Learning With HOVIS Genie/App

Android Robot Program-ming

Fundamentals of robot programming

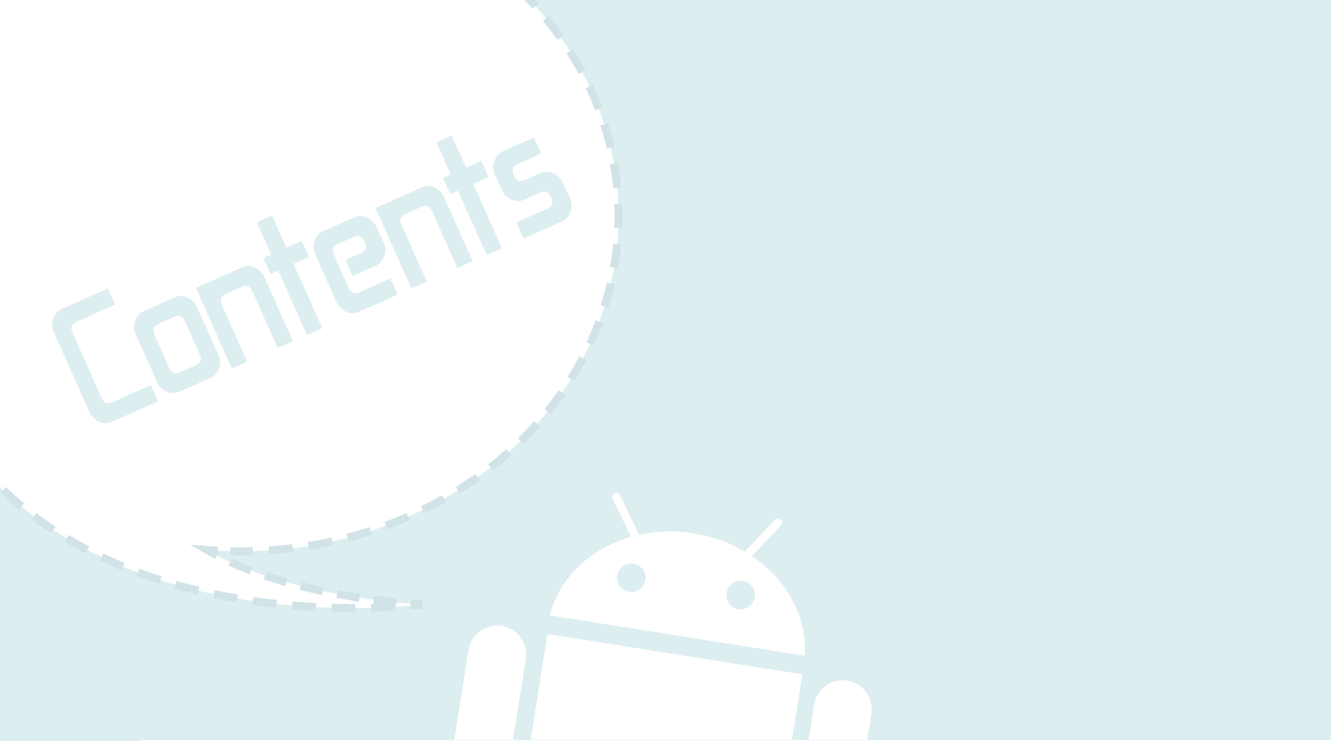
Dongbu Robot Archive

(Download manuals and applications.)

www.dongburobot.com

Dongbu Robot

Contents



01

Introducing Android Robot Hovis Genie

Android
Robot
Programming

04 Starting Hovis Genie Robot Programming

05 Hovis Genie Robot API (Application Programming Interface)

06 Hovis Genie Examples

01 Introducing Android Robot Hovis Genie

1.1 HOVIS Development Background

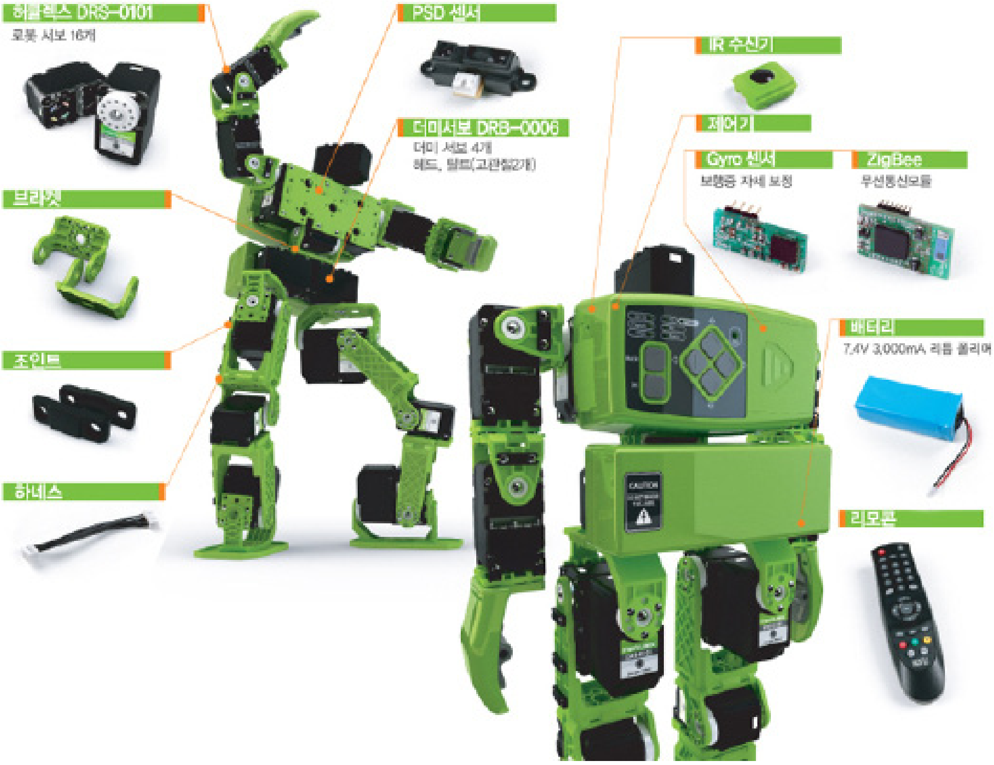
Previously developed service robots were large units incorporating touch screens made for PCs which users operated from standing positions. These robots were mostly developed as part of public sector projects for use in public facilities rather than for use by the consumers. These service robots had limited service contents and could only be programmed by the supplier (manufacturer). Android robot Hovis was developed to overcome these limitations and to expand the services provided by the robot.

The name HOVIS is derived from 'Home Service' and as the name implies Hovis was developed to provide practical services to the consumers. In order for robot to be of practical use at home, it needs software that is capable of handling various situations that can arise at home. Hovis Genie uses the same method as the smart phones to receive software or Apps it requires to perform various tasks required by the users. Apps or software will be developed and supplied by the engineers and application developers using the Android platform. Dongbu Robot will supply the Android robot with Android terminal installed and provide the software (API) and the programming method to create robot service applications

(1) Hovis Lite

Before describing Hovis Genie in detail, brief description of Hovis Lite is required.

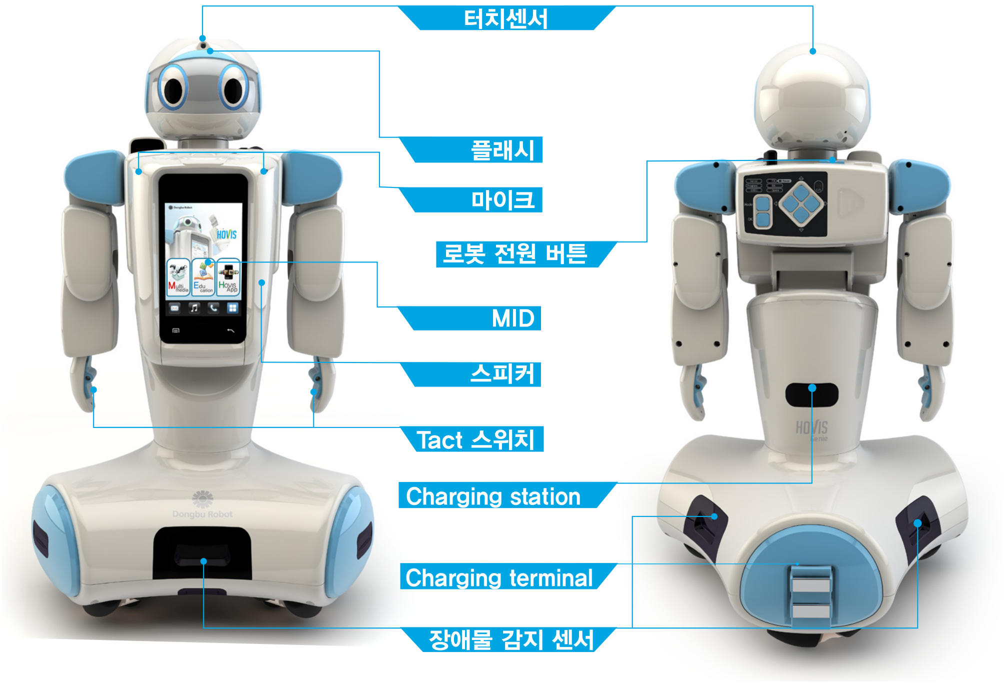
1.2 HOVIS Genie Features



Hovis Lite is a humanoid robot with an 8bit controller using ATmega 128 microprocessor. Developed for robot education, Hovis Lite incorporates servo motors, various sensors, and versatile body. 8bit controller mentioned above called DRC is the brain of the robot. To control the DRC, motor, and various sensors, Hovis Lite is supplied with software tool DR–Visual Logic. DR–Visual Logic is a graphic development language with C language like features such as variable delcaration, function implementation, and logic formation. Also, to faciliate in creating motion for this complex16 axis robot, Hovis Lite is also supplied with motion develoment tool DR–SIM as well. Hovis Genie is based on Hovis Lite with the major differences being the Omni wheel replacing the legs, external casing, and addition of Android terminal which is used to control the DRC and various attached sensors.

