

5.8 머리 LED 제어 관련 함수

dmel_hri_set_brow_beam

```
void dmel_hri_set_brow_beam(ComponentName cn,
                            int brightness)
                            throws android.os.RemoteException
```

이마 LED를 설정한다.
Parameters:
cn – Component 이름
brightness – 0 : 끄기, 1 : 약하게 키기, 2 : 중간 키기, 3 : 밝게 키기
Throws:
android.os.RemoteException

dmel_hri_set_eye_led

void dmel_hri_set_eye_led(ComponentName cn,
int eyeCode,
int colorCode)
throws android.os.RemoteException
눈 LED를 설정한다.
Parameters:
cn – Component 이름
eyeCode – 0 : 기존상태유지, 1 : 빙글빙글, 2 : 눈썹깜박, 3 : 눈썹정지, 4 : 전체 켜기, 5 : 전체 깜박, 6 : 좌우 효과, 7 : 상하 효과, 8 : 끄기
colorCode – 1 : RED LED, 2 : BLUE LED, 3 : PURPLE (RED + BLUE) LED
Throws:
android.os.RemoteException

dmel_hri_set_ear_led

void dmel_hri_set_ear_led(ComponentName cn,
int earCode)
throws android.os.RemoteException
귀 LED를 설정한다.
Parameters:
cn – Component 이름
earCode – 0 : 기존상태유지, 1 : 켜기, 2 : 끄기, 3 : 깜박
Throws:
android.os.RemoteException

dmel_hri_set_mouth_led

void dmel_hri_set_mouth_led(ComponentName cn,
int mouthCode)
throws android.os.RemoteException
입 LED를 설정한다.
Parameters:
cn – Component 이름
mouthCode – 0 : 기존상태유지, 1 : 가운데 ON, 2 : 세개 ON, 3 : 가운데 깜박, 4 : 세개다 깜박, 5 : 말하기
Throws:
android.os.RemoteException

06 Hovis Genie 기본예제

Chapter 6에서는 로봇 서비스 이용한 본격적인 로봇 프로그래밍을 시작합니다. 우리는 지난 Chapter 4에서 GenieApiDemo 프로젝트를 생성하여 로봇 서비스를 이용하기 위한 기반 작업을 하였습니다. 이번 장에서는 여러 가지 로봇 제어 관련 예제를 추가하여 GenieApiDemo 프로젝트를 완성하도록 하겠습니다.

GenieApiDemo는 총 7개의 화면이 결합되어 있습니다. 다시 말하면, GenieApiDemo는 7개의 Activity가 결합한 형태의 App입니다. 향후 각 절에서 추가하고 수정하여야 하는 파일은 [표 6-1]과 같으며, GenieApiDemo App에 포함된 7개의 Activity를 보여줍니다.

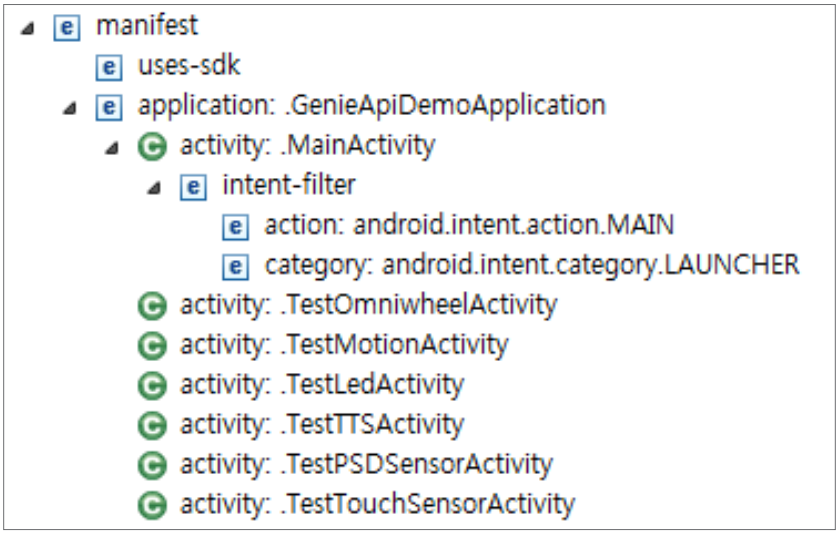
절	내 용	UI 레이아웃	소스코드
6.1.	메인화면	activity_main.xml	MainActivity.java
6.2.	구동부 제어	activity_testomniwheel.xml	TestOmniwheelActivity.java
6.3.	모션 제어	activity_testmotion.xml	TestMotionActivity.java
6.4.	머리 LED 제어	activity_testled.xml	TestLedActivity.java
6.5.	TTS 제어	activity_testtts.xml	TestTTSActivity.java
6.6.	PSD 센서 제어	activity_testpsdsensor.xml	TestPSDSensorActivity.java
6.7.	터치 센서 제어	activity_testtouchsensor.xml	TestTouchSensorActivity.java

[표 6-1] GenieApiDemo의 화면 구성

이번 장의 각 절에서 예제를 하나씩 작성함에 따라 App을 구성하는 Activity가 추가됩니다. 따라서, App의 구조가 변경됩니다. 이러한 변경사항은 AndroidMainfest.xml에 추가 선언하여야 합니다. AndroidManifest.xml을 열고, 프로젝트 생성할 때 이미 추가 된 MainActivity를 제외한 나머지 6개의 Activity를 [Listing 6-1]의 22~27줄을 참고하여 추가합니다.

```
1: <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2:     package="com.dongburobot.genieapidemo"
3:     android:versionCode="1"
4:     android:versionName="1.0" >
5:
6:     <uses-sdk
7:         android:minSdkVersion="10"
8:         android:targetSdkVersion="10" />
9:
10:    <application
11:        android:name=".GenieApiDemoApplication"
12:        android:icon="@drawable/ic_launcher"
13:        android:label="@string/app_name"
14:        android:theme="@style/AppTheme" >
15:        <activity
16:            android:name=".MainActivity">
17:            <intent-filter>
18:                <action android:name="android.intent.action.MAIN" />
19:                <category android:name="android.intent.category.LAUNCHER" />
20:            </intent-filter>
21:        </activity>
22:        <activity android:name=".TestOmniwheelActivity" />
23:        <activity android:name=".TestMotionActivity" />
24:        <activity android:name=".TestLedActivity" />
25:        <activity android:name=".TestTTSActivity" />
26:        <activity android:name=".TestPSDSensorActivity" />
27:        <activity android:name=".TestTouchSensorActivity" />
28:    </application>
29:
30: </manifest>
```

[Listing 4-2] BaseActivity.java



[그림 6-1] AndroidManifest.xml의 구조

AndroidManifest.xml을 통해 바라 본 GenieApiDemo App의 구조는 [그림 6-1]과 같습니다.
GenieApiDemo 〈application〉(=App)은 GenieApiDemoApplication.java에 의해 동작하며, 7개의 〈activity〉를 가집니다. 각 〈activity〉는 MainActivity.java, …, TestTouch-SensorActivity.java에 의해 동작되며, 그 중 MainActivity는 LAUNCHER 카테고리와 MAIN 액션을 가지는 〈intent-filter〉가 있으므로, 런처에서 App을 시작할 때 처음 등장하는 화면 임도 아울러 일러주고 있습니다.

우리는 본격적인 예제 작성에 앞서 일괄적으로 AndroidManifest.xml에 App의 구조를 정의하였습니다. 그러므로 각 절에서 예제를 작성할 때 AndroidManifest.xml을 더 이상 신경 쓰지 않아도 됩니다.
그러나 실제로 App을 개발할 때 많은 개발자가 AndroidManifest.xml에 App의 구조를 제대로 지정하지 않아 많은 에러를 발생시키므로 주의하여야 하겠습니다.

마지막으로, 이번 장부터는 예제를 작성하는 방법을 위주로 필요한 부분을 설명합니다. 부족한 부분은 안드로이드 개발자 사이트 또는 다른 안드로이드 기본서를 참고하시기 바랍니다.

6.1 메인 화면 구성

안드로이드에서 하나의 하면은 하나의 Activity로 구성됩니다. 그리고 이 Activity는 보통 UI의 레이아웃을 구성하는 XML과 자바 파일을 포함합니다. 메인화면도 마찬가지로 XML과 자바 파일을 가집니다. XML파일은 activity_main.xml이며, 자바파일은 MainActivity.java입니다.

activity_main.xml과 MainActivity.java는 지난 4장에서 GenieApiDemo 프로젝트를 생성 할 때, 이미 프로젝트에 자동 포함되었습니다. 그러므로 프로젝트에 추가할 파일은 없습니다.

이클립스 Package Explorer의 GenieApiDemo 〉 res 〉 layout 폴더는 프로젝트에서 사용하는 모든 UI 레이아웃을 포함하고 있습니다. activity_main.xml 파일도 이곳에 위치합니다.
activity_main.xml 파일을 찾아 더블클릭하여 열고 [Listing 6-2]대로 소스를 추가합니다.

```
1: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2:     xmlns:tools="http://schemas.android.com/tools"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent" >
5:
6:         <GridView
7:             android:id="@+id/gridview_apidemos"
8:             android:layout_width="match_parent"
9:             android:layout_height="match_parent"
10:            android:padding="10dp"
11:            android:verticalSpacing="10dp"
12:            android:horizontalSpacing="10dp"
13:            android:numColumns="auto_fit"
14:            android:columnWidth="60dp"
15:            android:stretchMode="columnWidth"
16:            android:gravity="center"
17:            android:background="#222222"
18:        />
19:
20: </RelativeLayout>
```

[Listing 6-2] activity_main.xml

activity_main.xml은 〈RelativeLayout〉을 최상위 태그로 하며 하위에 〈GridView〉를 포함한 구조입니다.
a여기서 RelativeLayout은 화면의 위젯을 어떠한 기준으로부터 상대적으로 배치하는 레이아웃이며, GridView는 화면에 특정 위젯을 격자모양을 배치합니다.

activity_main.xml의 구조에 의해 구성된 UI를 화면에서 나타내면 [그림 6-2]과 같습니다.

[그림 6-2] 메인화면

〈GridView〉는 화면 전체를 차지합니다. 8~9줄의 크기 속성 android:layout_width와 android:layout_height에 지정된 값 "match_parent"는 상위 위젯과 크기를 동일하게 만들어주는데, 상위 속성인 〈ReleativeLayout〉이 화면 전체를 차지하기 때문입니다.

화면 전체를 차지하는 〈GridView〉는 격자모양로 하위 위젯을 포함합니다. 〈GridView〉의 나머지 속성은 하위 위젯을 어떻게 배치하는지에 대한 설정입니다. 현재 하위 위젯 배치는 android:numColumns="auto_fit"으로 설정되어, 하위 위젯의 크기가 한 행에 몇 개의 위젯을 배치하는 가에 대해 영향을 미칩니다.

➔ **ReleativeLayout의 주요속성**

http://developer.android.com/guide/topics/ui/layout/relative.html

➔ **GridView의 주요속성**

http://developer.android.com/guide/topics/ui/layout/gridview.html

〈GridView〉의 하위 위젯은 아이콘 이미지와 텍스트로 구성되어 있습니다. 이를 정의하는 또 다른 XML파일이 필요합니다. /res/layout에 demoitems.xml 파일을 추가하고 소스코드를 다음과 같이 수정합니다.

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="fill_parent"
4:     android:layout_height="fill_parent"
5:     android:orientation="vertical"
6:     android:gravity="center">
7:
8:     <ImageView
9:         android:layout_width="65dp"
10:        android:layout_height="65dp"
11:        android:padding="6dp"
12:        android:gravity="center"
13:        android:id="@+id/demo_icon"/>
14:    <TextView
15:        android:layout_width="65dp"
16:        android:layout_height="30dp"
17:        android:gravity="center"
18:        android:ellipsize="end"
19:        android:singleLine="true"
20:        android:id="@+id/demo_name"/>
21:
22: </LinearLayout>
```

[Listing 6-3] demoitems.xml

demoitems.xml은 〈ImageView〉와 〈TextView〉로 구성되어 있습니다. 이는 [그림6-2]의 이미지 아이콘과 텍스트를 각각 표현합니다. 이는 〈GridView〉에 표현되는 위젯의 구조를 표현한 것일 뿐 실제 데이터 소스를 〈GridView〉에 받아들인 것은 아닙니다. 실제 데이터 소스를 받아들이는 일은 MainActivity.java에서 이루어집니다. 이어서 MainActivity.java를 열고 [Listing 6-4]을 참조하여 소스코드를 작성합니다.

1) BaseActivity 클래스 상속 (18번째 줄)

```
public class MainActivity extends BaseActivity …
```

BaseActivity는 4장에서 작성했던 클래스로, 로봇 프로그래밍의 특성에 따라 Activity 클래스를 재정의 한 클래스입니다. BaseActivity는 MID의 전체화면을 사용하며, Activity가 안드로이드 스택에 쌓이는 것을 방지하는 역할을 합니다. 또한 서비스를 사용하기 위한 전역변수인 mBinder를 사용할 수 있도록 해줍니다.

2) MainActivity 클래스의 멤버변수 선언 (19~23번째 줄)

```
public LinkedList<String> mDemoNameList;
public LinkedList<Integer> mDemoImgList;
private GridView mMainGrid;
private MainAdapter mMainAdapter;
```

mMainGrid는 activity_main.xml에 선언된 GridView를 할당하기 위한 변수입니다. mMainAdapter는 (4)에서 작성할 Inner 클래스 MainAdapter의 객체로서 GridView에 할당할 데이터 소스를 설정하기 위하여 사용됩니다. GridView의 하위 위젯은 아이콘 이미지와 텍스트로 이루어지는데, 이 둘에 해당하는 정보는 각각 mDemoImgList와 mDemoNameList에 할당합니다.

3) onCreate() 작성 (26~50번째 줄)

(2)에서 선언된 멤버변수를 활용하는 소스코드가 들어갑니다.

28번째 줄에서는 setContentView()를 이용하여 MainActivity에 activity_main.xml을 할당합니다. 30~44번째 줄에는 GridView에 하위 위젯에 출력할 데이터를 LinkedList에 할당합니다. 46~48번째 줄은 GridView에 Adapter를 할당하는 코드입니다. Adapter란 다수의 데이터 소스를 공급할 때 역할을 합니다. 마지막, 49번째 줄은 GridView에 할당된 각 위젯이 클릭이 가능하도록 설정하는 코드입니다.

참고 : 안드로이드에서 리소스 할당 (R.java)

R.drawable.파일명 ➔ /res/drawable–*dpi의 이미지 리소스

R.layout.파일명 ➔ /res/layout에 있는 레이아웃 XML 파일 접근

R.id.ID명 ➔ 각 레이아웃에서 파일에서 선언된 위젯을 ID를 통해 접근

4) MainActivity의 Inner 클래스인 MainAdapter작성 (97~126번째 줄)

이번 예제코드에서 가장 중요한 부분입니다. MainAdapter는 BaseAdapter를 상속받은 우리만의 Adapter 클래스입니다. MainAdapter는 GridViewd에 데이터 소스를 공급하는 역할을 합니다. MainAdapter는 BaseAdapter를 상속받았으므로, getView(), getCount(), getItem(), getItemId()를 오버라이딩해야 합니다.

MainAdapter에 구현된 getView(), getCount()은 Adapter가 적용된 위젯이 출력될 때 호출됩니다. getItem()과 getItemId()는 반드시 구현되어야 하나, 실제 역할은 없으므로 생략합니다.

getView() 메서드는 GridView가 화면에 아이콘과 텍스트로 구성된 하위 위젯을 그릴 때마다 한번 씩 호출됩니다. MainActivity가 만들어 질 때 onCreate()에서 이미 GridView에 출력할 실제 데이터 이미지와 텍스트를 할당한 바 있습니다. getView()에서는 이 데이터를 이용합니다.

getCount()는 GridView에 출력할 실제 데이터의 개수를 반환합니다. 우리의 예제에서는 6개의 이미지와 텍스트가 있으므로 6을 반환합니다. 여기서 반환된 6은 getView()를 6번 호출하게 하는 역할을 하며 결국 GridView에는 6개의 아이콘과 텍스트가 나타나게 됩니다.

Adapter는 향후 리스트UI 등 다수의 데이터 소스를 요구하는 소스코드에서 항상 사용됩니다. 자주 사용되는 부분이므로 잘 숙지하시기 바랍니다.

5) OnItemClickListener 인터페이스 다중 상속 (18번째 줄)

```
public class MainActivity extends BaseActivity implements OnItemClickListener
```

GridView에 아이콘을 터치할 때 그 터치 이벤트를 처리하기 위해서 OnItemClickListener 인터페이스를 다중 상속합니다. 'implements OnItemClickListener'를 입력하고 'MainActivity ' 위의 커서를 올려 onItemClick()을 추가합니다.

6) onItemClick() 작성 (58~95번째 줄)

onItemClick()은 GridView에서 아이콘 터치 이벤트를 처리해주는 메소드입니다. onItemClick()의 인자 중 int position은 GridView에서 터치 된 아이콘의 위치를 나타냅니다. position값은 좌측 상단이 0이며 하나씩 증가합니다. 우리의 예제에서는 Wheel 아이콘이 0을 반환하며, 터치센서 아이콘이 5를 반환합니다.

onItemClick()은 반환 된 position값에 따라 각각의 예제 Activity로 화면을 전환하게 됩니다. 다른 Activity를 시작하는 것의 코드는 다음과 같습니다.

```
Intent intent = new Intent(this, 클래스명.class);
startActivity(intent);
```

startActivity()에 의해 새로운 Actvitiy로 전환되면 MainActivity는 onPause()함수를 호출하나, 정의되어 있지 않으므로 부모 클래스인 BaseActivity의 onPause()함수를 호출합니다. BaseAcitivity의 onPause()함수를 호출하면 finish()에 의해 Activity가 완전히 종료됩니다 (4장 참조).

```
 1: package com.dongburobot.genieapidemo;
 2:
 3: import java.util.LinkedList;
 4:
 5: import android.content.Intent;
 6: import android.os.Bundle;
 7: import android.view.LayoutInflater;
 8: import android.view.Menu;
 9: import android.view.View;
10: import android.view.ViewGroup;
11: import android.widget.AdapterView;
12: import android.widget.AdapterView.OnItemClickListener;
13: import android.widget.BaseAdapter;
14: import android.widget.GridView;
15: import android.widget.ImageView;
16: import android.widget.TextView;
17:
18: public class MainActivity extends BaseActivity implements OnItemClickListener {
19:     public LinkedList<String> mDemoNameList;
```

```
20:    public LinkedList<Integer> mDemoImgList;
21:
22:    private GridView mMainGrid;
23:    private MainAdapter mMainAdapter;
24:
25:    @Override
26:    public void onCreate(Bundle savedInstanceState) {
27:        super.onCreate(savedInstanceState);
28:        setContentView(R.layout.activity_main);
29:
30:        mDemoNameList = new LinkedList<String>();
31:        mDemoImgList = new LinkedList<Integer>();
32:        mDemoNameList.add("Wheel");
33:        mDemoNameList.add("Motion");
34:        mDemoNameList.add("LED");
35:        mDemoNameList.add("TTS");
36:        mDemoNameList.add("거리센서");
37:        mDemoNameList.add("터치센서");
38:
39:        mDemoImgList.add(R.drawable.icon1);
40:        mDemoImgList.add(R.drawable.icon1);
41:        mDemoImgList.add(R.drawable.icon1);
42:        mDemoImgList.add(R.drawable.icon1);
43:        mDemoImgList.add(R.drawable.icon1);
44:        mDemoImgList.add(R.drawable.icon1);
45:
46:        mMainGrid = (GridView) findViewById(R.id.gridview_apidemos);
47:        mMainAdapter = new MainAdapter();
48:        mMainGrid.setAdapter(mMainAdapter);
49:        mMainGrid.setOnItemClickListener(this);
50:    }
51:
52:    @Override
53:    public boolean onCreateOptionsMenu(Menu menu) {
54:        getMenuInflater().inflate(R.menu.activity_main, menu);
55:        return true;
```

```
56:     }
57:
58:     public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
59:
60:         Intent intent;
61:
62:         switch (position) {
63:         case 0:
64:             // 바퀴(옴니휠) 제어 Activity로 이동
65:             intent = new Intent(this, TestOmniwheelActivity.class);
66:             startActivity(intent);
67:             break;
68:         case 1:
69:             // 모션 제어 Activity로 이동
70:             intent = new Intent(this, TestMotionActivity.class);
71:             startActivity(intent);
72:             break;
73:         case 2:
74:             // 머리 LED 제어 Activity로 이동
75:             intent = new Intent(this, TestLedActivity.class);
76:             startActivity(intent);
77:             break;
78:         case 3:
79:             // TTS Activity 이동
80:             intent = new Intent(this, TestTTSActivity.class);
81:             startActivity(intent);
82:             break;
83:         case 4:
84:             // PSD Activity 이동
85:             intent = new Intent(this, TestPSDSensorActivity.class);
86:             startActivity(intent);
87:             break;
88:         case 5:
89:             // Touch Activity 이동
90:             intent = new Intent(this, TestTouchSensorActivity.class);
91:             startActivity(intent);
```

```
92:         default:
93:             break;
94:         }
95:     }
96:
97:     public class MainAdapter extends BaseAdapter {
98:
99:         public int getCount() {
100:            return mDemoImgList.size();
101:        }
102:
103:        public Object getItem(int position) {
104:            return null;
105:        }
106:
107:        public long getItemId(int position) {
108:            return position;
109:        }
110:
111:        public View getView(int position, View convertView, ViewGroup parent) {
112:            if (convertView == null) {
113:                LayoutInflater li =  (LayoutInflater) getApplicationContext()
114:                        .getSystemService(LAYOUT_INFLATER_SERVICE);
115:                convertView = li.inflate(R.layout.demoitems, null);
116:            }
117:
118:            ImageView icon = (ImageView) convertView.findViewById(R.id.demo_icon);
119:            TextView name = (TextView) convertView.findViewById(R.id.demo_name);
120:
121:            icon.setBackgroundResource(mDemoImgList.get(position));
122:            name.setText(mDemoNameList.get(position));
123:
124:            return convertView;
125:        }
126:    }
127: }
```

[Listing 6-4] MainActivity.java

6.2 구동부 제어

구동부 제어는 Hovis Genie의 옴니휠바퀴 제어하는 과정을 테스트하기 위한 예제입니다. 이번 예제에서는 로봇이 전방 50cm 전진 후, 제자리로 돌아오도록 하는 프로그램을 작성해보겠습니다.

구동부 제어 Activity는 activity_testomniwheel.xml과 TestOmniwheelActivity.java로 구성됩니다. 각각의 파일을 /res/layout과 /src/com/dongburobot/genieapi에 추가합니다.

먼저 구동부 제어 Activitiy의 UI를 구성합니다. activitiy_testomniwheel.xml 파일을 열고 소스 코드를 다음과 같이 수정합니다.

```
1: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2:     xmlns:tools="http://schemas.android.com/tools"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent" >
5:
6:     <!-- 테스트 타이틀 -->
7:     <TextView
8:         android:id="@+id/tv_title"
9:         android:layout_width="match_parent"
10:        android:layout_height="85dp"
11:        android:text="Omniwheel test"
12:        />
13:
14:    <!-- 로봇 이미지 -->
15:    <ImageView
16:        android:id="@+id/iv_image"
17:        android:layout_width="match_parent"
18:        android:layout_height="320dp"
19:        android:layout_below="@id/tv_title"
20:        android:contentDescription="이미지"
21:        />
22:
23:    <!-- 시작과 종료 버튼 -->
24:    <LinearLayout
25:        android:layout_width="match_parent"
26:        android:layout_height="wrap_content"
```

```
27:         android:layout_alignParentBottom="true">
28:         <Button
29:             android:id="@+id/btn_start"
30:             android:layout_width="160dp"
31:             android:layout_height="wrap_content"
32:             android:text="시작"
33:             />
34:         <Button
35:             android:id="@+id/btn_quit"
36:             android:layout_width="160dp"
37:             android:layout_height="wrap_content"
38:             android:text="종료"
39:             />
40:     </LinearLayout>
41:
42: </RelativeLayout>
```

[Listing 6-5] activity_testomniwheel.xml

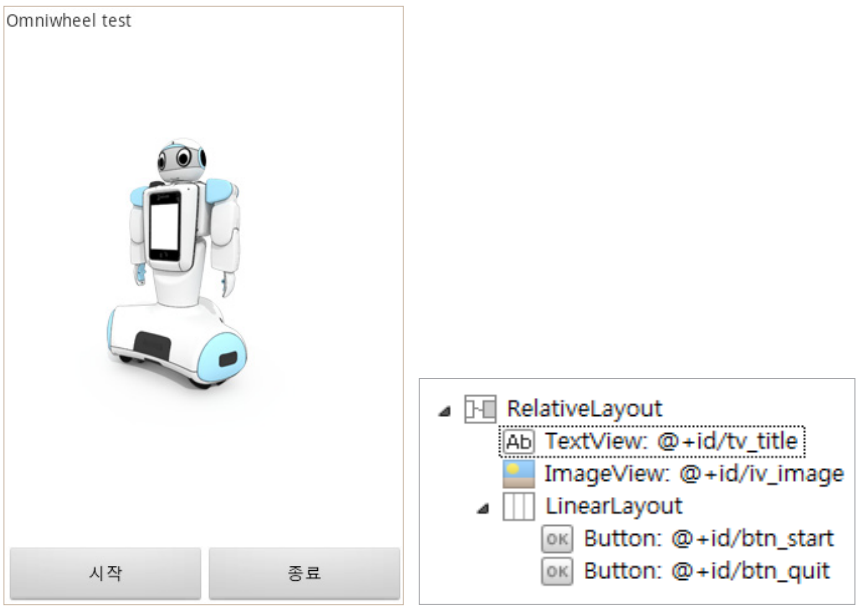
activity_testomniwheel.xml의 구성은 <RelativeLayout>을 최상위 태그로 하여 <TextView>, <ImageView>, <LinearLayout>을 포함합니다. 여기서 <LinearLayout>은 다시 <Button> 위젯 두 개를 포함합니다.

<RelativeLayout>을 사용하면 각 위젯마다 위치를 상대적으로 지정하는 속성을 반드시 기술하여야 합니다. 만약, 상대적인 위치 속성을 지정하지 않게 되면 부모의 좌측상단을 기준으로 위젯이 배치됩니다. 이전 절의 예제의 경우 <RelativeLayout>에 <GridView> 위젯 하나만 존재하였기 때문에 생략을 하였지만, 이번 예제처럼 다수의 위젯이 있는 경우 위치 속성 기술의 생략은 에러의 원인이 됩니다.

<RelativeLayout>의 존재하는 위젯 중 <TextView>에는 위치 속성이 없습니다. 그러므로 좌측상단에 배치가 됩니다. <ImageView>의 위치 속성은 19번째 줄에 있습니다. 이 라인은 <ImageView>를 id가 tv_title인 <TextView> 아래에 배치합니다. <LinearLayout>의 위치 속성은 27번 째줄이며, <LinearLayout>은 <RelativeLayout>의 가장 아래에 배치됩니다.

```
19:        android:layout_below="@id/tv_title"
...
27:        android:layout_alignParentBottom="true"
...
```

이렇게 구성된 UI화면은 최종적으로 다음과 같습니다.



[그림 6-3] 구동부 제어

이이서, 이클립스 Package Explorer의 src 〉 com.dongburobot.genieapi에서 TestOmniwheelActivity.java파일을 선택하고 [Listing 6-6]를 참조하여 소스코드를 추가합니다. 작성방법은 다음과 같습니다. 이번 예제는 wheel_1.png가 필요합니다. 이 파일을 res/drawable-mdpi에 복사합니다. (www.dongburobot.com 자료실 참조)

1) BaseActivity 상속

```
public class TestOmniwheelActivity extends BaseActivity ···
```

2) 위젯 관련 멤버변수 선언

```
private ImageView mIvWheelImg;
private Button mBtnStart;
private Button mBtnQuit;
```

3) onCreate() 작성

onCreate()에서는 레이아웃 XML파일을 연결하고, 각 멤버변수에 위젯 리소스를 설정합니다. 21번째 줄의 setContentView()는 activity_testomniwheel.xml의 정의 된 UI를 현재 Activity의 화면에 표현합니다. 23~25번째 줄은 [그림 6-2]의 로봇 이미지를 설정하는 코드입니다. 27~30번째 줄은 버튼에 터치이벤트를 설정하는 코드입니다.

4) OnClickListener 인터페이스 다중 상속 후 onClick() 작성

이번 예제는 '시작'과 '종료' 두 가지 버튼이 있습니다. 이 버튼을 눌렀을 때, 터치이벤트를 처리하기 위한 소스 코드를 작성합니다. 우선 OnClickListener 인터페이스를 다중상속합니다.

```
public class TestOmniwheelActivity extends BaseActivity implements OnClickListener …
```

'implements OnClickListener'를 추가하고 'TestOmniwheelActivity' 위의 마우스 커서를 올리면 onClick()을 추가할 수 있습니다.

5) moveOmniWheel(), stopOmniWheel() 작성

moveOmniWheel()은 로봇을 50cm 전진하고, 제자리에 돌아오는 함수이며, stopOmniWheel()은 로봇을 중단하는 코드입니다. 상세코드는 다시 설명합니다.

6) onPause() 작성

onPause()는 TestOmniwheelActivity가 화면에서 사라지는 순간 호출되는 함수입니다. 44번째 줄의 super.onPause()는 TestOmniwheelActivity의 부모 클래스인 BaseActivity()의 onPause()함수를 호출합니다. BaseActivity의 onPause()는 finish()를 호출하여 Activity를 완전히 종료시키므로 안드로이드 스택에 Activity를 남기지 않습니다.

7) onKeyDown() 작성

MainActivity도 BaseActivity를 상속받아 작성되었습니다. 그러므로 MainActivity의 GridView에서 아이콘을 클릭하여 다른 예제의 Activity로 전환될 때, MainActivity도 역시 안드로이드 스택에 저장되지 않습니다.
보통 대개의 App은 A -> B Activity로 전환하면서 A Activity가 안드로이드 스택에 저장되기 때문에, Back버튼을 누르면 A Activity로 돌아가게됩니다. 그러나, MainActivity는 안드로이드 메모리 스택에 저장되지 않았기 때문에 Back버튼을 누르면 App이 종료됩니다. 34~40줄의 onKeyDown()은 Back버튼을 누르면 MainActivity로 전환하는 코드입니다. 이 코드로 인해 Back버튼을 누를 때, App이 종료되지 않고 메인화면으로 가게 됩니다.

TestOmniwheelActivity.java의 소스코드는 다음과 같습니다.