(2) Hovis Genie

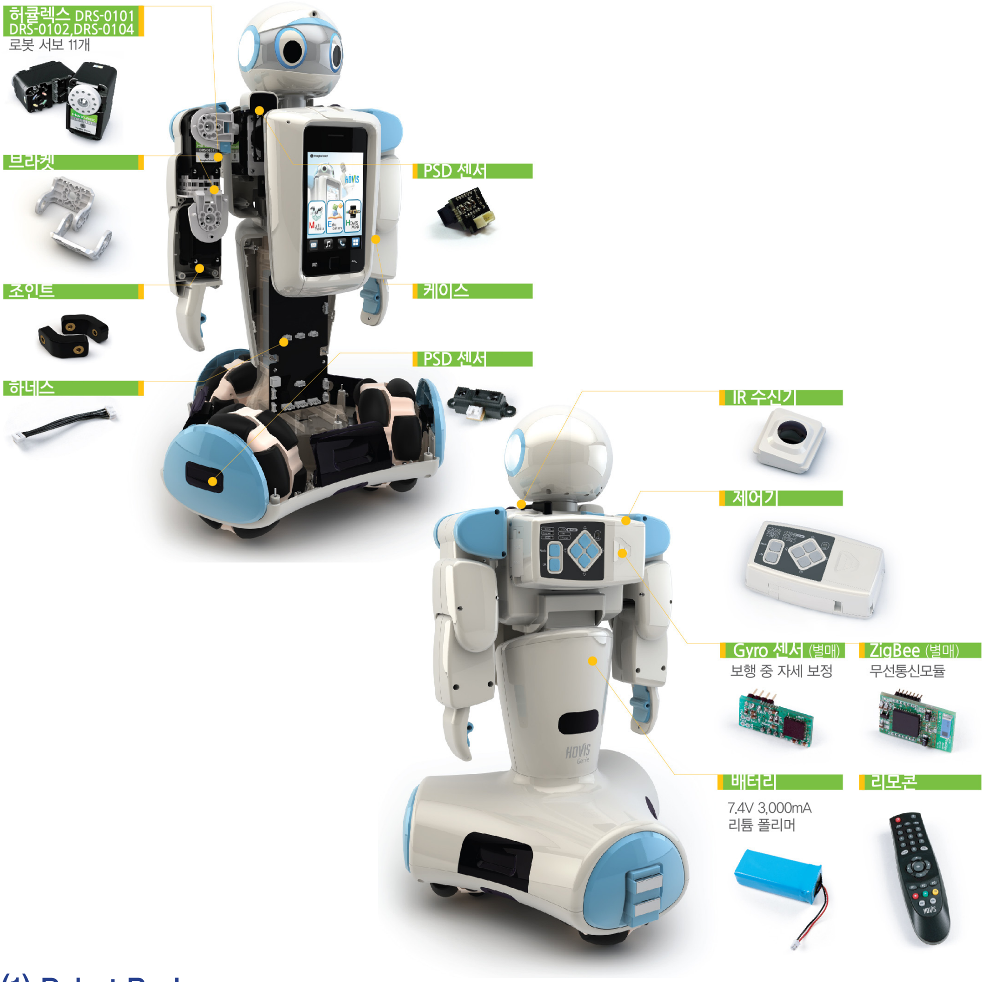
Hovis Genie was developed to provide services at home. Variety of required services can be performed by Genie by running service applications as an App using the Android terminal.



Hovis Genie can express facial emotions using the head touch sensor, flash, and eye LEDs. Tact switches located at the palm of each hand can also be used to solicit response as well. Omni wheels located at the base provides stability and ease of use by allowing Genie to move in any direction without turning, facilitate automatic charging, and makes it possible for Genie to be used for educational purposes. Obstacle detection sensors located near the omni wheel enables Genie to avoid obstacles while moving. Hovis Genie is available as a kit or preassembled unit. Just like Hovis Lite, kit format is for educational purposeses where each owner would learn to assemble and program the robot. Preassembled unit which could also be used for educational purposes does not involve any assembly or programming and comes ready to be used at home simply by running the installed Apps. Assembly instructions for the kit format can be found at archives section of our website dongburobot.com or downloaded from www.hovis.co.kr/guide .

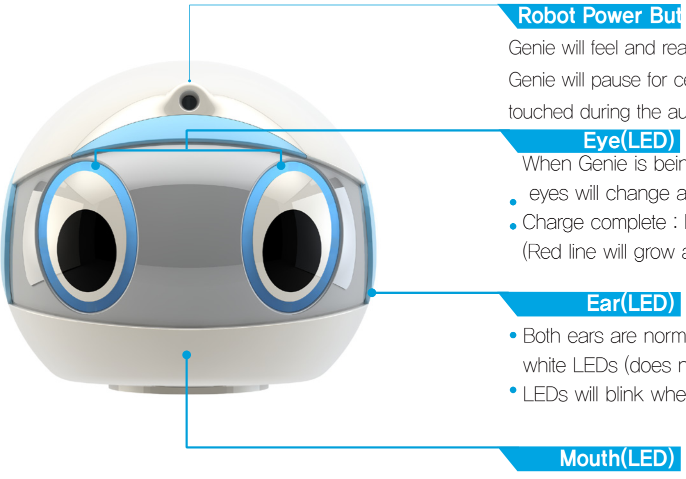
1.3 HOVIS Genie Hardware

Hovis Genie is can be seen as union of two separate parts, robot body and the MID. MID can be attached/detached from the robot and used as regular Android terminal when detached from the robot.



(1) Robot Body

Body is composed of motors, brackets, sensors, IR receiver, and other hardware covered by external shell. Charging stataion, power adapters, remote control, and other accessories are included with the robot. Genie uses MID touch screen, speech recognition, head touch sensor, palm tact switches, and sensors near the omni wheel to receive input. Output is done through MID screen, voice, music, head LEDs, and motion using motors. Combination of various input and output allows Genie to perform the required services and also makes programming possible.



Robot Power But

Genie will feel and react to head stroke or touch.
Genie will pause for certain period when head is touched during the automous mode movement.

Eye(LED)

- When Genie is being charged, color around the eyes will change according to the charge state.
- Charge complete : Blue Charging : Red
(Red line will grow as Genie becomes charged)

Ear(LED)

- Both ears are normally(autonomous/stop mode) lit with white LEDs (does not blink).
- LEDs will blink when in speech recognition mode.

Mouth(LED)

1~3 white LEDs will blink when Genie talks or when Genie's emotion changes.

(3) Hovis Genie Specification

Hovis Genie is about as tall as an adult's knee. Genie is capable of automatic charging and Genie can also be manipulated by remote control using the camera through the MID. Specifications as as follows.

	Item	Detailed Description
Robot	Model Name	HOVIS Genie
	Size	255(W) X 236.5(D) X 408.5(H)
	Weight	3.4kg
	Motor	11pcs
	Sensor	Obstacle detection sensor, cliff detection snsor, IR sensor, touch sensor Tactile sensor, localization sensor.
	Drivetrain	Omni-directional Drive / HerkuleX 0102 3ea
	Network	IR Receiver, ZigBee[option]
	External Speaker	External 2 way stereo Speaker(1W 2Ch)
	Battery Capacity	7.4V / 3,000mAh / Li-Po
MID	Display	3.5"TFT(480X320)
	Screen Input Method	Capacitive Full Touch
	Main Processor	MID : S5PC110, 1GHz
	Camera	Front Rear Camera
	Audio	1ch MIC(MID internal)
	Internal Memory	256MB
	External Memory	Micro SD 4G
	Network	Wi-Fi
Other	Remote Control	IR type / ZigBee Remocon[option]

1.4 HOVIS Genie Software

From the development perspective, Hovis Genie has two distinguishing software features. Firmware which go into the robot controller and the sensor controller board, and the application software for developing MID appliations. Firmware is an AVR program created using C language and MID application software is an Android program created using Java. Dongbu Robot provides API for robot service through Android engineers.