```
 1: package com.dongburobot.genieapidemo;
 2:
 3: import android.content.Intent;
 4: import android.os.Bundle;
 5: import android.os.RemoteException;
 6: import android.view.KeyEvent;
 7: import android.view.View;
 8: import android.view.View.OnClickListener;
 9: import android.widget.Button;
10: import android.widget.ImageView;
11:
12: public class TestOmniwheelActivity extends BaseActivity
                                                    implements OnClickListener {
13:
14:     private ImageView mIvWheelImg;
15:     private Button mBtnStart;
16:     private Button mBtnQuit;
17:
18:     @Override
19:     protected void onCreate(Bundle savedInstanceState) {
20:         super.onCreate(savedInstanceState);
21:         setContentView(R.layout.activity_testomniwheel);
22:
23:         mIvWheelImg = (ImageView)findViewById(R.id.iv_image);
24:         mIvWheelImg.setImageResource(R.drawable.wheel_1);
25:         mIvWheelImg.setScaleType(ImageView.ScaleType.FIT_XY);
26:
27:         mBtnStart = (Button)findViewById(R.id.btn_start);
28:         mBtnQuit = (Button)findViewById(R.id.btn_quit);
29:         mBtnStart.setOnClickListener(this);
30:         mBtnQuit.setOnClickListener(this);
31:     }
32:
33:     @Override
34:     public boolean onKeyDown(int keyCode, KeyEvent event) {
35:         if (keyCode == KeyEvent.KEYCODE_BACK) {
```

```
36:             Intent intent = new Intent(this, MainActivity.class);
37:             startActivity(intent);
38:         }
39:         return super.onKeyDown(keyCode, event);
40:     }
41:
42:     @Override
43:     protected void onPause() {
44:         super.onPause();
45:     }
46:
47:     public void onClick(View v) {
48:         switch (v.getId()) {
49:         case R.id.btn_start:
50:             moveOmniwheel();
51:             break;
52:         case R.id.btn_quit:
53:             stopOmniwheel();
54:             break;
55:         default:
56:             break;
57:         }
58:     }
59:
60:     private void moveOmniwheel() {
61:         if (myRobotApp.mBinder == null)
62:             return;
63:
64:         try {
65:           myRobotApp.mBinder.dmel_robot_set_pose(getComponentName(), 0, 0, 0);
66:             myRobotApp.mBinder.dmel_navigate_prepare(getComponentName());
67:             myRobotApp.mBinder.
                dmel_navigate_add_goal_pose(getComponentName(), 0.5, 0, 0, true, 1);
68:             myRobotApp.mBinder.
                dmel_navigate_add_goal_pose(getComponentName(), 0, 0, 0, true, 1);
69:             myRobotApp.mBinder.dmel_navigate_start(getComponentName());
```

```
70:         } catch (RemoteException e) {
71:             e.printStackTrace();
72:         }
73:     }
74:
75:     private void stopOmniwheel() {
76:         if (myRobotApp.mBinder == null)
77:             return;
78:
79:         try {
80:             myRobotApp.mBinder.dmel_navigate_stop(getComponentName());
81:         } catch (RemoteException e) {
82:             e.printStackTrace();
83:         }
84:     }
85: }
```

[Listing 6-6] TestOmniwheelActivity.java

이번 예제에서 가장 중요한 부분은 시작과 종료버튼을 터치할 때 호출되는 moveOmni wheel()과 stopOmniwheel() 함수입니다. 이 두 함수는 로봇 서비스를 이용하여 구동부를 제어합니다. moveOmniwheel()은 로봇을 50cm 전진시키고, 제자리로 돌아오도록 동작시키며, stopOmniwheel()은 로봇의 구동을 멈추게 합니다.

moveOmniwheel()은 로봇 서비스를 이용합니다. 로봇 서비스는 GenieApiDemo App이 실행될 때 바인딩됩니다. 바인딩 된 서비스는 IHovisGenieService의 객체인 mBinder를 통해 로봇 API를 호출할 수 있습니다. mBinder는 다음과 같이 GenieApiApplication에 선언되어 있습니다.

GenieApiDemoApplication.java 14번째 줄:

```
public IHovisGenieService mBinder = null;
```

GenieApiDemoApplication 클래스에 정의된 mBinder에 접근하기 위해서는GenieApi-Application의 객체가 필요합니다. 이 객체는 BaseActivity에 선언되어 있습니다.

BaseActivity.java 10번째 줄:

```
protected GenieApiDemoApplication myRobotApp;
```

현재 TestOmniwheelActivity 클래스는 BaseActivity의 자식 클래스이므로 myRobotApp을 선언하지 않아도 사용할 수 있습니다. 그래서 최종적으로 로봇의 서비스는 myRobotApp.mBinder를 사용하여 접근할 수 있습니다.

moveOmniwheel()의 61번째 줄은 myRobotApp.mBinder의 객체가 null인지 검사합니다. 로봇 서비스는 외부에서 동작하기 때문에 비록 App을 시작할 때 서비스에 바인딩 하였더라도, 중간에 바인딩이 끊길 수 있습니다. 로봇 서비스가 바인딩이 끊긴 상태라면 로봇 제어 시 오류가 발생하므로, 이 코드를 통해 에러를 사전에 방지합니다.

64~72번째 줄은 로봇 서비스를 이용하는 코드입니다. 로봇 서비스는 원격의 프로세스와 상호간의 통신을 하는 구조이기 때문에 RemoteException이 발생할 수 있습니다. 로봇 서비스는 예외처리를 위해 반드시 try-catch문을 써주어야 합니다.

try-catch문으로 감싸진 부분에는 다음과 같이 실제 로봇 서비스의 API함수를 호출하는 부분이 있습니다. 각 로봇 서비스 함수는 다음과 같으며, 한 줄씩 살펴보도록 하겠습니다.

dmel_robot_set_pose

```
boolean dmel_robot_set_pose(ComponentName cn,
                            double x,
                            double y,
                            double theta)
                           throws android.os.RemoteException
```

로봇의 현재 위치를 x, ,y, theta로 설정한다. 로봇의 현재 위치를 (0, 0, 0)으로 설정하면 현 위치가 기준점이 된다.

Parameters:
cn – Component 이름
x – (meter)
y – (meter)
theta – (Radian –PI ~ +PI)

Returns:
성공시 true, 실패시 false 반환

Throws:
android.os.RemoteException

dmel_navigate_prepare

void dmel_navigate_prepare(ComponentName cn)
throws android.os.RemoteException
로봇 네비게이션을 위한 초기화를 수행한다. 초기화시 로봇의 현 위치를 (0, 0, 0)으로 설정하며, 현재 Heading을 0도로 설정한다.
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

dmel_navigate_add_goal_pose

boolean dmel_navigate_add_goal_pose(ComponentName cn,
double x,
double y,
double theta,
boolean rflag,
int time)
throws android.os.RemoteException
로봇이 방문할 지점을 추가한다. 로봇에 설정된 시작점 (0, 0, 0)을 기준으로 방문할 지점을 결정한다. (dmel_navigate_prepare() 참조) 로봇은 설정된 방문 지점을 dmel_navigate_start()를 통해 방문을 시작한다.
Parameters:
cn – Component 이름
x – 시작점 기준 x (meter)
y – 시작점 기준 y (meter)
theta – 시작점에서의 로봇 Heading 기준 로봇 방향 theta (Radian –PI ~ +PI)
rflag – true시 로봇이 방문지점 도착시 theta도 방향을 바라봄
time – 로봇이 방문지점 도착시의 대기시간 (sec)
Returns:
Throws:
android.os.RemoteException

dmel_navigate_start

boolean dmel_navigate_start(ComponentName cn)
throws android.os.RemoteException
로봇 네비게이션 수행한다. 로봇 네비게이션은 미리 설정된 goal_pose를 한 지점 씩 방문한다.
Parameters:
cn – Component 이름
Returns:
성공시 true, 실패시 false 반환
Throws:
android.os.RemoteException

dmel_robot_set_pose()함수는 로봇의 현재 위치를 x, y, theta로 설정합니다. 여기서는 0, 0, 0으로 설정했으므로 현재 로봇이 놓인 위치를 좌표평면상 x좌표 0, y좌표 0, 그리고 로봇의 바라보는 방향을 0도로 설정하게 됩니다. 인자 값의 단위는 x, y가 m이며, theta값은 radian입니다.

dmel_navigate_prepare()는 로봇 네비게이션을 위한 초기화를 수행하는 함수입니다. 그 기능은 dmel_robot_set_pose()와 동일합니다. (위의 예제에서 두 라인 중 하나는 생략가능) dmel_navigate_add_goal_pose()는 로봇의 이동 포인트를 지정하는 역할을 합니다. dmel_navigate_add_goal_pose()는 이동지점 정보를 추가하기만 하고, 실제 로봇의 구동은 dmel_navigate_start() 함수가 호출될 때 시작합니다.

첫번째 dmel_navigate_add_goal_pose()의 인자는 ComponentName인 첫 번째 인자를 제외하고 x미터, y미터, theta(radian)를 의미합니다. 코드에서는 0.5, 0, 0은 50cm 전진한 지점에서 로봇이 움직이기 전 바라보고 있던 방향을 뜻합니다. 이 함수의 다섯 번째 인자는 로봇이 도착 시 세 번째 인자에서 지정한 theta방향을 바라보게 할 건지의 대한 true, false를 지정하고, 여섯 번째 인자는 로봇이 도착 후, 도착 지점에서의 대기시간(sec)을 뜻합니다. 이 코드에서는 true, 1 이므로 로봇이 도착 후 0도를 바라보고 1초 대기합니다.

두 번째 dmel_navigate_add_goal_pose()의 인자는 0, 0, 0, true, 1입니다. 앞의 세 개의 인자는 제자리를 뜻합니다. 네번째 인자가 true이므로 로봇이 제자리로 돌아오면 다시 정방향을 바라보도록 회전을 하며, 다섯번째 인자에 의해 도착 후 1초 대기합니다.

이후, dmel_navigate_start()함수가 호출되면, 실제 로봇이 구동합니다.

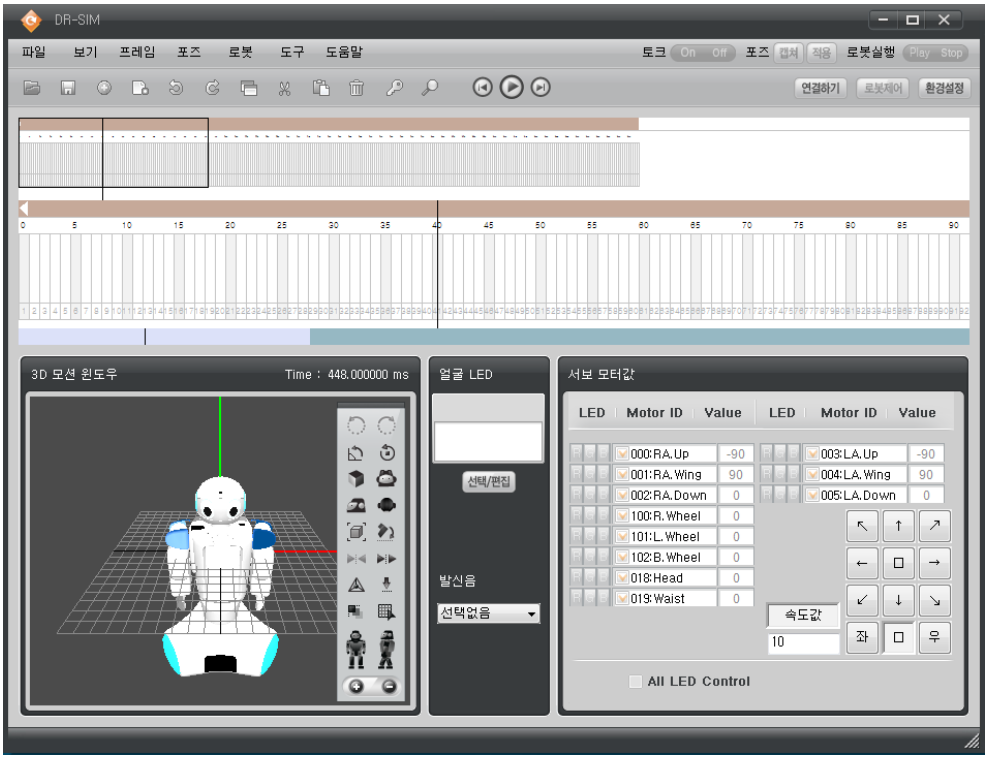
```
myRobotApp.mBinder.dmel_robot_set_pose(getComponentName(), 0, 0, 0);
myRobotApp.mBinder.dmel_navigate_prepare(getComponentName());
myRobotApp.mBinder.dmel_navigate_add_goal_pose(getComponentName(), 0.5, 0, 0, true, 1);
myRobotApp.mBinder.dmel_navigate_add_goal_pose(getComponentName(), 0, 0, 0, true, 1);
myRobotApp.mBinder.dmel_navigate_start(getComponentName());
```

6.3 모션 제어

이번 예제는 Hovis Genie에서 모션을 동작하는 예제입니다. 모션 제어란 로봇의 구동부 및 관절을 구성하는 모터를 동작시킴으로써 로봇 움직임을 제어하는 것을 말합니다. Hovis Genie는 총 11개의 모터를 탑재하고 있습니다. 각 팔에 3개의 모터가 있으며, 허리 모터, 머리 모터, 그리고 3개의 옴니휠 구동부 모터가 있습니다.

모션 제어를 하기 위해 각각의 모터 제어가 필요합니다. 그러나 모터를 하나씩 제어한다면 제어가 어려워질 뿐만 아니라 효율적으로 모션을 제작하기가 어렵습니다. 그러한 이유로, 동부로봇에서는 로봇 모션 에디터 툴인 DR-SIM을 제공합니다. 유저는 DR-SIM을 이용해서 Hovis Genie의 모션을 편집할 수 있습니다.

http://www.dongburobot.com/jsp/cms/view.jsp?code=100122&isSkin=Y&cmd=view&boardCode=100074&bseq=3703



[그림 6-4] DR-SIM 모션 에디터 프로그램

DR-SIM은 가상으로 표현 된 3D 로봇을 기반으로 모션을 제작하며, 타임라인을 제공하여 시간의 흐름에 따라 모션을 편집할 수 있는 프로그램입니다. DR-SIM을 통해서 만든 모션 파일은 Hovis Genie 모션 제어에 사용되며, 확장자는 DMT입니다. (DR-SIM 사용법은 타 매뉴얼 참조)

Hovis Genie의 모션파일은 MID의 SD 카드에 탑재되며, 로봇 모션 파일의 위치는 /dongbu/motion입니다.

로봇 모션 폴더의 접근은 MID를 USB 메모리 장치를 사용함으로써 가능합니다. MID상에서 설정 〉 애플리케이션 〉 개발 〉 USB 디버깅 항목의 체크를 해제하고 PC와 연결하면 MID의 SD Card를 열어 볼 수 있습니다. 그리고 /dongbu/motion으로 접근합니다.

MID의 모션 폴더에는 이미 다양한 모션이 탑재되어 있습니다. 이 모션 파일들은 기본 모션이며 리스트는 다음과 같습니다.

파일명	내 용
01m_turn_l.dmt	제자리에서 몸 회전 (우)
01m_turn_r.dmt	제자리에서 몸 회전 (좌)
01n_adbomen0.dmt	한 손으로 배 만지기
01n_adbomen1.dmt	양 손으로 배 만지기
01n_aging.dmt	모터 Aging
01n_bend_weist.dmt	허리 숙이기
01n_body.dmt	몸 닦기
01n_clap.dmt	가슴 앞에서 박수
01n_command1.dmt	지휘하기1
01n_command2.dmt	지휘하기2
01n_command3.dmt	지휘하기3
01n_command4.dmt	지휘하기4
01n_dance.dmt	춤추기
01n_face.dmt	세수
01n_front_hand.dmt	앞으로 나란히하고 머리 두리번
01n_goodjob.dmt	한 팔 앞으로 내밀기 (최고야!)

파일명	내 용
01n_gymnastics.dmt	국민체조
01n_hide.dmt	양팔로 얼굴 가리기
01n_hurrah0.dmt	돌아다니면서 만세
01n_hurrah1.dmt	만세한 채로 양 팔 좌우로 흔들기 : 다른 방향
01n_hurrah2.dmt	만세한 채로 양 팔 좌우로 흔들기 : 같은 방향
01n_kiss0.dmt	뽀뽀1 (한 손 입에 붙였다 떼기)
01n_kiss1.dmt	뽀뽀2 (양 손 입에 붙였다 떼기)
01n_love.dmt	제자리 사랑해 (양 팔 위로 올려 하트[원] 만들기)
01n_muscle	근육 자랑
01n_pos1	포즈 1
01n_scrape.dmt	눈 비비기(눈 앞에 한 손 좌우 움직이기)
01n_scratch	머리 긁기
01n_self_msg.dmt	스스로 안마하기
01n_shake_head0.dmt	한 팔 위로 들고 흔들기
01n_shake_head1.dmt	양팔 벌리고 흔들기 (팔 올리지 않고)
01n_shake_head2.dmt	양 팔 번갈아 앞뒤로 흔들며 허우적 거리기
01n_shake_head.dmt	고개 빠르게 흔들기 (절레절레)
01n_sidebyside.dmt	한 팔 앞으로 나란히
01n_sorrow.dmt	울기 (눈 앞에 양손 좌우 움직이기)
01n_stare0.dmt	제자리에서 두리번거리기
01n_stare1.dmt	한 손 머리에 대고 두리번거리기
01n_stare2.dmt	한 손 귀 옆에 대고 두리번거리기
01n_stretching.dmt	기지개
01n_tekwuando.dmt	태권도
01n_tekwuando_wide.dmt	태권도 (MID 가로시)

파일명	내 용
01n_tooth.dmt	양치질
01n_turn_360_0.dmt	제자리에서 360도 회전 (1회)
01n_turn_360_1.dmt	제자리에서 360도 회전 (반복)
01n_turn_head_l.dmt	고개 돌리기 (우)
01n_turn_head_r.dmt	고개 돌리기 (좌)
01n_turn_love.dmt	한바퀴 회전 사랑해 (양 팔 위로 올려 하트[원] 만들기)
01n_turn_shake.dmt	가슴 앞에 양팔 접고 바퀴 좌우 흔들기 (몸 흔드는 것처럼)
01n_up_hand.dmt	한 팔 위로 올리기
01n_watch_clock.dmt	손목 시계 보는 듯한 모션 (팔꿈치 접어 얼굴 가까이로)
01n_wave.dmt	양팔 웨이브
basicpose.dmt	기본 자세
mid_rotate.dmt	MID가 가로일 때 세로로 만들기
motion_highwalk.dmt	빨리 걷기
motion_slow_walk.dmt	천천히 걷기
motion_standby.dmt	준비 자세
motion_walk.dmt	보통 걷기

이번 예제는 미리 탑재 된 기본 모션을 활용하여 로봇 모션을 실행해보도록 하겠습니다. 모션 제어는 activity_testmotion.xml과 TestMotionActivity.java로 구성되어 있으므로, 이 두 개의 파일을 GenieApiDemo 프로젝트에 추가합니다. 그리고 activity_testmotion.xml을 다음과 같이 수정하여 모션 제어에 대한 UI를 구성합니다.

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent" >
5:
6:     <!— 테스트 타이틀 —>
7:     <TextView
```

```
 8:          android:id="@+id/tv_title"
 9:          android:layout_width="match_parent"
10:          android:layout_height="85dp"
11:         android:text="Motion test"
12:          />
13:
14:     <!— 모션 리스트 —>
15:     <ListView
16:          android:id="@+id/lv_motions"
17:          android:layout_width="match_parent"
18:          android:layout_height="wrap_content"
19:          android:layout_below="@id/tv_title"
20:          />
21:
22: </RelativeLayout>
```

[Listing 6–7] activity_testmotion.xml

activity_testmotion.xml의 최상위 〈RelativeLayout〉은 테스트 타이틀을 표현하는 〈TextView〉와 모션 리스트를 표현하기 위한 〈ListView〉를 포함하고 있습니다. 이에 대한 UI 구성을 화면에서 표현하면 [그림 6–5]와 같습니다.

[그림 6–5] 모션 제어

TestMotionAcitivity.java에서 ListView 항목설정의 핵심은 Adapter입니다. 6.1절에서 메인화면의 GridView를 구성시 다양한 데이터 소스를 받아들이기 위해서 Adapter를 사용하였듯이, 여기에서도 Adapter를 사용합니다.
다만, 6.1절은 이미지와 텍스트로 구성 된 다수의 데이터 소스를 받아들이기 위해 demoitems.xml에 하위 위젯을 정의하고 MainAdapter라는 클래스를 작성하였지만, 여기에서는 'Walk', 'tekwuando' 등 하나의 TextView만을 사용하므로 안드로이a가 제공하는 기본 ArrayAdapter를 사용합니다. 다만, 하나의 TextView로 구성 된 하위 위젯의 정의는 필요하므로 /res/layout에 listitems.xml을 추가하고 소스를 추가합니다.

```
1: 〈?xml version="1.0" encoding="utf-8"?〉
2: 〈LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent"
5:     android:orientation="vertical" 〉
6:
7:     〈TextView
8:         android:id="@+id/tv_name"
9:         android:layout_width="wrap_content"
10:        android:layout_height="40dp"
11:        android:gravity="center_vertical"
12:        /〉
13:
14: 〈/LinearLayout〉
```

[Listing 6-7] listitems.xml

listitems.xml은 가로가 〈TextView〉에 할당되는 문자열의 크기이고, 세로가 40dp인 TextView를 나타내며, [그림 6-4]의 'Walk'와 'tekwuando'에 해당하는 위젯입니다.

안드로이드 크기 단위
px – 픽셀(pixel)
dp – deviece-independent pixel

안드로이드에서 크기를 나타내는 여러 단위가 있으나, 주로 dp(dpi)를 사용됩니다. 그 이유는 px단위로 지정된 위젯은 기기의 해상도 차이에 따라 화면에 다르게 나타날 수 있기 때문입니다. dp를 단위로 지정하면 어떤 화면에서나 똑같은 비율의 크기로 보이므로, dp를 사용하는 것이 좋습니다.

마지막으로 TestMotionActivity.java를 [Listing 6-8]을 참조하여 다음과 같은 순서로 작성합니다.

1) BaseActivity 상속

```
public class TestMotionActivity extends BaseActivity …
```

2) 위젯 관련 멤버변수 선언

```
private ListView mMotionListView;
```

3) onCreate() 작성 – ListView와 Adapter

TestMotionActivity가 처음 화면에 나타날 때, 실행되는 코드입니다. 24번째 줄의 setContentView() 함수는 activity_testmotion.xml에 정의 된 UI를 불러옵니다. 26번째 줄에서는 ListView ID를 통해 멤버변수 mMotionListView에 리소스를 할당합니다.

28~30줄은 ArrayList에 문자열을 할당합니다. 32~35줄에서는 ArrayAdapter의 객체 adapter를 생성하는데 이 부분에서 가장 핵심 코드입니다. ArrayAdapter의 생성자와 해당 소스코드는 다음과 같습니다.

```
ArrayAdapter(Context context, int resource, int textViewResourceId, List<T> objects)

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                                        R.layout.listitems, R.id.tv_name, arrayList);
```

위 코드의 두 번째 인자는 조금 전에 선언한 listitems.xml의 리소스이며, 세 번째 인자는 listitems.xml에 선언 된 TextView의 ID입니다. 그리고 마지막 네 번째 인자는 28~30줄에서 할당한 리스트 문자열입니다. 이런 인자 값들을 통하여 adapter는 ListView에 공급할 데이터의 형태와 값을 유지합니다. 마지막으로, 36번째 줄에서 ListView에 이 adapter를 붙여줌으로써 ListView에 데이터 할당을 하게 됩니다. 37번째 줄은 ListView에 터치 이벤트를 등록합니다.

4) onItemClickListener 인터페이스 다중상속 후 onItemClick() 작성

유저의 터치 이벤트를 처리하는 함수입니다. ListView의 위치는 0부터 카운트 되므로, 이 예제에서 'Walk'는 0이고, 'tekwuando'는 1입니다.

```
1: package com.dongburobot.genieapidemo;
2:
3: import java.util.ArrayList;
4:
5: import android.content.Intent;
6: import android.os.Bundle;
7: import android.os.RemoteException;
8: import android.util.Log;
9: import android.view.KeyEvent;
10: import android.view.View;
11: import android.widget.AdapterView;
12: import android.widget.AdapterView.OnItemClickListener;
13: import android.widget.ArrayAdapter;
14: import android.widget.ListView;
15:
16: public class TestMotionActivity extends BaseActivity
                                                implements OnItemClickListener {
17:
18:     private ListView mMotionListView;
19:
20:     @Override
21:     protected void onCreate(Bundle savedInstanceState) {
22:         // TODO Auto-generated method stub
23:         super.onCreate(savedInstanceState);
24:         setContentView(R.layout.activity_testmotion);
```

```
25:
26:         mMotionListView = (ListView)findViewById(R.id.lv_motions);
27:
28:         ArrayList<String> arrayList = new ArrayList<String>();
29:         arrayList.add("Walk");
30:         arrayList.add("tekwuando");
31:
32:         ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
33:                 R.layout.listitems,
34:                 R.id.tv_name,
35:                 arrayList);
36:         mMotionListView.setAdapter(adapter);
37:         mMotionListView.setOnItemClickListener(this);
38:     }
39:
40:     @Override
41:     public boolean onKeyDown(int keyCode, KeyEvent event) {
42:         if (keyCode == KeyEvent.KEYCODE_BACK) {
43:             Intent intent = new Intent(this, MainActivity.class);
44:             startActivity(intent);
45:         }
46:         return super.onKeyDown(keyCode, event);
47:     }
48:
49:     public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
50:         if (myRobotApp.mBinder == null)
51:             return;
52:
53:         try {
54:             myRobotApp.mBinder.dmel_motion_stop(getComponentName());
55:
56:             switch (position) {
57:             case 0:
58:                 myRobotApp.mBinder.
                    dmel_motion_play(getComponentName(), "motion_walk.dmt");
```

```
59:                 Log.i("motion", "0 clicked");
60:                 break;
61:             case 1:
62:                 myRobotApp.mBinder.
                    dmel_motion_play(getComponentName(), "01n_tekwuando_wide.dmt");
63:                 Log.i("motion", "1 clicked");
64:             default:
65:                 break;
66:             }
67:
68:             Thread.sleep(1000);
69:         } catch (RemoteException e) {
70:             e.printStackTrace();
71:         } catch (InterruptedException e) {
72:             e.printStackTrace();
73:         }
74:     }
75: }
```

[Listing 6–7] listitems.xml

onItemClick()은 유저가 리스트에서 항목을 터치할 때 수행됩니다. 리스트의 항목이 터치되면 50번째 줄에서 myRobotApp.mBinder를 검사하여 서비스가 이상 없이 연결되어 있는 지 확인합니다. 이상이 없다면 현재 수행되고 있는 모션을 중지하는 명령을 내린 후, 모션을 시작합니다.

```
myRobotApp.mBinder.dmel_motion_stop(getComponentName());
...
myRobotApp.mBinder.dmel_motion_play(getComponentName(), 파일명);
```

dmel_motion_play

void dmel_motion_play(ComponentName cn,
java.lang.String motion)
throws android.os.RemoteException
로봇 모션 파일을 재생한다. (기본경로는 /sdcard/dongbu/motion/ 으로 지정되어 있음)
Parameters:
cn – Component 이름
motion – 모션파일이름 ex) “01n_shake_head.dmt”
Throws:
android.os.RemoteException

dmel_motion_stop

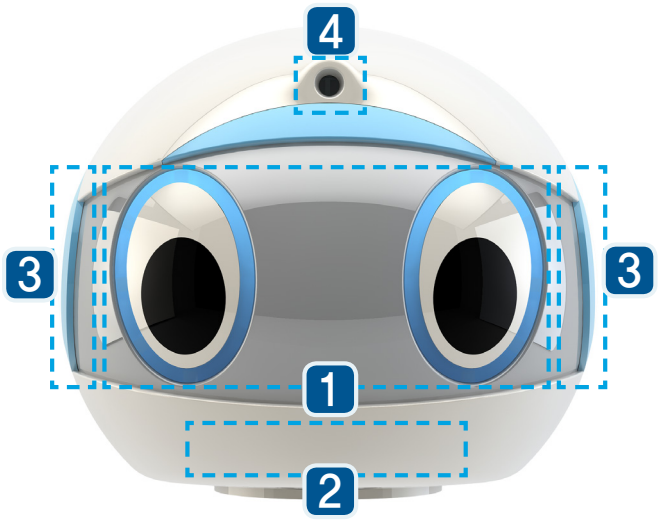
void dmel_motion_stop(ComponentName cn)
throws android.os.RemoteException
현재 실행중인 모션을 중단한다
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

예제 56~66번째 줄에서는 리스트 상의 터치 된 위치에 따라 모션을 수행해 주는 코드입니다. 이 코드에 따라 ‘Walk’선택 시 motion_walk.dmt, ‘tekwuando’ 선택 시 01n_tekwuando_wide.dmt 모션 파일을 재생합니다.

여기서 주의할 점이 모션파일이 반드시 SD 카드에 /dongbu/motion에 존재하여야 합니다. 여러분은 추가적으로 파일 존재 여부에 대한 예외처리도 해보시길 바랍니다. 또한 다른 기본 모션을 리스트에서 늘려가면서 연습해 보시고, DR-SIM으로 직접 작성한 모션도 제어해보십시오.

6.4 머리 LED 제어

이번 절에서는 Hovis Genie의 머리 LED 제어 대해서 알아보겠습니다. Hovis Genie의 머리에는 LED를 제어할 수 있는 보드가 있어, 눈, 입, 귀, 그리고 이마의 LED를 키고 끌 수 있습니다. 구체적으로 LED의 구성은 다음 그림과 같습니다.