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

Above boards used for controlling robot motors and sensors are installed internally and at the back of the robot. MPSU and OPSU receives main commands from the MID and contain embedded firmware pograms. MPSU is also called DRC and it is identical to the DRC found in Hovis Lite. After receiving commands from the MID, MPSU operates the motor & head LEDs, MID Side Board, head touch sensor, and shoulder distance sensor. OPSU located internally near the abdomen provides values to the floor detection sensor, distance sensor, and charging station IR sensor. MID Side Board located on the chest below the MID controls the speaker, MID charging, and MID and MPSU communication level change. Both MPSU and OPSU contain ATmega128 microprocessor and AVR programming is used to create the firmware. Hovis Lite includes DRC (MPSU) programming method and instructions whereas modification of factory embedded firmware is restricted on Hovis Genie since Genie's main purpose is to implement service through the MID.

(2) MID – Android

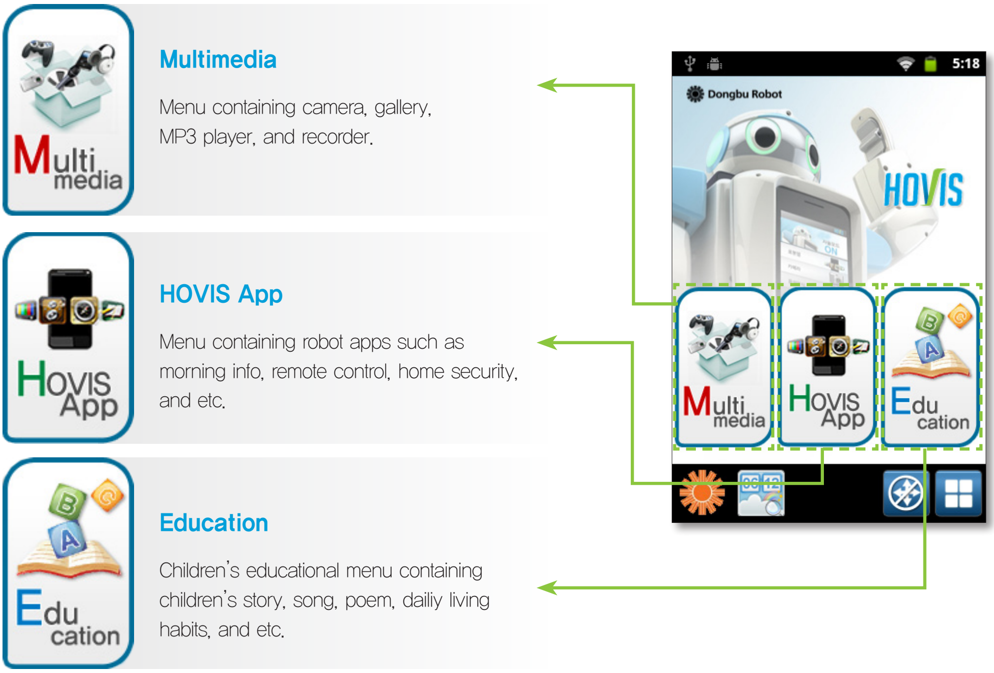
MID is abbrivation of Multimedia Internet Device. MID by Dongbu Robot with Android OS is able to take advantage of the Apps available in the Android market to download new services. While MID is similar in many ways to the the standard Android terminal, MID contains robot service program that is used to control Hovis Genie. MID is comprised of system software (firmware) and application. Android OS which is based on Linux forms the terminal firmware. While programming materials for the system software will be released by Dongbu Robot at a later date, ultimate objective of Genie software devlopment lies with Android MID applications. There are already hundres of thousands of Android applications created using JAVA language available for download in the Android market. In the same manner, Hovis Genie Android robot applications can be developed and downloaded to the MID to provide robot services.

1.5 Running HOVIS Genie Application

When Hovis Genie MID is first turned on, Screen will go directly to Hovis related main menu rather than standard Andoid terminal programs.

(1) Main Screen

When robot is first turned on, screen will first show picture of the robot heart with sound and then move directly to the main screen. Main menu screen is comprised of 3 main menus at the top and shortcut buttons, home button, and robot soft button at the bottom.



Multimedia

Menu containing camera, gallery, MP3 player, and recorder.

HOVIS App

Menu containing robot apps such as morning info, remote control, home security, and etc.

Education

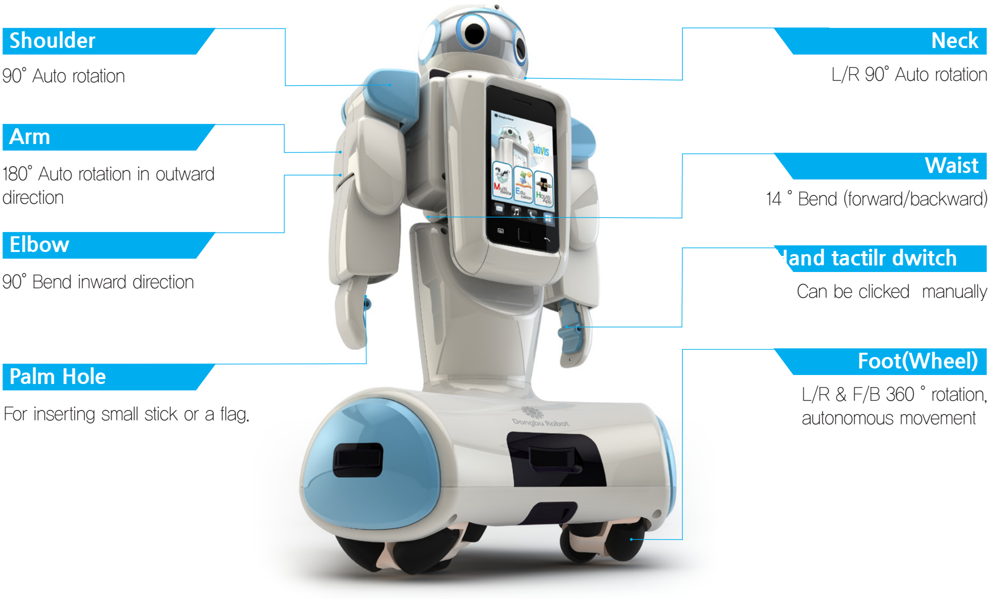
Children's educational menu containing children's story, song, poem, dailiy living habits, and etc.

Main menu is comprised of Multimedia menu containing camera, gallery, and sound related menus. Hovis App menu contains morning info, remote control, and other robot service App menus. Education menu contains children's stories, songs, poems, and other education related menus. Multimedia menu is collection of standard multimedia applications which are useful to the robot. Hovois App menu is collection of robot related applications which can make robot peform various functions such as motion. LED, touch, voice, and etc. Lastly, Hovis Genie was developed as home service robot and children's education is an important part of the service Genie is able to provide. There are many children's educational contents such as songs and stories which were created using Flash or other Apps available to the children.

Hovis Apps and educational content apps are available for download from the Dongbu Robot MID App store as well as other applications and contents. Also, previously down-loaded applications can be updated using the App store when update becomes available. App store is currently not 100% open to the public but limited App store for Dongbu robot is operational.

(2) Autonomous Operation

People tend to imagine about C-3PO or terminator robots seen in the movies when thinkng about robots. Unfortunately or fortunately, such robots are not possible to build with current state of our technology. The most widely used and practical robots in the world today are industrial robots which are very good at performing repetitive tasks with precision but these robots are more like factory automation equipment rather than the robot that we imagine. Hovis Genie is a humanoid robot similar to C-3PO seen in the Star Wars movies. Genie may not be as practical as the industrial robots but home service Genie can provide greatly increases the practical aspect of Genie comapred to the previous generation service robots. One of the most sought-after function for the robot developers world wide is the development of fully independent robot such as C-3PO seen in the movie which is able to operate fully independently without human instruction or intervention but such a robot does not exist today. Robots such as Genie can respond to external inputs from various sensors accoring to the preprogrammed response but autonomous operation of the robots today are still very limited.


Shoulder
90˚ Auto rotation
Arm
180˚ Auto rotation in outward direction
Elbow
90˚ Bend inward direction
Palm Hole
For inserting small stick or a flag.
Neck
L/R 90˚ Auto rotation
Waist
14 ˚ Bend (forward/backward)
Hand tactilr dwitch
Can be clicked manually
Foot(Wheel)
L/R & F/B 360 ˚ rotation, autonomous movement

1 Response to an obstacle

When Genie meets an obstacle during movement, Genie will turn and change direction.

2 Response to head touch

1 Genie will act pleased when touched softly on the head

2 Genie will show displeasure when hit on the head·

3 Response to hunger

Genie will begin automatic recharging when battery level falls below 20%.