[그림 6–6] Hovis Genie 머리 LED 배치 (1–눈, 2–입, 3–귀, 4–이마)

1번 위치는 눈 LED입니다. 한쪽 눈에는 빨간색과 파란색 LED가 각 8개씩 배치되어 16개의 LED를 가지며, 양쪽 눈에는 총 32개의 LED가 달려있습니다. 눈 LED의 경우 파란색, 빨간색을 각각 설정할 수 있으며, 파란색과 빨간색을 동시에 켜서 보라색 효과를 낼 수 있습니다.

2번 위치는 입 LED입니다. 입 LED는 총 3개로 구성되어 있습니다. 3번은 귀 LED이며 양쪽에 있습니다. 귀의 경우 한쪽 귀에 각 4개씩 LED를 가지나, 모두 키거나 끄는 등 일괄적인 제어만 가능합니다. 4번 위치는 이마의 휘도 LED로 밝기의 강도를 조절할 수 있는 LED입니다. 참고로, 입과 귀 그리고 이마의 LED는 단색입니다.

로봇 서비스는 머리 LED 제어를 효율적으로 하기 위해 각각의 LED 제어가 아닌 모듈 별 효과 위주의 함수를 제공합니다. 이 함수들은 LED를 통해 로봇의 감정 표현 및 시각 효과에 유용하게 사용됩니다.

머리 LED 제어에 필요한 activity_testled.xml과 TestLedActivity.java를 GenieApiDemo 프로젝트에 추가하고, activity_testled.xml을 열어 예제를 위한 UI를 구성합니다.

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent" >
5:
6:     <!-- 테스트 타이틀 -->
7:     <TextView
8:         android:id="@+id/tv_title"
9:         android:layout_width="match_parent"
10:        android:layout_height="85dp"
11:        android:text="LED test"
12:        />
13:
14:    <!-- LED 제어 리스트 -->
15:    <ListView
16:        android:id="@+id/lv_ledcontrols"
17:        android:layout_width="match_parent"
18:        android:layout_height="wrap_content"
19:        android:layout_below="@id/tv_title"
20:        />
21:
22: </RelativeLayout>
```

[Listing 6-7] activity_testmotion.xml

머리 LED 제어의 UI는 이전 6.3절의 모션 제어와 거의 유사하며, 화면은 [그림 6-7]와 같습니다.

[그림 6–7] 머리 LED 제어

이번 예제도 마찬가지로 ListView에 한 항목이 될 하위 위젯을 작성하고, onCreate()에서 문자열 데이터를 추가하여 Adapter를 만듭니다. 그리고 Adapter를 ListView에 등록함으로 써 화면에 나타나게 합니다. 최종적으로 ListView에 터치 이벤트를 설정하고 onItem–ClickListener 인터페이스를 상속받아 onItemClick()을 작성합니다.

ListView의 하위 위젯을 지정하는 listitems.xml은 6.3절에서 이미 작성했으므로 생략합니다. TestLedActivity.java의 작성법 역시 유사하므로, 6.3절을 참고합니다.

```
 1: package com.dongburobot.genieapidemo;
 2:
 3: import java.util.ArrayList;
 4:
 5: import android.content.Intent;
 6: import android.os.Bundle;
 7: import android.os.RemoteException;
 8: import android.view.KeyEvent;
 9: import android.view.View;
10: import android.widget.AdapterView;
11: import android.widget.AdapterView.OnItemClickListener;
12: import android.widget.ArrayAdapter;
13: import android.widget.ListView;
```

```
14:
15: public class TestLedActivity extends BaseActivity implements OnItemClickListener {
16:
17:     private ListView mLedControlListView;
18:
19:     @Override
20:     protected void onCreate(Bundle savedInstanceState) {
21:         // TODO Auto-generated method stub
22:         super.onCreate(savedInstanceState);
23:         setContentView(R.layout.activity_testled);
24:
25:         mLedControlListView = (ListView)findViewById(R.id.lv_ledcontrols);
26:
27:         ArrayList<String> arrayList = new ArrayList<String>();
28:         arrayList.add("모두끄기");
29:         arrayList.add("모두켜기");
30:
31:         ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
32:                 R.layout.listitems,
33:                 R.id.tv_name,
34:                 arrayList);
35:         mLedControlListView.setAdapter(adapter);
36:         mLedControlListView.setOnItemClickListener(this);
37:
38:     }
39:
40:     @Override
41:     public boolean onKeyDown(int keyCode, KeyEvent event) {
42:         if (keyCode == KeyEvent.KEYCODE_BACK) {
43:             Intent intent = new Intent(this, MainActivity.class);
44:             startActivity(intent);
45:         }
46:         return super.onKeyDown(keyCode, event);
47:     }
48:
49:     public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
50:         turnAllLedOff();
```

```
51:         try {
52:             Thread.sleep(300);
53:         } catch (InterruptedException e) {
54:             e.printStackTrace();
55:         }
56:
57:         switch (position) {
58:         case 0:
59:             turnAllLedOff();
60:             break;
61:         case 1:
62:             turnAllLedOn();
63:             break;
64:         default:
65:             break;
66:         }
67:
68:     }
69:
70:
71:     // beam 0 : 끄기, 1 : 약하게 키기, 2 : 중간 키기, 3 : 밝게 키기
72:     // eyeCode – 0 : 기존상태유지, 1 : 빙글빙글, 2 : 눈썹깜박, 3 : 눈썹정지,
73:     //              4 : 전체 켜기, 5 : 전체 깜박, 6 : 좌우 효과, 7 : 상하 효과, 8 : 끄기

74:     // earCode – 0 : 기존상태유지, 1 : 켜기, 2 : 끄기, 3 : 깜박
75:     // colorCode – 1 : RED LED, 2 : BLUE LED, 3 : PURPLE (RED + BLUE) LED
76:     // mouthCode – 0 : 기존상태유지, 1 : 가운데 ON, 2 : 세개 ON, 3 : 가운데 깜박,
                                                4 : 세개다 깜박, 5 : 말하기, 6: 끄기
77:     private void turnAllLedOff() {
78:         if (myRobotApp.mBinder == null)
79:             return;
80:
81:         try {
82:             myRobotApp.mBinder.dmel_hri_set_mouth_led(getComponentName(), 6);
83:             myRobotApp.mBinder.dmel_hri_set_ear_led(getComponentName(), 2);
84:             myRobotApp.mBinder.dmel_hri_set_brow_beam(getComponentName(), 0);
```

```
85:                 myRobotApp.mBinder.dmel_hri_set_eye_led(getComponentName(), 8, 1);
86:             } catch (RemoteException e) {
87:                 e.printStackTrace();
88:             }
89:
90:         }
91:
92:         private void turnAllLedOn() {
93:             if (myRobotApp.mBinder == null)
94:                 return;
95:
96:             try {
97:                 myRobotApp.mBinder.dmel_hri_set_mouth_led(getComponentName(), 2);
98:                 myRobotApp.mBinder.dmel_hri_set_brow_beam(getComponentName(), 2);
99:                 myRobotApp.mBinder.dmel_hri_set_ear_led(getComponentName(), 1);
100:                myRobotApp.mBinder.dmel_hri_set_eye_led(getComponentName(), 4, 2);
101:            } catch (RemoteException e) {
102:                e.printStackTrace();
103:            }
104:        }
105: }
```

[Listing 6-10] TestLedActivity.java

이번 예제의 ListView 항목은 '모두끄기' 그리고 '모두켜기'가 있습니다. 각각의 터치는 onItemClick()의 57~65번째 줄에서 turnAllLedOff()와 turnAllLedOn()을 호출하여 머리 LED를 모두 키거나 끄게 됩니다.
turnAllLedOff()와 turnAllLedOn()은 다음 네 가지의 로봇 서비스 함수를 호출합니다. 다만, 인자 값에 따라 LED 동작 차이가 있습니다. 각각의 함수는 다음과 같습니다.

dmel_hri_set_brow_beam

```
void dmel_hri_set_brow_beam(ComponentName cn,
                            int brightness)
                            throws android.os.RemoteException
```

이마 LED를 설정한다.
Parameters:
cn – Component 이름
brightness – 0 : 끄기, 1 : 약하게 키기, 2 : 중간 키기, 3 : 밝게 키기
Throws:
android.os.RemoteException

dmel_hri_set_eye_led

```
void dmel_hri_set_eye_led(ComponentName cn,
                          int eyeCode,
                          int colorCode)
                          throws android.os.RemoteException
```

눈 LED를 설정한다.
Parameters:
cn – Component 이름
eyeCode – 0 : 기존상태유지, 1 : 빙글빙글, 2 : 눈썹깜박, 3 : 눈썹정지, 4 : 전체 켜기, 5 : 전체 깜박, 6 : 좌우 효과, 7 : 상하 효과, 8 : 끄기
colorCode – 1 : RED LED, 2 : BLUE LED, 3 : PURPLE (RED + BLUE) LED
Throws:
android.os.RemoteException

dmel_hri_set_ear_led

```
void dmel_hri_set_ear_led(ComponentName cn,
                          int earCode)
                          throws android.os.RemoteException
```

귀 LED를 설정한다.
Parameters:
cn – Component 이름
earCode – 0 : 기존상태유지, 1 : 켜기, 2 : 끄기, 3 : 깜박
Throws:
android.os.RemoteException

dmel_hri_set_mouth_led

```
void dmel_hri_set_mouth_led(ComponentName cn,
                            int mouthCode)
                            throws android.os.RemoteException
```

입 LED를 설정한다.
Parameters:
cn – Component 이름
mouthCode – 0 : 기존상태유지, 1 : 가운데 ON, 2 : 세개 ON, 3 : 가운데 깜박, 4 : 세개다 깜박, 5 : 말하기
Throws:
android.os.RemoteException

입, 이마, 귀 LED는 단색으로 로봇 서비스 함수에 색깔을 지정하는 인자가 없지만, 눈 LED 제어 함수인 dmel_hri_set_eye_led()에만 눈 색깔 값을 지정하는 인자가 있습니다. 앞서 말했듯이 눈은 한쪽당 8개씩의 빨간색, 파란색 LED가 있기 때문입니다. (보라색은 빨간색과 파란색 LED가 모두 켜져 색이 합성되어 나타나는 색입니다.) 눈 LED의 경우 32개의 LED를 각각 제어하는 것이 복잡성을 증대시키므로, 미리 정의 된 효과 위주로 LED를 플레이합니다.

각 함수의 인자에 해당하는 코드 값은 [표 6-3]와 같습니다.

코드 값	눈 LED	눈 색깔	귀	입	이마
0	상태유지		상태유지	상태유지	끄기
1	빙글빙글	빨간	켜기	가운데 ON	약
2	눈썹깜박	파란	끄기	세 개 ON	중
3	눈썹정지	보라	깜박	가운데 깜박	강
4	전체켜기			세 개 깜박	
5	전체깜박			말하기	
6	좌우효과			끄기	
7	상하효과				
8	끄기				

모든 LED를 끄게 만드는 82~85줄의 turnAllLedOff()를 표 [6-3]를 참조하여 살펴보면 다음과 같습니다.

```
myRobotApp.mBinder.dmel_hri_set_mouth_led(getComponentName(), 6);
myRobotApp.mBinder.dmel_hri_set_ear_led(getComponentName(), 2); myRobotApp.
mBinder.dmel_hri_set_brow_beam(getComponentName(), 0); myRobotApp.mBinder.
dmel_hri_set_eye_led(getComponentName(), 8, 1)
```

82번째 줄의 dmel_hri_set_mouth_led() 함수의 인자 값으로 지정 된 코드 값은 6입니다. [표6-2]에서 입의 6은 끄기입니다. 그러므로 입 LED 3개가 모두 꺼집니다. 83번 째 줄의 hri_set_ear_led()의 코드 값은 2입니다. 이 코드 값은 귀 LED를 모두 끄게 합니다. 다음 줄의 dmel_hri_set_brow_beam()은 이마 휘도 LED 제어 함수로 코드 값 0은 이마의 휘도 LED를 끄게 합니다. 마지막 줄의 dmel_hri_set_eye_led()의 인자는 눈 LED 코드 값이 8이고, 눈 색깔 코드 값이 1입니다. 눈 LED 코드값 8은 LED를 끄게 만듭니다. 그러므로 눈 색깔 값인 1은 영향을 미치지 못합니다. 비록 모든 눈 LED를 끄게 되더라도 눈 색깔 값은 아무 색깔로 반드시 지정하여야 합니다.

마찬가지로 97~100줄의 turnAllLedOn()를 [표 6-2]를 참조하여 해석하면, 모든 LED를 켜주게 됩니다.

```
myRobotApp.mBinder.dmel_hri_set_mouth_led(getComponentName(), 2);
myRobotApp.mBinder.dmel_hri_set_brow_beam(getComponentName(), 2); myRobot-
App.mBinder.dmel_hri_set_ear_led(getComponentName(), 1); myRobotApp.mBinder.
dmel_hri_set_eye_led(getComponentName(), 4, 2);
```

이번 예제도 화면의 리스트에 다른 항목들을 추가해보고 다양한 LED 효과를 직접 만들어 보시기 바랍니다. 간혹 귀와 입 LED가 제어가 되지 않는 경우가 있을 수 있습니다. 이 경우는 입의 가운데 LED가 깜박거립니다. 이는 LED를 제어하는 머리 보드의 에러상태를 나타내는 것이며, 이는 로봇 자율모드에서 자동으로 복구됩니다.

6.5 TTS 제어

TTS란 Text-to-Speech의 약자로 텍스트를 로봇음성로 바꿔주는 기능입니다. 예를 들어, 로봇에 텍스트로 "안녕하세요"를 입력해주면, 로봇이 "안녕하세요"라고 말하는 기능입니다. TTS 기능을 활용하면 로봇이 사람처럼 말하는 효과를 낼 수 있기 때문에, 현재 거의 모든 로봇에서 사용되고 있습니다.
Hovis Genie에서는 한국어 TTS 엔진을 탑재하고 있습니다. 이번 예제는 한국어 TTS를 이용하여 로봇이 말하는 기능을 구현합니다.

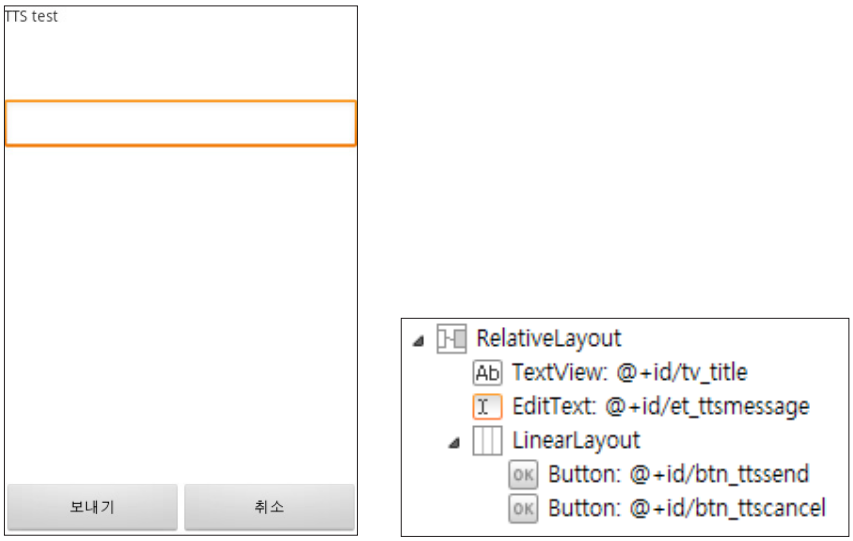
TTS 제어 Activity에 UI를 구성하는 activity_testtts.xml을 /res/layout에 추가하고 이에 상응하는 TestTTSActivity.java를 /src/com/dongburobot/genieapidemo에 추가합니다. 이어서 activity_testtts.xml을 열고 [Listing 6-11]을 참조하여 코드를 수정합니다.

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent" >
5:
6:     <!-- 테스트 타이틀 -->
7:     <TextView
8:         android:id="@+id/tv_title"
9:         android:layout_width="match_parent"
10:        android:layout_height="85dp"
11:        android:text="LED test"
12:        />
13:
14:    <!-- LED 제어 리스트 -->
15:    <EditText
16:        android:id="@+id/et_ttsmessage"
17:        android:layout_width="match_parent"
18:        android:layout_height="wrap_content"
19:        android:hint="문자를 입력하세요."
20:        android:maxLength="140"
```

```
20:          android:maxLength="140"
21:          android:layout_below="@id/tv_title"
22:          />
23:
24:          <!— 시작과 종료 버튼 —>
24:          <!— 시작과 종료 버튼 —>
25:      <LinearLayout
26:          android:layout_width="match_parent"
27:          android:layout_height="wrap_content"
28:          android:layout_alignParentBottom="true">
29:          <Button
30:              android:id="@+id/btn_ttssend"
31:              android:layout_width="160dp"
32:              android:layout_height="wrap_content"
33:              android:text="보내기"
34:              />
35:          <Button
36:              android:id="@+id/btn_ttscancel"
37:              android:layout_width="160dp"
38:              android:layout_height="wrap_content"
39:              android:text="취소"
40:              />
41:      </LinearLayout>
42: </RelativeLayout>
```