4 Action by time

1 In the morning, Genie will ask about today's weather and say "have a nice day".

2 At lunch time, Genie will say "it's time for lunch" and ask if you have had lunch yet.

3 In the evening, Genie will ask how today's weather was and also ask what you did today. At 9PM, Genie will say "it's time for bed", and at 10PM, Genie will say "let's get ready for bed" and show motion simulating brushing teeth or washing face.

5 Autonomous Operation

Being bored, expressing love, playing cute, asking time, playing alone, stretching, scratching head, exercising, conducting, and other actions and voice expressions.

6 Response to click on the palm tactile switch

1 Left hand click – Genie will say 'Hi' and greet you or kiss you to show affection.

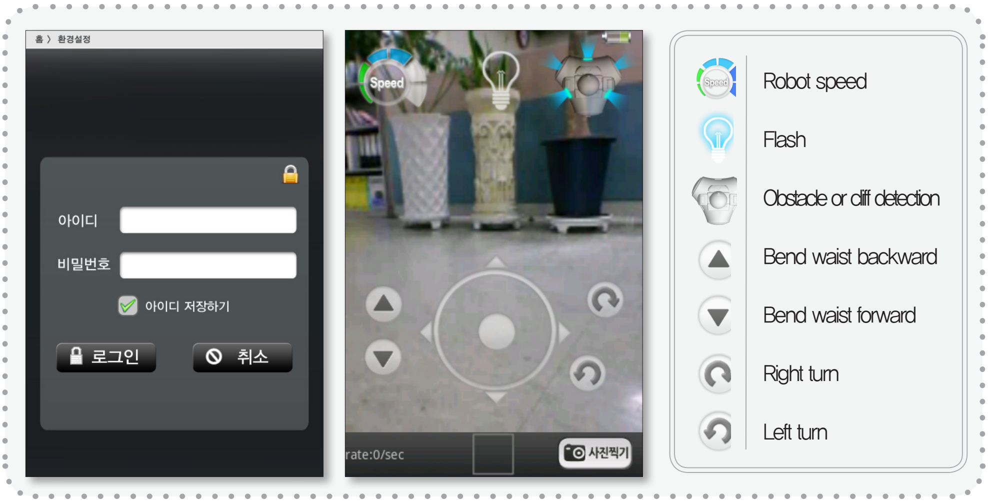
2 Right hand click – **Genie will simulate motion looking at the wrist watch and tell the** time and today's date.

3 **Both hands click** – Place light bouncy ball betewen Genie's hands and Genie will lift the ball above it's head and throw the ball.

When Genie meets an obstacle, lower front distance sensor responds and when Genie is touched on the head, head touch sensor responds. Genie will move towards the charging station when the battery level falls. Genie can also be made to act out diffrent motion patterns at specified times or made to dance. Autonomous operation will depend on how detailed the programmer is able to program Genie.

(3) Remote Control

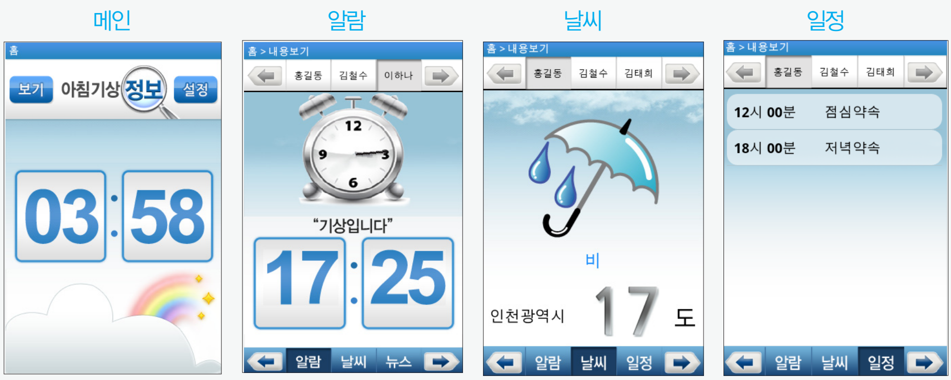
Remote control is one of the most poluplar method of robot control. Vacuum cleaner with camera can be used to monitor the home or be controlled from remote location or Mars exploration robots can be made to explore planet Mars through remote control. NASA Mars Rover Opportunity robot explorer sent to Mars in 2004 operated on Mars for 7 years using the camera on board the robot to send video data to Earth and receving instructions back from Earth. As radio signal travels at same speed as the speed of light, instructions sent to Mars took 4 minutes and 27 seconds to arrive. However, regardless whether the robot is a home vacuum cleaner or Mars explorer, basic method of remote control using the camera is identical.



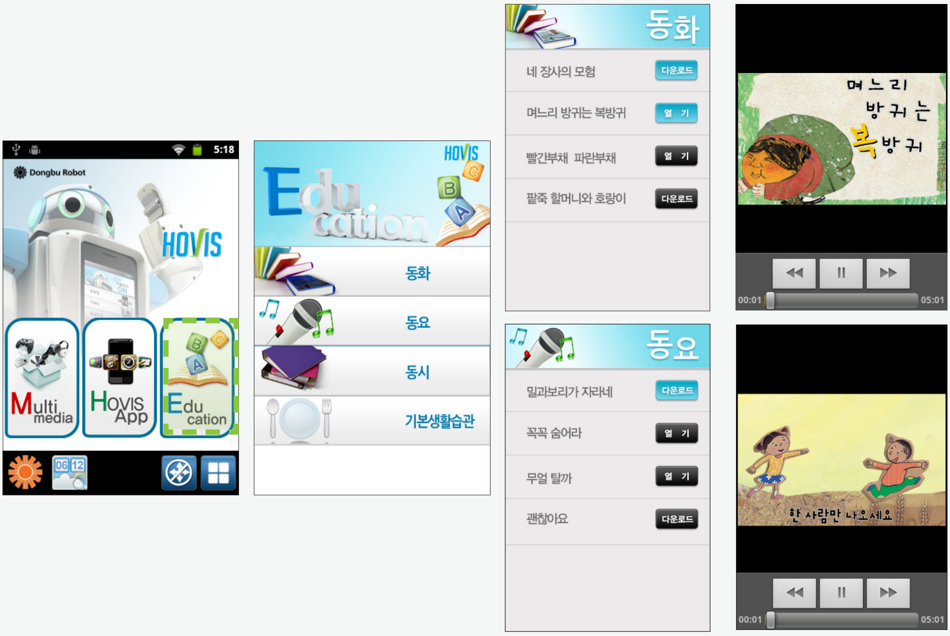
Hovis Genie can also be controlled remotely through the camera located at front of the MID. User must first install Hovis Genie control application on the smart phone and then login using registered ID and password to connect to Hovis Genie MID through the remote server. MID screen will show 'under remote control' once the connection is made. Hovis Genie movement can now be controlled using the movement related UI in the smart phone or robot motion saved in the MID can be run from remote location. Control application also has texting feature which can be used to send txt message to the MID to be read out aloud by Genie using TTS. Unlike other applications, remote control application requires two applications to work. Receiver application in the MID and the control application installed in the smart phone. Most of the currently existing applications are installed in the MID and must be programmed to connect to the robot system. However, control application installed in the smart phone can be developed in similar manner to other Android Apps. This allows application developers or enigneers to develope their own custom Apps to be installed in the smart phones to provide diverse remote control services such as motion control with music or voice output or robot movement control using the smart phone gyro sensor, and other services not found in the App provided by the manufacturer.

(4) Adapting Contents for Robot Service

Robot can be thought of as moving terminal. Our ultimate goal is to create an ecosystem where users will be able to pick and choose desired robot applications for their use from many different available applications. The best way of achieving this goal would be to adapt hundres of thousands of available Android Apps for use by the robot. Good example of this type of adaptation would be Genie's Morning Info application where existing smart phone Apps alarm, schedule, and weather are integrated with the robot motion to create an application adapted for robot use. Another good example of adapation for robot use would be children's stories or songs which are readily available. Simply adding robot motion to the existing story or song app would turn regular song or story into an interesting 3-D content which children would find much more interesting. The difference between regular Android content and robot content could be as simple as an addition of motion but the end result of such integration could result in much more interesting and useful product for the users. With participation of App developers and engineers we can expect to find many more interesting and useful applications to become available in the future.



[Morning Info Content]



[Children's story, song contents]

(5) Purpose & Application of Provided Software

By inclduing Android Geinie API, examples and educational material for each module with Hovi Genie, Dongbu Robot wish to provide an environment for Hovis Genie application development. While Hovis Genie was developed with home service in mind, our objective is also to provide support to the application developers and engineers to develope new applications and to adapt and existing Android applications for use with Hovis Genie. Hovis Genie robot service software can be divided into three types of software; autonomous operation, remote control, and information processing. Sample of each type of software will be supplied by Dongbu Robot for use as an example to develope new applications.