[Listing 6–7] activity_testmotion.xml

TTS 제어 화면의 레이아웃 구성은 최상위가 〈RelativieLayout〉으로 구성되어, 각 위젯의 배치를 위젯 간의 상대적인 위치로 지정합니다. 〈TextView〉의 경우 상대 위치 속성이 없기 때문에 화면의 좌측상단에 배치됩니다. 〈EditText〉는 21번째 줄에 지정된 속성에 의해 〈TextView〉에 아래에 배치됩니다. 〈LinearLayout〉은 28번째 줄의 속성에 의해 〈RelativeLayout〉에 좌측하단에 배치됩니다. 여기서 〈LinearLayout〉은 하위에 두 개의 〈Button〉을 가지는 데 이 위젯이 〈LinearLayout〉의 영향을 받아 최하단에 각각 배치됩니다. 이렇게 배치 된 위젯들의 화면 구성은 [그림 6-8]과 같습니다.



[그림 6-8] TTS 제어 화면

TTS 제어 화면에서 유저의 입력은 〈EditText〉를 통해 이루어집니다. 〈EditText〉 위젯은 [그림 6-8]의 황색 테두리로 둘러진 부분입니다. 〈EditText〉에 터치를 하게 되면 키보드가 화면에 나오고 텍스트를 입력할 수 있으며, Back버튼을 누르면 키보드가 사라집니다. 〈EditText〉의 입력은 20번째 줄에 기술 된 android:maxLength="140" 속성에 의해 140글자까지 입력할 수 있습니다. 그리고 〈EditText〉의 텍스트 입력이 많아서 줄이 넘어간다면android:layout_height="wrap_content"에 의해 〈EditText〉의 크기는 내용만큼 늘어납니다.

예제를 위한 UI가 구성되었으므로, TestTTSActivity.java 파일을 열고 소스 코드를 다음과 같이 수정합니다.

1) BaseActivity 상속

```
public class TestOmniwheelActivity extends BaseActivity …
```

2) 위젯 관련 멤버변수 선언

```
private EditText mTTSEditText;
private Button mBtnTTSSend;
private Button mBtnTTSCancel;
```

3) onCreate() 작성

onCreate()에서는 레이아웃 XML파일을 로드하고, 각 위젯 멤버변수에 리소스를 할당합니다. 그리고 각 버튼에 터치이벤트를 설정합니다.

4) OnClickListener 인터페이스 다중 상속 후 onClick() 작성

이번 예제는 '보내기'와 '취소' 두 가지 버튼이 있습니다. 이 버튼을 눌렀을 때, 터치이벤트를 처리하기 위한 소스 코드를 작성합니다. 우선 OnClickListener 인터페이스를 다중상속합니다.

```
public class TestTTSActivity extends BaseActivity implements OnClickListener …
```

'implements OnClickListener'를 추가하고 'TestTTSActivity' 위의 마우스 커서를 올리면 onClick()을 추가할 수 있습니다.

5) playTTS(), stopTTS() 작성

TTS를 재생하고 중단하는 메소드인 playTTS()와 stopTTS()를 작성합니다. 이 두 메소드는 유저가 '보내기', '취소' 버튼을 누를 때, onClick()에 의해 호출됩니다.

6) onBackPressed() 작성

이전 예제까지는 onKeyDown()을 오버라이드하여 Back버튼 처리를 하였습니다. onKeyDown()의 경우 어떤 키가 눌려도 호출되며, 함수 내부에서 눌러진 버튼이 Back버튼인지 확인하여 처리하지만, onBackPressed()는 오직 Back버튼이 눌린 경우만 호출됩니다.
로봇 App은 항상 하나의 Activity만 유지하고 다른 Activity가 스택에 쌓이지 않도록 설계되어 있어 Back버튼을 누를 때 App이 종료될 수 있는데, onBackPressed()는 이를 방지하고 메인화면으로 돌아가게 하는 역할을 합니다.

```
 1: package com.dongburobot.genieapidemo;
 2:
 3: import android.content.Intent;
 4: import android.os.Bundle;
 5: import android.os.RemoteException;
 6: import android.view.View;
 7: import android.view.View.OnClickListener;
 8: import android.widget.Button;
 9: import android.widget.EditText;
10:
11: public class TestTTSActivity extends BaseActivity implements OnClickListener {
12:
13:     private EditText mTTSEditText;
14:     private Button mBtnTTSSend;
15:     private Button mBtnTTSCancel;
16:
17:     @Override
18:     protected void onCreate(Bundle savedInstanceState) {
19:         super.onCreate(savedInstanceState);
20:         setContentView(R.layout.activity_testtts);
21:
22:         mTTSEditText = (EditText)findViewById(R.id.et_ttsmessage);
23:
24:         mBtnTTSSend = (Button)findViewById(R.id.btn_ttssend);
25:         mBtnTTSCancel = (Button)findViewById(R.id.btn_ttscancel);
26:         mBtnTTSSend.setOnClickListener(this);
27:         mBtnTTSCancel.setOnClickListener(this);
28:     }
29:
30:     @Override
31:     public void onBackPressed() {
32:         super.onBackPressed();
33:         Intent intent = new Intent(this, MainActivity.class);
34:         startActivity(intent);
35:     }
36:
```

```
37:    public void onClick(View v) {
38:        switch (v.getId()) {
39:        case R.id.btn_ttssend:
40:            playTTS(mTTSEditText.getText().toString());
41:            break;
42:        case R.id.btn_ttscancel:
43:            stopTTS();
44:            break;
45:
46:        default:
47:            break;
48:        }
49:    }
50:
51:    private void playTTS(String msg) {
52:        stopTTS();
53:
54:        if (myRobotApp.mBinder == null)
55:            return;
56:
57:        try {
58:            myRobotApp.mBinder.dmel_tts_speak(getComponentName(), msg);
59:        } catch (RemoteException e) {
60:            e.printStackTrace();
61:        }
62:        return;
63:    }
64:
65:    private void stopTTS() {
66:        if (myRobotApp.mBinder == null)
67:            return;
68:
69:        try {
70:            myRobotApp.mBinder.dmel_tts_stop(getComponentName());
71:            Thread.sleep(200);
72:        } catch (RemoteException e) {
```

```
73:                 e.printStackTrace();
74:             } catch (InterruptedException e) {
75:                 e.printStackTrace();
76:             }
77:             return;
78:         }
79: }
```

[Listing 6-12] TestTTSActivity.java

이번 예제의 핵심은 51~79의 onClick()에 의해 호출되는 playTTS()와 stopTTS() 입니다.

playTTS()는 String형의 문자열을 인자로 받습니다. 예제에서는 유저가 '보내기' 버튼을 누를 때 EditText에 입력한 문자열을 String형으로 변환하여 인자로 취하며, 40번째 줄에 해당합니다.

```
40: playTTS(mTTSEditText.getText().toString());
```

이는 playTTS() 58번째 줄의 TTS 관련 로봇 서비스함수인 dmel_tts_speak()에 전달되어 텍스트를 소리로 치환합니다.

```
private void playTTS(String msg) {
...
58:       myRobotApp.mBinder.dmel_tts_speak(getComponentName(), msg);
...
}
```

dmel_tts_speak

```
void dmel_tts_speak(ComponentName cn,
                    java.lang.String text)
                    throws android.os.RemoteException
TTS(Text-to-Speech)를 Play한다.
Parameters:
cn - Component 이름
text - 음성으로 변환할 문자열 String ex)"안녕하세요. 호비스지니입니다."
Throws:
android.os.RemoteException
```

playTTS() 52번째 줄에서는 TTS의 정확한 동작을 보장하기 stopTTS()를 수행합니다. TTS 명령의 우선순위는 로봇 자원을 먼저 선점한 TTS가 더 높습니다. 그러므로 TTS가 이미 실행되고 있다면, 로봇 서비스는 이후의 TTS명령을 무시하게 됩니다.

65~78줄의 stopTTS()는 현재 수행되고 있는 TTS명령을 중단하는 역할을 합니다. stopTTS()는 내부적으로 dmel_tts_stop()을 수행합니다. dmel_tts_stop()은 TTS를 실질적으로 중단하도록 하는데, 중단까지의 약간의 소요시간이 발생할 수 있습니다. 그러므로 프로그램을 안전하게 동작하도록 Thread.sleep(200)을 이용해서 약간의 딜레이를 주는 것이 좋습니다.

```
myRobotApp.mBinder.dmel_tts_stop(getComponentName());
Thread.sleep(200);
```

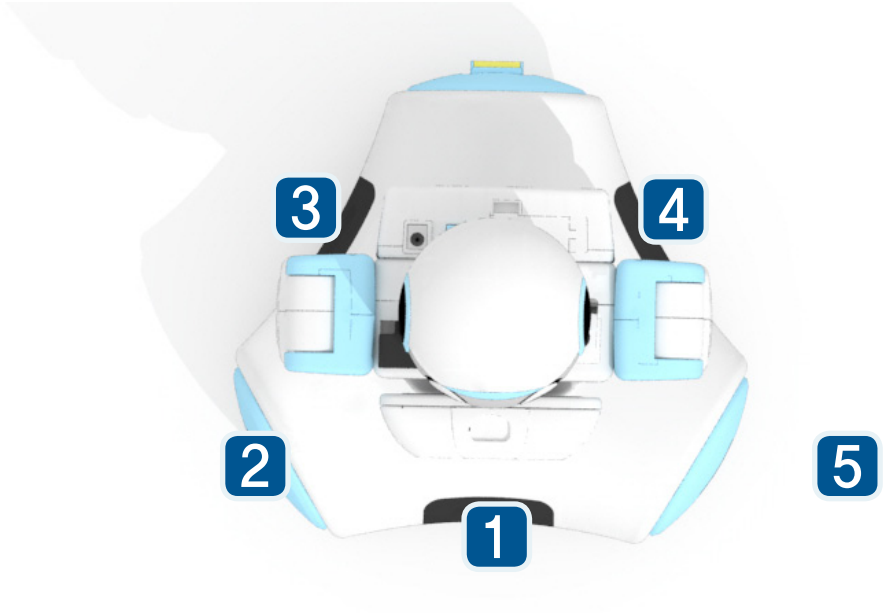
dmel_tts_stop

```
void dmel_tts_stop(ComponentName cn)
                    throws android.os.RemoteException
TTS(Text-to-Speech)를 중단한다.
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException
```

지금까지 로봇서비스 함수를 사용한 playTTS()와 stopTTS()를 통해 TTS 제어를 해보았습니다. TTS는 활용도가 높으며, 로봇을 똑똑하게 보이는 데에 사용할 수 있습니다. 다만, TTS에게 사람과 같은 발음을 기대하기는 무리가 있습니다. 이 경우 텍스트를 발음나는 대로 써보는 것도 TTS를 통한 로봇 음성을 질을 높이는 한가지 노하우가 될 수 있습니다. 예를 들면, “괜찮아요?” 대신에 “괜차나요?” 또는 “학업” 대신에 “하겁”을 사용할 수 있습니다.

6.6 PSD 센서 제어

Hovis Genie는 주변환경을 인지하기 위해 5개의 PSD(Position Sensitive Device) 센서를 탑재하고 있습니다. PSD 센서는 주변의 거리 정보를 최대 80cm까지 획득하여 장애물을 인지합니다. 로봇은 PSD 센서를 통해 장애물을 회피하거나 여러 주행 전략을 수립할 수 있습니다.



[그림 6-9] 거리 센서의 배치

Hovis Genie에는 정면을 기준으로 시계방향으로 1~5번의 PSD 센서가 있습니다. 이번 예제는 각 PSD의 값을 실시간으로 획득하여 화면에 출력해보도록 하겠습니다.

PSD 센서 제어의 UI파일은 activity_testpsdsensor.xml이며, 해당 소스코드는 TestPSDActivity.java입니다. 해당 파일을 GenieApiDemo 프로젝트에 추가합니다. 이어서, activity_testpsdsensor.xml을 열고 다음과 같이 수정합니다.

```
1: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2:     xmlns:tools="http://schemas.android.com/tools"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent" >
5:
6:     <!-- 테스트 타이틀 -->
7:     <TextView
8:         android:id="@+id/tv_title"
9:         android:layout_width="match_parent"
10:        android:layout_height="85dp"
11:        android:text="PSD Sensor test"
12:        />
13:
14:    <LinearLayout
15:        android:layout_width="match_parent"
16:        android:layout_height="wrap_content"
17:        android:layout_below="@id/tv_title"
18:        android:orientation="vertical"
19:        >
20:        <!-- 1번 PSD -->
21:        <LinearLayout
22:            android:layout_width="match_parent"
23:            android:layout_height="wrap_content"
24:            >
25:            <TextView
26:                android:layout_width="70dp"
27:                android:layout_height="wrap_content"
28:                android:text="전방"
29:                />
30:            <TextView
31:                android:id="@+id/tv_psd1"
32:                android:layout_width="40dp"
33:                android:layout_height="wrap_content"
34:                />
35:            <SeekBar
36:                android:id="@+id/seekbar_psd1"
```

```
37:                 android:layout_width="200dp"
38:                 android:layout_height="wrap_content"
39:                 android:max="80"
40:                 />
41:         </LinearLayout>
42:
43:         <!-- 2번 PSD -->
44:           <LinearLayout
45:            android:layout_width="match_parent"
46:            android:layout_height="wrap_content"
47:            >
48:            <TextView
49:                 android:layout_width="70dp"
50:                 android:layout_height="wrap_content"
51:                android:text="전방우측"
52:                 />
53:            <TextView
54:                 android:id="@+id/tv_psd2"
55:                 android:layout_width="40dp"
56:                 android:layout_height="wrap_content"
57:                 />
58:            <SeekBar
59:                 android:id="@+id/seekbar_psd2"
60:                 android:layout_width="200dp"
61:                android:layout_height="wrap_content"
62:                 android:max="80"
63:                 />
64:         </LinearLayout>
65:
66:           <!-- 3번 PSD -->
67:           <LinearLayout
68:            android:layout_width="match_parent"
69:            android:layout_height="wrap_content"
70:            >
71:           <TextView
```

```
72:                 android:layout_width="70dp"
73:                 android:layout_height="wrap_content"
74:                 android:text="후방우측"
75:                 />
76:             <TextView
77:                 android:id="@+id/tv_psd3"
78:                 android:layout_width="40dp"
79:     }
80:
81:             <SeekBar
82:                 android:id="@+id/seekbar_psd3"
83:                 android:layout_width="200dp"
84:                 android:layout_height="wrap_content"
85:                 android:max="80"
86:                 />
87:         </LinearLayout>
88:
89:             <!-- 4번 PSD -->
90:             <LinearLayout
91:             android:layout_width="match_parent"
92:              android:layout_height="wrap_content"
93:             >
94:             <TextView
95:                 android:layout_width="70dp"
96:                 android:layout_height="wrap_content"
97:                 android:text="후방좌측"
98:                 />
99:             <TextView
100:                android:id="@+id/tv_psd4"
101:               android:layout_width="40dp"
102:                android:layout_height="wrap_content"
103:                />
104:            <SeekBar
105:                android:id="@+id/seekbar_psd4"
106:                android:layout_width="200dp"
107:                android:layout_height="wrap_content"
```

```
                android:max="80"
                />
        </LinearLayout>

        <!-- 5번 PSD -->
        <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         >
         <TextView
             android:layout_width="70dp"
             android:layout_height="wrap_content"
              android:text="전방좌측"
             />
          <TextView
              android:id="@+id/tv_psd5"
              android:layout_width="40dp"
              android:layout_height="wrap_content"
              />
          <SeekBar
              android:id="@+id/seekbar_psd5"
              android:layout_width="200dp"
              android:layout_height="wrap_content"
             android:max="80"
              />
      </LinearLayout>
  </LinearLayout>

  <!-- 시작과 종료 버튼 -->
  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true">
      <Button
          android:id="@+id/btn_start"
          android:layout_width="160dp"
```

```
144:            android:layout_height="wrap_content"
145:            android:text="시작"
146:            />
147:        <Button
148:            android:id="@+id/btn_stop"
149:            android:layout_width="160dp"
150:            android:layout_height="wrap_content"
151:            android:text="종료"
152:            />
153:    </LinearLayout>
154:
155: </RelativeLayout>
```

[Lisiting 6-11] activity_testpsdsensor.xml

이번 PSD 제어 화면은 UI 소스코드가 다소 깁니다. 그러나 반복되는 위젯이 있기 때문에 구조는 단순합니다.

최상위 태그는 <RelativeLayout>입니다. <RelatvieLayout>의 하위태그는 7~12줄의 <TextView>, 14~134줄의 <LinearLayout>, 그리고 137~153줄의 <LinearLayout>입니다. 여기서 <TextView>는 "PSD Sensor test"를 출력하고, <LinearLayout> 5개의 PSD를 정보를 표현하며, 마지막 <LinearLayout>은 '시작'과 '종료'버튼을 표현합니다.

<RelativeLayout>에 포함된 태그는 각 위젯의 위치를 상대적으로 표현해야 합니다. 7~12번째 줄의 <TextView>는 위치 지정 속성이 없으므로 부모 태그의 좌측상단에 배치됩니다. 14~134줄의 <LinearLayout>은 17번째 줄의 속성에 의해 <TextView> 하단에 배치됩니다. 137~153줄의 <LinearLayout>은 140번째 줄의 속성에 의해 부모인 <RelativeLayout>의 최하단에 배치됩니다.

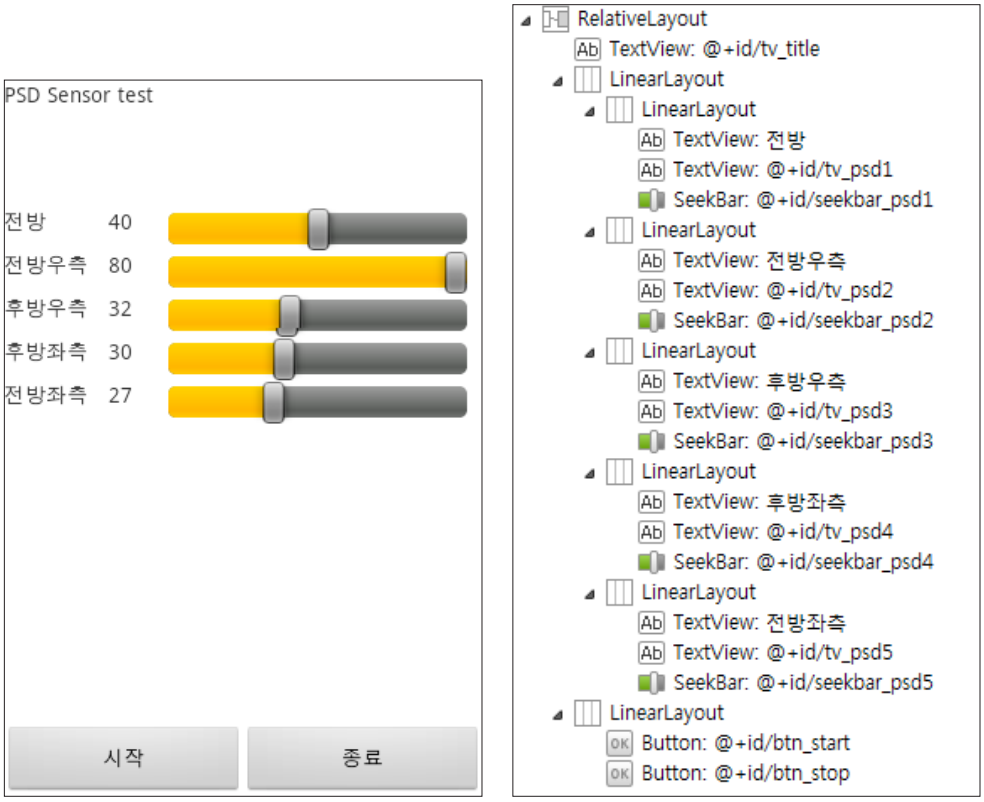
<RelativeLayout>의 하위 태그인 14~134줄의 <LinearLayout>은 1번부터 5번 PSD 센서의 정보를 출력하는 <LinearLayout> 5개를 포함합니다. 5개의 <LinearLayout>은 수직으로 배치되는데, 이는 부모 태그인 14~134줄의 <LinearLayout>의 18번째 줄의 속성에 의해 결정됩니다.

5개의 <LinearLayout> 각각은 실제 PSD 정보를 출력하는 3개의 위젯인 <TextView>, <TextView>, 그리고 <SeekBar>를 하위태그로 가집니다. 각각의 <LinearLayout>는 수평, 수직배치에 대한 설정을 하지 않았으므로, 포함된 3개의 위젯은 가로로 배치됩니다.

5개의 〈LinearLayout〉은 포함 된 3개의 위젯은 ID만 다르고 완전히 같으므로, 하나의 〈LinearLayout〉만 살펴보겠습니다. 21~41줄의 〈LinearLayout〉은 1번 PSD에 대한 정보를 출력합니다. 첫 번째 하위 태그인 〈TextView〉는 센서 배치 위치를 텍스트로 표현합니다. 두번 째 〈TextView〉는 PSD 센서로 측정된 거리 값을 cm 단위로 출력합니다. 마지막 〈SeekBar〉는 거리 값을 그래픽하게 표현합니다. 〈SeekBar〉에는 android:max="80"이라는 속성이 있습니다. 이는 〈SeekBar〉의 표현범위를 0~80로 지정하며, 80은 PSD 센서가 측정할 수 있는 최대거리 값이기도 합니다.

마지막으로 가장 하단에 〈RelativeLayout〉의 하위 태그인 〈LinearLayout〉이 있습니다. 137~153줄의 〈LinearLayout〉은 두 개의 〈Button〉 위젯을 포함합니다. 이 〈LinearLayout〉에서는 수평, 수직배치에 대한 속성이 지정되지 않았으므로, 버튼들은 수평으로 배치됩니다.

이렇게 구성 된 UI는 [그림 6-10]과 같습니다.



[그림 6-10] PSD 센서 제어 화면

UI구성을 완료하였으므로, TestPSDActivity.java를 열어 PSD 센서 제어를 위한 프로그래밍을 시작합니다. 다음은 소스코드 작성의 가이드라인입니다.

1) BaseActivity 상속

```
public class TestPSDActivity extends BaseActivity …
```

2) 위젯 관련 멤버변수 선언 및 Handler 선언

17~23줄을 참고하여 위젯과 Handler 객체를 멤버변수로 선언합니다. mTvPSD는 각 PSD의 거리 값을 TextView에 출력하기 위해 사용되는 TextView 객체변수이며, mSbPSD는 거리 값을 SeekBar에 출력하기 위한 객체변수입니다. mBtnStart와 mBtnStop은 버튼을 제어하기 위한 버튼 객체입니다.

```
private TextView[ ] mTvPSD = new TextView[5];
private SeekBar[ ] mSbPSD = new SeekBar[5];
private Button mBtnStart;
private Button mBtnStop;
private Handler mHandler;
```

이번 예제의 핵심은 Handler의 사용입니다. Handler는 PSD 센서 거리 값을 실시간으로 획득하여 화면에 업데이트하는 역할을 합니다.
Handler란 Thread와 유사한 기능을 합니다. Thread는 한 프로세스 내에서 독자적인 작업을 병렬적으로 수행합니다. 이번 예제의 경우 PSD 센서 값을 지속적으로 읽어 화면에 업데이트 하는 작업을 실시간으로 수행하면서, 유저의 터치 입력을 처리해야 되므로 Thread의 기능을 사용할 필요가 있습니다.
그러나 결정적인 Thread의 단점은 GUI(Graphic User Interface)에서 안전하지 않다는 점입니다. 안드로이드상에서 Thread로 화면에 관련 된 작업을 하면 대다수의 경우 에러가 발생합니다. 이를 방지하기 위해서 Handler를 사용합니다.
Handler의 내부적으로는 Thread와 Thread Queue를 가지고 있으며, 자세한 내용은 안드로이드 개발자 사이트를 참고합니다. 여기에서는, 'UI에 관련 실시간 작업은 무조건 Handler를 사용한다' 정도로 정리합니다.

3) onCreate() 작성

onCreate()에서는 28~45줄을 통해 레이아웃 XML파일을 로드하고, 각 위젯 멤버변수에 리소스를 할당합니다. 그리고 각 버튼에 터치이벤트를 설정합니다.

onCreate()에서 가장 중요한 부분은 47~71줄의 Handler입니다. 등록된 Handler는 메시지에 의해 수행을 시작하거나 종료합니다. 이는 소스코드 작성 후 다시 설명합니다.

4) OnClickListener 인터페이스 다중 상속 후 onClick() 작성

이번 예제는 '시작'와 '종료' 두 가지 버튼이 있습니다. '시작'은 PSD 센서 값을 읽기를 시작하는 부분이며, '종료'는 이를 종료합니다. onClickListener 인터페이스를 다중상속하고 onClick()을 작성합니다.

```
public class TestPSDActivity extends BaseActivity implements OnClickListener …
```

이클립스에서 'implements OnClickListener'를 추가하고 'TestPSDActivity' 위의 마우스 커서를 올리면 onClick()을 추가할 수 있습니다. onClick()은 눌린 버튼에 따라 Handler에 메시지를 전달하거나 중단하는 역할을 합니다.

5) onPause() 작성

onPause()는 Activity가 종료될 때 호출되는 메소드입니다. 이번 PSD 제어 Activity에서는 종료될 때 Handler의 수행을 종료하는 코드를 추가합니다.

6) onBackPressed() 작성.

로봇 App은 항상 하나의 Activity만 유지하고 다른 Activity가 스택에 쌓이지 않도록 설계되어 있어 Back버튼을 누를 때 App이 종료될 수 있는데, onBackPressed()는 이를 방지하고 메인화면으로 돌아가게 하는 역할을 합니다.

```
 1: package com.dongburobot.genieapidemo;
 2:
 3: import android.annotation.SuppressLint;
 4: import android.content.Intent;
 5: import android.os.Bundle;
 6: import android.os.Handler;
 7: import android.os.RemoteException;
 8: import android.util.Log;
 9: import android.view.View;
10: import android.view.View.OnClickListener;
11: import android.widget.Button;
12: import android.widget.SeekBar;
13: import android.widget.TextView;
14:
15: public class TestPSDSensorActivity extends BaseActivity implements OnClickListener {
16:
17:     private TextView[] mTvPSD = new TextView[5];
18:     private SeekBar[] mSbPSD = new SeekBar[5];
19:
20:     private Button mBtnStart;
21:     private Button mBtnStop;
22:
23:     private Handler mHandler;
24:
25:     @Override
26:     protected void onCreate(Bundle savedInstanceState) {
27:         super.onCreate(savedInstanceState);
28:         setContentView(R.layout.activity_testpsdsensor);
29:
30:         mTvPSD[0] = (TextView)findViewById(R.id.tv_psd1);
31:         mTvPSD[1] = (TextView)findViewById(R.id.tv_psd2);
32:         mTvPSD[2] = (TextView)findViewById(R.id.tv_psd3);
33:         mTvPSD[3] = (TextView)findViewById(R.id.tv_psd4);
34:         mTvPSD[4] = (TextView)findViewById(R.id.tv_psd5);
35:
36:         mSbPSD[0] = (SeekBar)findViewById(R.id.seekbar_psd1);
```

```
37:         mSbPSD[1] = (SeekBar)findViewById(R.id.seekbar_psd2);
38:         mSbPSD[2] = (SeekBar)findViewById(R.id.seekbar_psd3);
39:         mSbPSD[3] = (SeekBar)findViewById(R.id.seekbar_psd4);
40:         mSbPSD[4] = (SeekBar)findViewById(R.id.seekbar_psd5);
41:
42:         mBtnStart = (Button)findViewById(R.id.btn_start);
43:         mBtnStop = (Button)findViewById(R.id.btn_stop);
44:         mBtnStart.setOnClickListener(this);
45:         mBtnStop.setOnClickListener(this);
46:
47:         mHandler = new Handler() {
48:             @SuppressLint("HandlerLeak")
49:             public void handleMessage(android.os.Message msg) {
50:                 if (myRobotApp.mBinder == null)
51:                     return;
52:
53:                 double[] psdValue = null;
54:
55:                 try {
56:                     psdValue =
                        myRobotApp.mBinder.dmel_robot_get_obs(getComponentName());
57:                 } catch (RemoteException e) {
58:                     e.printStackTrace();
59:                 }
60:
61:                 //UI 반영
62:                 for (int i = 0; i < 5; i++) {
63:                     if ( (int)(psdValue[i]*100) != 100 ) {
64:                         mTvPSD[i].setText( String.valueOf((int)(psdValue[i]*100)) );
65:                         mSbPSD[i].setProgress((int)(psdValue[i]*100));
66:                     }
67:                 }
68:                 mHandler.sendEmptyMessageDelayed(0, 100);
69:             };
70:         };
71:     }
72:
```

```
73:     @Override
74:     public void onBackPressed() {
75:         super.onBackPressed();
76:
77:         Intent intent = new Intent(this, MainActivity.class);
78:         startActivity(intent);
79:     }
80:
81:     @Override
82:     protected void onPause() {
83:         if (mHandler != null) {
84:             if (mHandler.hasMessages(0))
85:                 mHandler.removeMessages(0);
86:
87:             mHandler = null;
88:         }
89:
90:         super.onPause();
91:     }
92:
93:     public void onClick(View v) {
94:         switch (v.getId()) {
95:         case R.id.btn_start:
96:             mHandler.sendEmptyMessage(0);
97:             break;
98:
99:         case R.id.btn_stop:
100:            if (mHandler.hasMessages(0))
101:                mHandler.removeMessages(0);
102:            break;
103:
104:        default:
105:            break;
106:        }
107:    }
108: }
```

[Listing 6–14] TestPSDActivity.java

TestPSDActivity가 화면에 나타날 때, onCreate()가 호출됩니다. onCreate()에서는 Handler를 등록합니다.

```
mHandler = new Handler() {
...
public void handleMessage(android.os.aMessage msg) {
...
};
};
```

mHandler에 Handler 객체를 할당할 때 반드시 재정의 해주어야 하는 메소드로 handleMessage()가 있습니다. handleMessage()는 Handler가 메시지를 받을 때 호출되며, 메시지를 구분할 수 있도록 android.os.Message msg를 인자로 가집니다.