02 Creating Development Environment or Android & Robot

This chapter will describe creating development environment for Android & Robot.

Develpment environment for Android & Robot is divided into 4 following stages.

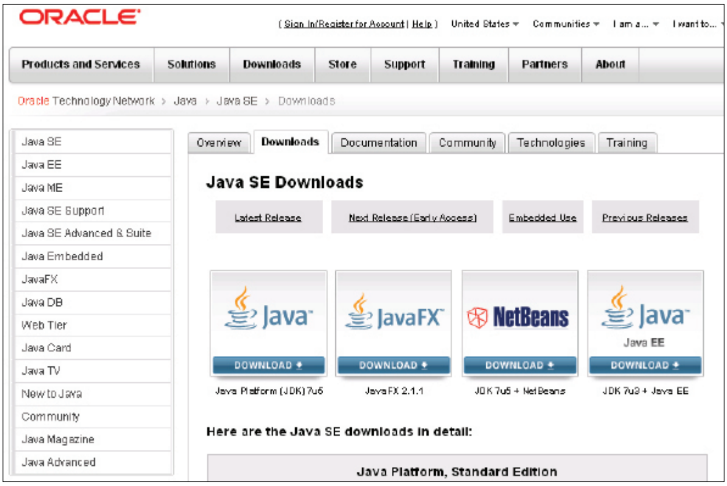
- JDK Installation
- Android SDK installation
- Eclipse installation and setup
- MID development board connection and debugging

2.1 JDK Installation

JDK(Java Developmenet Kit) is a Java development tool. As Android is based on Java, Java development tool is necessary, JDK has to be installed first as it affects the development enviornment.

01

Go to the Java site and click Java Download icon.
http://www.oracle.com/technetwork/java/javase/downloads/index.html



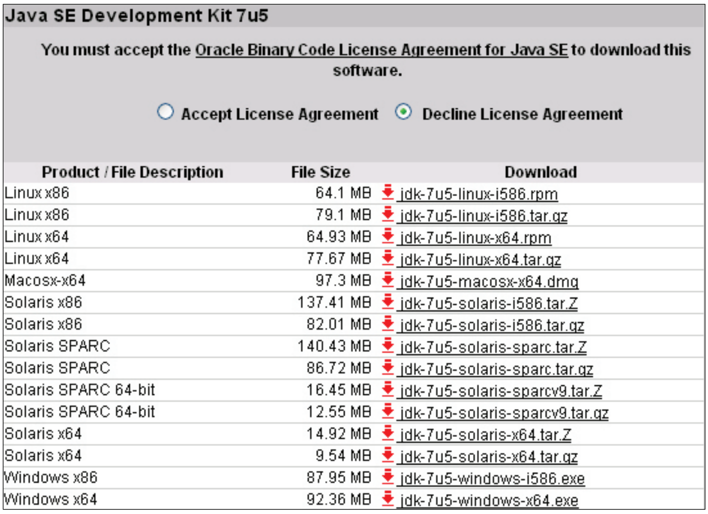
Java SE Downloads

Here are the Java SE downloads in detail:

Java Platform, Standard Edition

02

From the Java SE Development Kit section, accept the license agreement and download the JDK matching your OS.



Java SE Development Kit 7u5

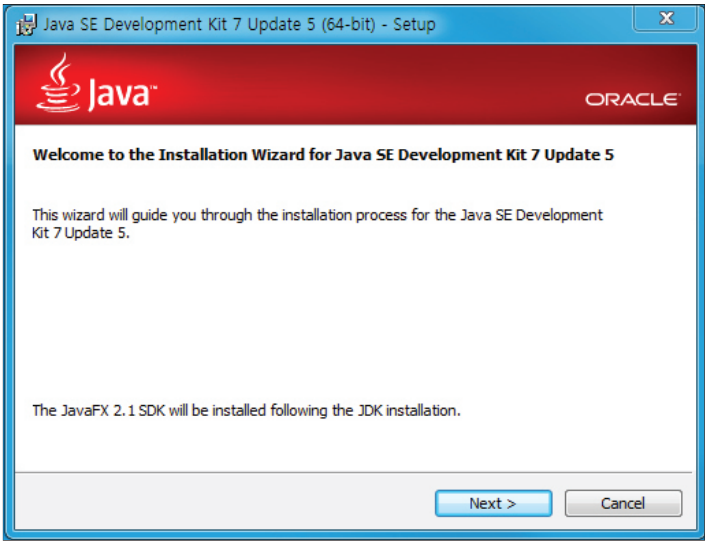
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ 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

Run the downloaded installation file to install JDK.



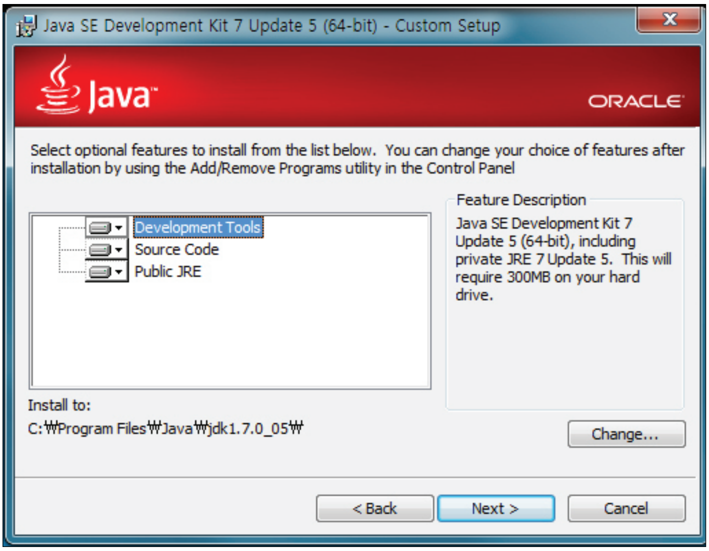
Welcome to the Installation Wizard for Java SE Development Kit 7 Update 5

This wizard will guide you through the installation process for the Java SE Development Kit 7 Update 5.

The JavaFX 2.1 SDK will be installed following the JDK installation.

04

Select JDK install location(Default destination is recommended. Do not select path with Korean language)



Java SE Development Kit 7 Update 5 (64-bit) - Custom Setup

Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel

- Development Tools
- Source Code
- Public JRE

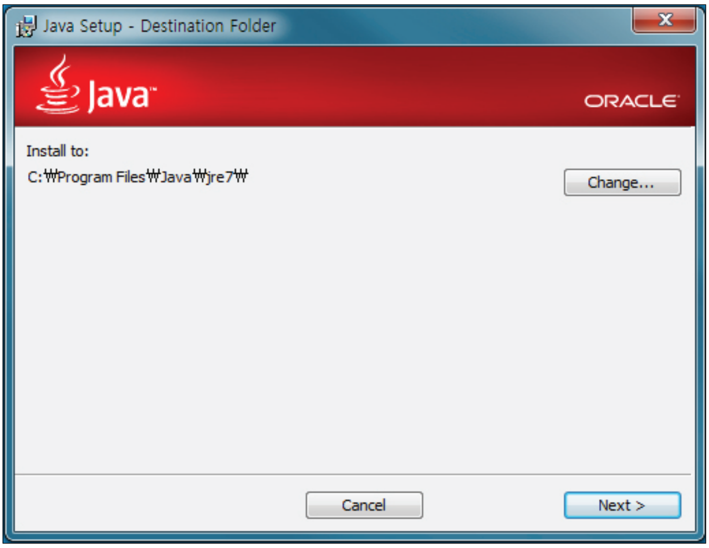
Feature Description

Java SE Development Kit 7 Update 5 (64-bit), including private JRE 7 Update 5. This will require 300MB on your hard drive.

Install to:
C:₩Program Files₩Java₩jdk1.7.0_05₩

05

Install JRE(Java Runtime Environment)

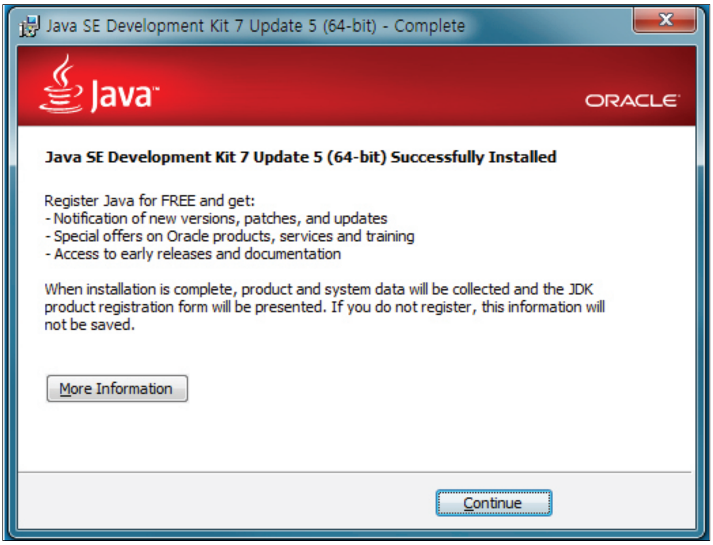


Java Setup - Destination Folder

Install to:
C:₩Program Files₩Java₩jre7₩

06

Installation complete
(JavaFX SDK does not have to be installed in the future)



Java SE Development Kit 7 Update 5 (64-bit) - Complete

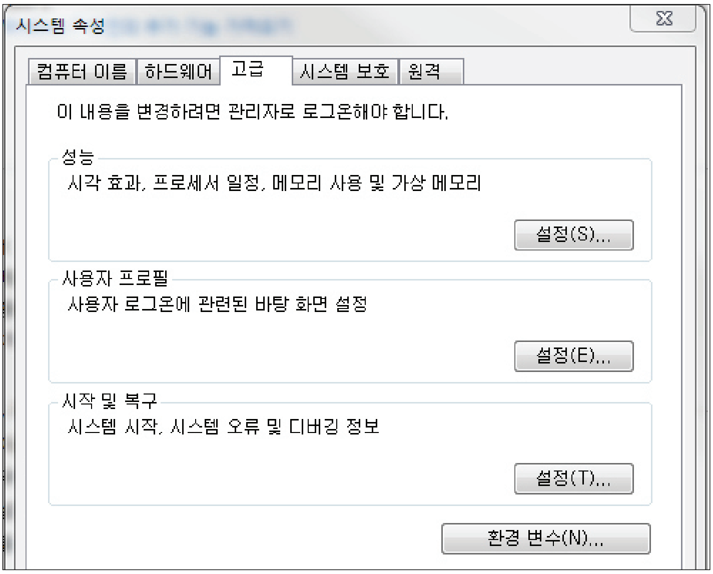
Java SE Development Kit 7 Update 5 (64-bit) Successfully Installed

Register Java for FREE and get:
- Notification of new versions, patches, and updates
- Special offers on Oracle products, services and training
- Access to early releases and documentation

When installation is complete, product and system data will be collected and the JDK product registration form will be presented. If you do not register, this information will not be saved.

07

Control panel 〉 system 〉 Advanced 〉 Click Environment variables(N)



시스템 속성

이 내용을 변경하려면 관리자로 로그온해야 합니다.

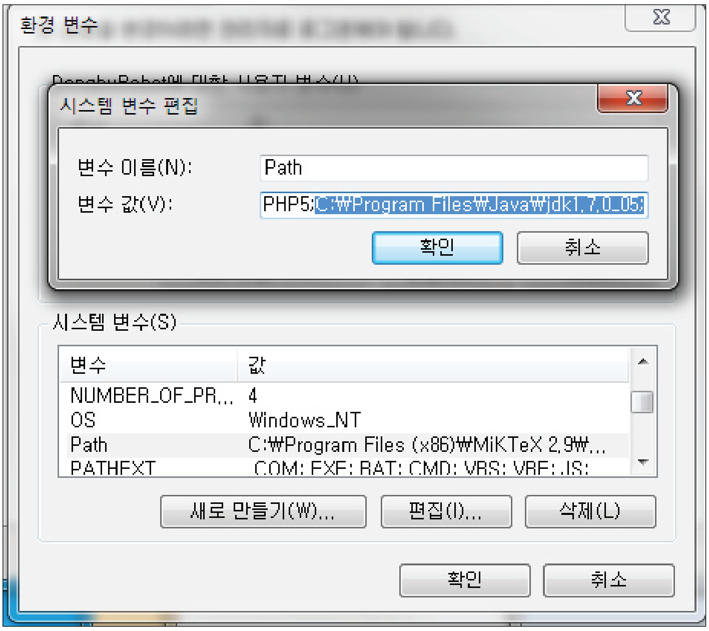
성능
시각 효과, 프로세서 일정, 메모리 사용 및 가상 메모리

사용자 프로필
사용자 로그온에 관련된 바탕 화면 설정

시작 및 복구
시스템 시작, 시스템 오류 및 디버깅 정보

08

Add path selected in step 4, except here C:₩Program Files₩Java₩jdk1.7.0_05₩



환경 변수

DongbuRobot에 대한 사용자 변수(U)

시스템 변수 편집

변수 이름(N): Path

변수 값(V): PHP5;C:₩Program Files₩Java₩jdk1.7.0_05;

시스템 변수(S)

변수	값
NUMBER_OF_PR...	4
OS	Windows_NT
Path	C:₩Program Files (x86)₩MiKTeX 2.9₩...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;

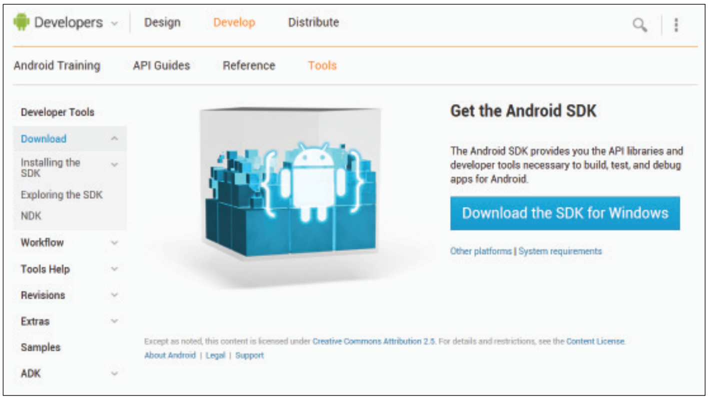
2.2 Android SDK Installation

Android SDK(Software Development Kit) is collection of libraries required for developing Android based applications. Android SDK can be downloaded from the official Android developers site http://developer.android.com/. Official Android developers site http://developer.android.com/ contains latest information, tutorials for beginners, API(Application Programing Interface) instructions and other inormation. As this manual is mainly concerned with Android robot programming, it is highly recommended for the user to visit the official Adroid developers site often for additional information regarding Android development.

Starting Android SDK. Refer to Android developers site and install the SDK following the instaructions below.

01

Go to Android developers site and download SDK (http://developer.android.com/sdk/index.html)



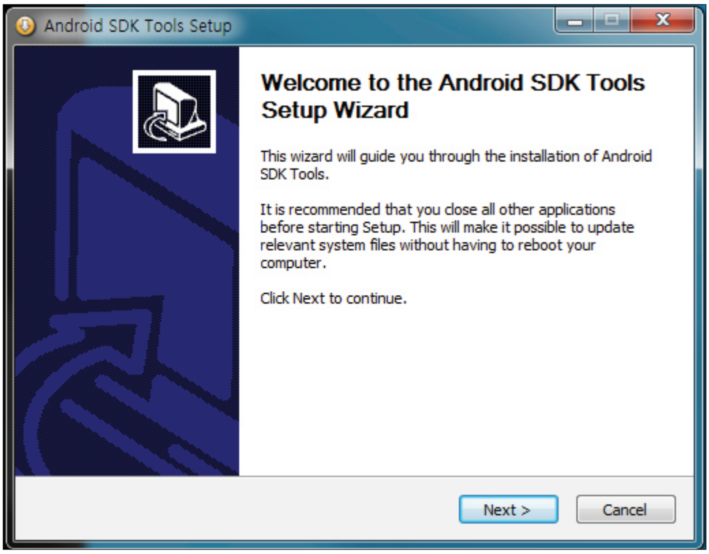
Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

Except as noted, this content is licensed under Creative Commons Attribution 2.5. For details and restrictions, see the Content License.

02

Run downloaded SDK installation file and click Next.