예제에서 등록된 Handler에 메시지를 보내는 주체는 '시작' 버튼입니다. '시작'버튼에 대한 처리는 onClick()의 96번째 줄에 정의되어 있으며 다음과 같습니다.

```
96: mHandler.sendEmptyMessage(0);
```

mHandler.sendEmptyMeesage()는 등록된 Handler로 메시지를 보냅니다. sendEmptyMessage()의 인자 값인 0은 Handler로 다수의 메시지를 전달할 때 각 메시지를 구분하는 역할을 합니다. 이 예제에서는 Handler가 메시지를 전달하는 것이 하나이므로 큰 의미는 없습니다. 다만, 메시지가 다수일 때 handleMessage()에서 msg.what을 통해 구분할 수 있습니다. sendEmptyMessage()외에도 메시지를 보내는 방법은 여러 가지가 있으며, 다음 링크를 참조합니다. (http://developer.android.com/reference/android/os/Handler.html)

'시작'버튼에 의해 Handler에 메시지가 전달되면, handleMessage()가 동작합니다. 50번째 줄에서 서비스가 이상 없이 바인딩 되었는지 확인 후, 56번째 줄에서 PSD 센서 거리 값을 리턴하는 로봇 서비스 함수 dmel_robot_get_obs()를 호출합니다.

dmel_robot_get_obs

double[ ] dmel_robot_get_obs(ComponentName cn)
throws android.os.RemoteException
전방 감지 PSD 센서 값을 반환하다. 전방부터 시계방향으로 0, 1, 2, 3, 4
Parameters:
cn – Component 이름
Returns:
전방 감지 센서 5개의 값을 double[0] ~ double[4]에 반환
Throws:
android.os.RemoteException

dmel_robot_get_obs()는 크기가 5인 double형 배열을 리턴합니다. 1번~5번 PSD 센서의 값은 각각 0~4번째 배열에 할당됩니다. 이 때 각 배열에 리턴되는 값은 단위가 m인 거리 값입니다. 각 PSD 센서 값의 정보를 출력하는 TextView와 SeekBar에서는 cm단위의 거리 값을 사용합니다. 62~67줄에서는 m를 cm로 변환합니다.

handleMessage()의 68번째 줄에서는 다시 한번 Handler에게 메시지를 보내 지금까지와 똑 같은 루틴을 반복하도록 합니다. 이는 지속적으로 PSD 센서 값을 화면에 업데이트 하기 위함입니다.

```
68: mHandler.sendEmptyMessageDelayed(0, 100);
```

mHandler에 Handler 객체를 할당할 때 반드시 재정의 해주어야 하는 메소드로 handleMessage()가 있습니다. handleMessage()는 Handler가 메시지를 받을 때 호출되며, 메시지를 구분할 수 있도록 android.os.Message msg를 인자로 가집니다.

예제에서 등록된 Handler에 메시지를 보내는 주체는 '시작' 버튼입니다. '시작'버튼에 대한 처리는 onClick()의 96번째 줄에 정의되어 있으며 다음과 같습니다.

```
100:    if (mHandler.hasMessages(0))
101:        mHandler.removeMessages(0);
```

100번째 줄에서는 Handler에 현재 0으로 구분된 메시지가 전달되고 있는지 확인합니다. 만약 메시지가 있다면, 101번째 줄에서 해당 메시지를 제거합니다. 이후에는, 더 이상 메시지가 Handler에 전달되지 않으며 handleMessage()도 호출되지 않으므로 PSD값 획득과 화면 업데이트를 종료하게 됩니다.

사실 Hovis Genie에는 하단 감지를 위해 3개의 PSD 센서가 더 부착되어 있으며, 로봇상의 배치는 다음과 같습니다.

[그림 6-11] 하단 감지 PSD 센서의 배치

하단 감지 센서의 로봇 서비스함수는 dmel_robot_get_cliff()입니다.

dmel_robot_get_cliff

double[] dmel_robot_get_cliff(ComponentName cn)
throws android.os.RemoteException
바닥 감지 PSD 센서 값을 반환한다. 전방부터 시계방향으로 0, 1, 2
Parameters:
cn – Component 이름
Returns:
바닥 감지 센서 3개의 값을 double[0] ~ double[2]에 반환
Throws:
android.os.RemoteException

이 부분의 대한 예제는 이번 예제와 매우 유사하므로 여러분의 몫으로 남겨둡니다. GenieApiDemo XML 파일과 Java 파일을 추가하여 새로운 Activity를 생성하여 연습해보시기 바랍니다. 그리고 MainActivity의 GridView 항목에 새로운 아이콘과 텍스트를 추가하는 것도 해 보시기 바랍니다. 마지막으로 AndroidManifest.xml에 새로운 Activity 추가도 잊지 마시기 바랍니다.

6.7 터치 센서 제어

이번 예제는 Hovis Genie의 터치 센서 제어입니다. 터치 센서는 로봇과 사용자간의 상호작용을 위해서 활용될 수 있습니다. 로봇에서 터치 센서는 총 3개가 있으며, 센서의 위치는 양 손바닥 그리고 머리입니다.

[그림 6–12] 터치 센서의 배치 (양 손바닥, 머리)

터치 센서 제어 Activity에 UI를 구성하는 activity_testtouchsensor.xml을 /res/layout에 추가하고 이에 상응하는 TestTouchSensorActivity.java를 /src/com/dongburobot/genieapidemo에 추가합니다.

이번 예제는 양손바닥과 머리 터치 여부를 TTS로 알려줍니다. 예제 실행하며 터치 여부를 UI에 표현하지 않습니다. 그래서 UI를 아주 간단히 구성합니다. 〈RelativeLayout〉을 최상위로 하고 〈TextView〉에 'Touch test'를 표시하여, 현재 Activity가 터치 센서 예제임을 알려주는 역할만 합니다. activity_testtouchsensor.xml의 소스코드를 다음과 같이 수정합니다.

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent" >
5:
6:     <!-- 테스트 타이틀 -->
7:     <TextView
8:         android:id="@+id/tv_title"
9:         android:layout_width="match_parent"
10:        android:layout_height="85dp"
11:        android:text="Touch test"
12:        />
13:
14: </RelativeLayout>
```

[Listing 6-15] activity_testtouchsensor.xml

[그림 6-13] 터치 센서 제어 화면

이어서, TestTouchSensorActivity.java 파일을 열고 소스 코드를 다음과 같은 방법으로 작성합니다.

1) BaseActivity 상속

```
public class TestTouchSensorActivity extends BaseActivity …
```

2) Handler 멤버 변수 객체 선언

```
private Handler mHandler;
```

3) onCreate() 작성

onCreate()에서는 setContentView()를 이용하여 레이아웃 XML파일을 로드합니다. 이번 예제 UI는 실행 시 화면 변경 사항이 없으므로 activity_testtouchsensor.xml에 선언 된 리소스들의 위젯 멤버변수 할당은 없습니다.

이전 절에서 PSD 센서 값을 실시간으로 획득하기 위해 Handler가 사용되었듯이, 이번 예제에서도 각 터치 센서의 터치 여부를 판단하기 위해 Handler를 사용하여 실시간으로 센서를 관찰합니다. Handler의 객체를 mHandler에 할당하고 handleMessage()를 오버라이딩합니다.

4) onPause() 작성

Activity 종료시 Handler 종료를 보장하기 위해서 Hanlder의 메시지가 있다면 제거합니다.

5) onBackPressed() 작성

로봇 App은 항상 하나의 Activity만 유지하고 다른 Activity가 스택에 쌓이지 않도록 설계되어 있어 Back버튼을 누를 때 App이 종료될 수 있는데, onBackPressed()는 이를 방지하고 메인화면으로 돌아가게 하는 역할을 합니다.

```
1: package com.dongburobot.genieapidemo;
2:
3: import android.content.Intent;
4: import android.os.Bundle;
5: import android.os.Handler;
6: import android.os.Message;
7: import android.os.RemoteException;
8:
9: public class TestTouchSensorActivity extends BaseActivity {
```

```
10:
11:     private Handler mHandler;
12:
13:     @Override
14:     protected void onCreate(Bundle savedInstanceState) {
15:         super.onCreate(savedInstanceState);
16:         setContentView(R.layout.activity_testtouchsensor);
17:
18:         mHandler = new Handler() {
19:             @Override
20:             public void handleMessage(Message msg) {
21:                 super.handleMessage(msg);
22:
23:                 if (myRobotApp.mBinder == null)
24:                     return;
25:
26:                 boolean[] isHandTouched = null;
27:                 boolean isHeadTouched;
28:                 boolean isNothingTouched = false;
29:
30:                 try {
31:                     isHandTouched =
        myRobotApp.mBinder.dmel_hri_get_hand_touch_info(getComponentName());
32:                     isHeadTouched =
        myRobotApp.mBinder.dmel_hri_get_head_touch_info(getComponentName());
33:
34:                     // 동시에 터치하는 것은 고려하지 않는다.
35:                     if (isHandTouched[0]) {
36:                         myRobotApp.mBinder.dmel_tts_speak(getComponentName(),
                                                    "왼손이 터치되었습니다");
37:                     } else if (isHandTouched[1]) {
38:                         myRobotApp.mBinder.dmel_tts_speak(getComponentName(),
                                                    "오른손이 터치되었습니다");
39:                     } else if (isHeadTouched) {
40:                         myRobotApp.mBinder.dmel_tts_speak(getComponentName(),
                                                    "머리가 터치되었습니다");
```

```
41:                     } else {
42:                         isNothingTouched = true;
43:                     }
44:                 }
45:                 catch (RemoteException e) {
46:                     e.printStackTrace();
47:                 }
48:
49:                 if (isNothingTouched)
50:                     mHandler.sendEmptyMessageDelayed(0, 100);
51:                 else
52:                     mHandler.sendEmptyMessageDelayed(0, 3000);
53:             }
54:         };
55:
56:         mHandler.sendEmptyMessage(1000);
57:     }
58:
59:     @Override
60:     protected void onPause() {
61:         super.onPause();
62:
63:         if (mHandler != null) {
64:             if (mHandler.hasMessages(0))
65:                 mHandler.removeMessages(0);
66:
67:             mHandler = null;
68:         }
69:     }
70:
71:     @Override
72:     public void onBackPressed() {
73:         super.onBackPressed();
74:
75:         Intent intent = new Intent(this, MainActivity.class);
76:         startActivity(intent);
77:     }
78: }
```

[Listing 6–16] TestTouchSensorActivity.java

MainActivity에서 터치 센서 아이콘을 터치하면 TestTouchSensorActivity가 시작됩니다. 이때 호출되는 onCreate()에서 Handler의 객체를 할당하고 handleMessage()를 정의합니다. 그리고 onCreate()의 마지막 줄인 56번째 줄에서 Handler에 메시지를 보냄으로써 정의 된 handleMessage()가 수행됩니다.
handleMessage() 31~32번째 줄의 로봇 서비스 함수 dmel_hri_get_head_touch_info()와 dmel_hri_get_hand_touch_info()는 로봇의 머리와 양 손바닥의 터치 여부를 반환합니다. 각 함수의 원형은 다음과 같습니다.

dmel_hri_get_head_touch_info

boolean dmel_hri_get_head_touch_info(ComponentName cn)
throws android.os.RemoteException
머리 터치 여부를 반환한다.
Parameters:
cn – Component 이름
Returns:
머리 터치시 true, 아니면 false를 반환
Throws:
android.os.RemoteException

dmel_hri_get_hand_touch_info

boolean[] dmel_hri_get_hand_touch_info(ComponentName cn)
throws android.os.RemoteException
양손의 터치 여부를 반환한다.
Parameters:
cn – Component 이름
Returns:
터치시 true, 아니면 false를 반환. boolean[2] (boolean[0] : 왼손터치, boolean[1] : 오른손터치)
Throws:
android.os.RemoteException

35~43번째 줄에서는 각 함수가 반환한 값을 할당한 isHandTouched[]와 isHeadTouched 값을 조사하여 터치 여부를 TTS로 알려줍니다. 다만 이 코드는 유저가 터치 센서를 동시에 터치 할 경우에 대한 처리는 하지 않습니다. 만약 동시에 터치가 되었다면, 왼손, 오른손, 머리 순으로 우선하여 TTS를 출력합니다.
로봇이 아무런 터치를 감지하지 못한다면, 42번째 줄에 의해 isNothingTouched가 true로 설정됩니다. 이 변수는 49~52줄의 if문에서 사용되는데, isNothingTouched가 true이면 100ms 후에 다시 센서 값을 읽도록 Handler에 메시지를 보내 실시간으로 센서 터치를 검사합니다.

반면에, isNothingTouched가 false이면 어떤 터치 센서는 터치를 감지했다는 뜻입니다. 이 경우 터치 된 센서에 따라 그에 맞는 TTS를 출력할 것입니다. 52번째 줄의 3000ms의 시간은 다시 Handler에 메시지를 보내기 전에 TTS 출력이 종료될 때까지의 시간을 보장합니다.

동부로봇 **호비스 지니** 및 **앱**을 이용하여,
로봇 프로그래밍의 원리를 **기초**부터 **재미**있게 **학습**할 수 있습니다.

Dongbu Robot

(주) 동부로봇 www.dongburobot.com
본 사 : 경기도 부천시 원미구 약대동 193 부천테크노파크 401동 11층, TEL : 032-329-5551 (내선 311),
FAX : 032-329-5569, E-MAIL : robotsales@dongbu.com
공 장 : 충청남도 천안시 서북구 직산읍 4산단 6길 27, TEL : 041-590-1700, FAX : 041-590-1701

93550
9 788998 873028
ISBN 978-89-98873-02-8