Welcome to the Android SDK Tools Setup Wizard

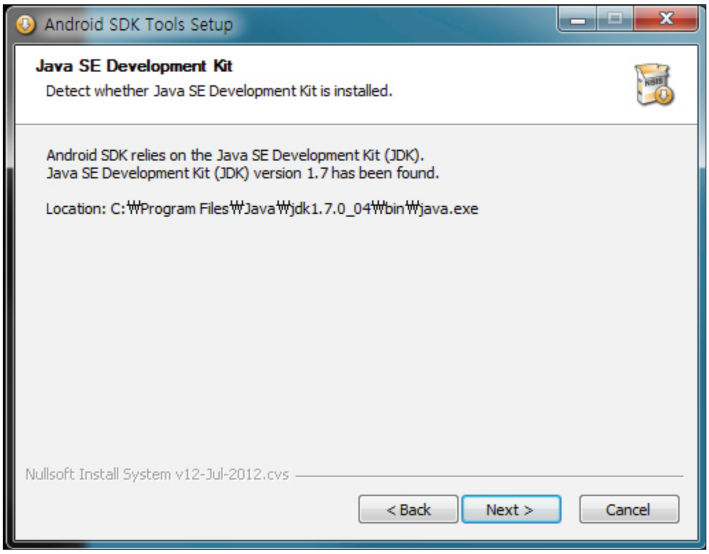
This wizard will guide you through the installation of Android SDK Tools.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

03

Installation will look for installed JDK. Click Next if JDK has been installed.
(Install JDK if it has not been installed)



Java SE Development Kit

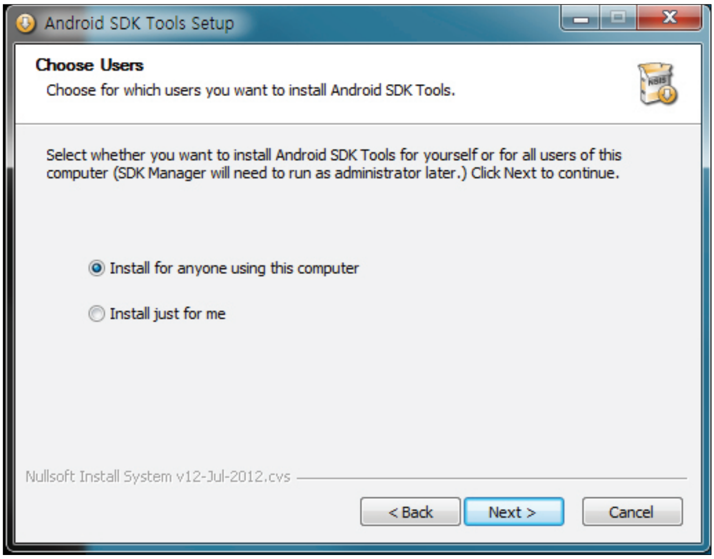
Detect whether Java SE Development Kit is installed.

Android SDK relies on the Java SE Development Kit (JDK).
Java SE Development Kit (JDK) version 1.7 has been found.

Location: C:₩Program Files₩Java₩jdk1.7.0_04₩bin₩java.exe

04

Select user option and click Next



Choose Users

Choose for which users you want to install Android SDK Tools.

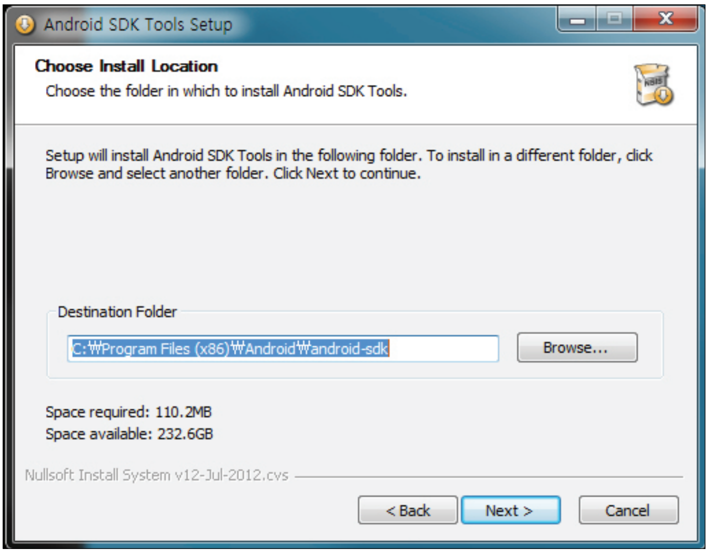
Select whether you want to install Android SDK Tools for yourself or for all users of this computer (SDK Manager will need to run as administrator later.) Click Next to continue.

● Install for anyone using this computer

○ Install just for me

05

Choose install location for Android SDK and click Next
(Default location recommended. Do not select path with Koreanlanguage)



Android SDK Tools Setup

Choose Install Location

Choose the folder in which to install Android SDK Tools.

Setup will install Android SDK Tools in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

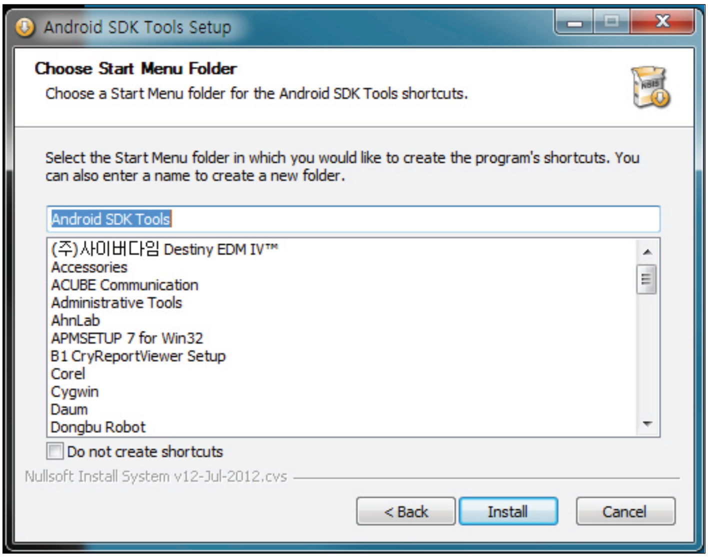
C:₩Program Files (x86)₩Android₩android-sdk

Space required: 110.2MB
Space available: 232.6GB

Nullsoft Install System v12-Jul-2012.cvs

06

Choose start menu folder and then click Next



Android SDK Tools Setup

Choose Start Menu Folder

Choose a Start Menu folder for the Android SDK Tools shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Android SDK Tools

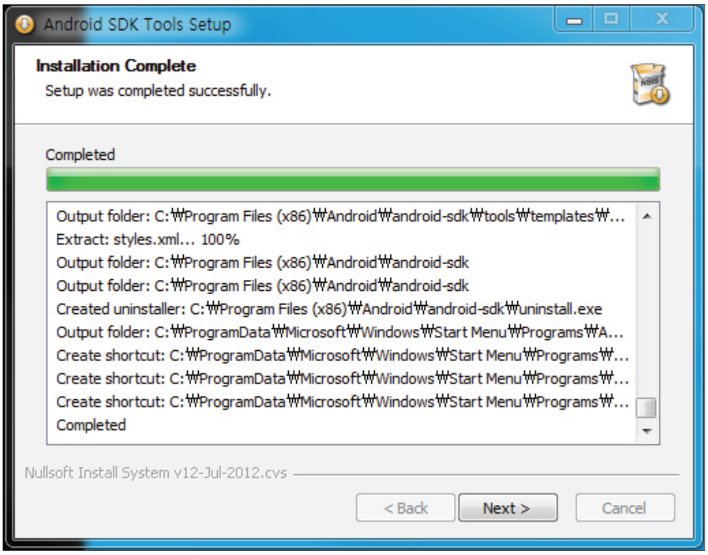
(주)사이버다임 Destiny EDM IV™
Accessories
ACUBE Communication
Administrative Tools
AhnLab
APMSETUP 7 for Win32
B1 CryReportViewer Setup
Corel
Cygwin
Daum
Dongbu Robot

□ Do not create shortcuts

Nullsoft Install System v12-Jul-2012.cvs

07

Click Next after installation is complete



Installation Complete

Setup was completed successfully.

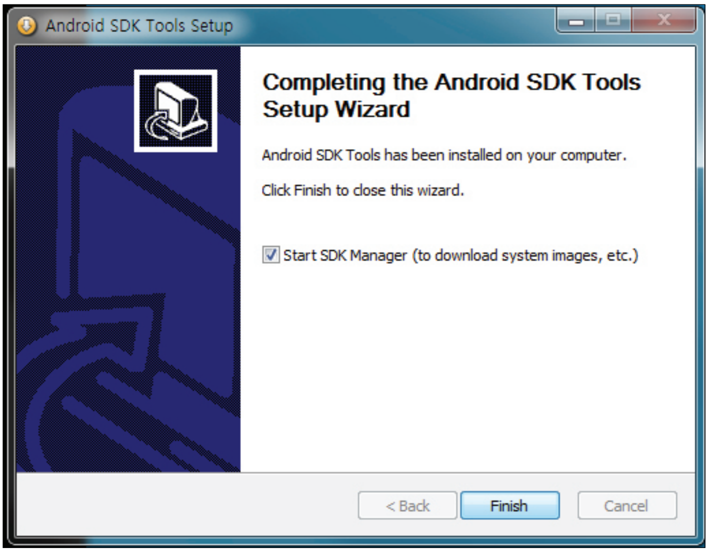
Completed

```
Output folder: C:₩Program Files (x86)₩Android₩android-sdk₩tools₩templates₩...
Extract: styles.xml... 100%
Output folder: C:₩Program Files (x86)₩Android₩android-sdk
Output folder: C:₩Program Files (x86)₩Android₩android-sdk
Created uninstaller: C:₩Program Files (x86)₩Android₩android-sdk₩uninstall.exe
Output folder: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩A...
Create shortcut: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩...
Create shortcut: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩...
Create shortcut: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩...
Completed
```

Nullsoft Install System v12-Jul-2012.cvs

08

Click Finish to run SDK Manager



Completing the Android SDK Tools Setup Wizard

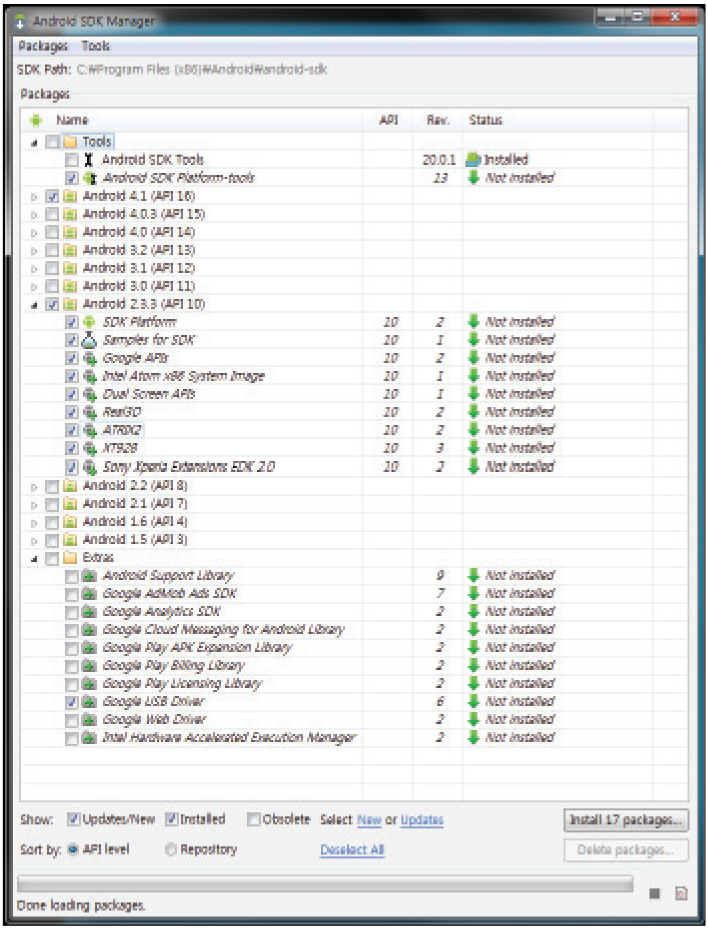
Android SDK Tools has been installed on your computer.

Click Finish to close this wizard.

☑ Start SDK Manager (to download system images, etc.)

09

Default selections are checked when SDK manager is run. Hovis Genie App develpment uses Android 2.3.3 (API 10). Check Android 2.3.3 and click Install.



SDK Path: C:₩Program Files (x86)₩Android₩android-sdk

Packages

Name	API	Rev.	Status
□ Tools			
□ Android SDK Tools		20.0.1	Installed
☑ Android SDK Platform-tools		13	Not installed
☑ Android 4.1 (API 16)			
□ Android 4.0.3 (API 15)			
□ Android 4.0 (API 14)			
□ Android 3.2 (API 13)			
□ Android 3.1 (API 12)			
□ Android 3.0 (API 11)			
☑ Android 2.3.3 (API 10)			
☑ SDK Platform	10	2	Not installed
☑ Samples for SDK	10	1	Not installed
☑ Google APIs	10	2	Not installed
☑ Intel Atom x86 System Image	10	1	Not installed
☑ Dual Screen APIs	10	1	Not installed
☑ Real3D	10	2	Not installed
☑ ATRIX2	10	2	Not installed
☑ XT928	10	3	Not installed
☑ Sony Xperia Extensions EDK 2.0	10	2	Not installed
□ Android 2.2 (API 8)			
□ Android 2.1 (API 7)			
□ Android 1.6 (API 4)			
□ Android 1.5 (API 3)			
□ Extras			
□ Android Support Library		9	Not installed
□ Google AdMob Ads SDK		7	Not installed
□ Google Analytics SDK		2	Not installed
□ Google Cloud Messaging for Android Library		2	Not installed
□ Google Play APK Expansion Library		2	Not installed
□ Google Play Billing Library		2	Not installed
□ Google Play Licensing Library		2	Not installed
☑ Google USB Driver		6	Not installed
□ Google Web Driver		2	Not installed
□ Intel Hardware Accelerated Execution Manager		2	Not installed

Show: ☑ Updates/New ☑ Installed □ Obsolete Select New or Updates

Sort by: ● API level ○ Repository Deselect All

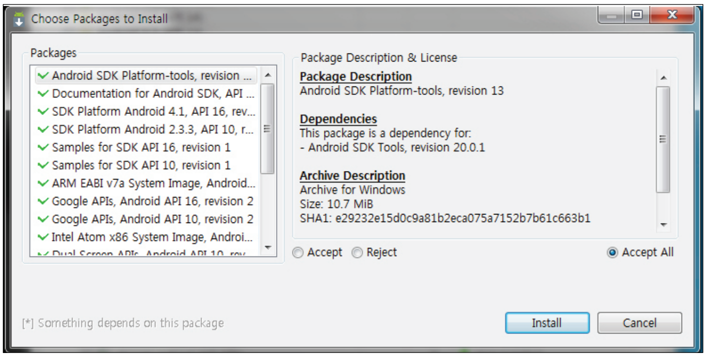
Install 17 packages...

Delete packages...

Done loading packages.

10

Accept license agreement and click install.(Installation takes about 1~3 hrs)



Packages

- ✓ Android SDK Platform-tools, revision ...
- ✓ Documentation for Android SDK, API ...
- ✓ SDK Platform Android 4.1, API 16, rev...
- ✓ SDK Platform Android 2.3.3, API 10, r...
- ✓ Samples for SDK API 16, revision 1
- ✓ Samples for SDK API 10, revision 1
- ✓ ARM EABI v7a System Image, Android...
- ✓ Google APIs, Android API 16, revision 2
- ✓ Google APIs, Android API 10, revision 2
- ✓ Intel Atom x86 System Image, Androi...
- ✓ Dual Screen APIs, Android API 10, rev...

Package Description & License

Package Description
Android SDK Platform-tools, revision 13

Dependencies
This package is a dependency for:
- Android SDK Tools, revision 20.0.1

Archive Description
Archive for Windows
Size: 10.7 MiB
SHA1: e29232e15d0c9a81b2eca075a7152b7b61c663b1

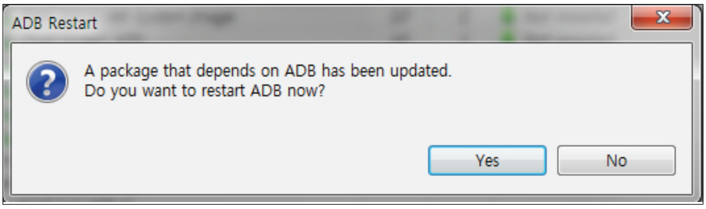
○ Accept ○ Reject ● Accept All

[*] Something depends on this package

Install Cancel

11

After installation is complete, click Yes to restart ADB and close SDK manager.



ADB Restart

A package that depends on ADB has been updated.
Do you want to restart ADB now?

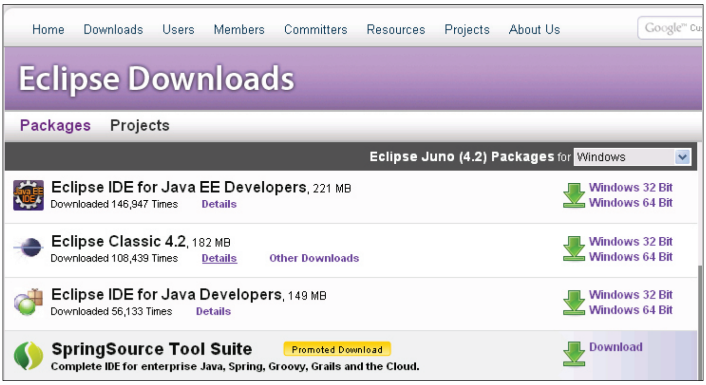
ADB (Android Debug Bridge) is a versatile command line tool that lets you communicate with an emulator instance or connected Android–powered device.

2.3 Eclipse Installation & Setup

There are many tool available for Android development and one of the most popular and well known tool is Eclipse which is used by the majority if not most of Android developers . Eclipse is an open source based integrated development environment providing comprehensive set of tools to simplify coding, debugging, and execution. Installing Eclipse is slighty different from installing other software. Like installing other Java based open source project software, downloaded file has to be decompressed to complete the installation. After Eclipse installation is complete, ADT Plugin has to be installed to add Android development support to Eclipse. ADT (Android Development Tools) are set of tools for developing Android applications quickly and easily.

01

Go to Eclipse site and select Eclipse IDE for Java EE Developers for your OS.
(http://www.eclipse.org/downloads/)
* Download Eclipse 3.6.2(Helios) or later version



Eclipse IDE for Java EE Developers, 221 MB
Downloaded 146,947 Times Details
Windows 32 Bit
Windows 64 Bit

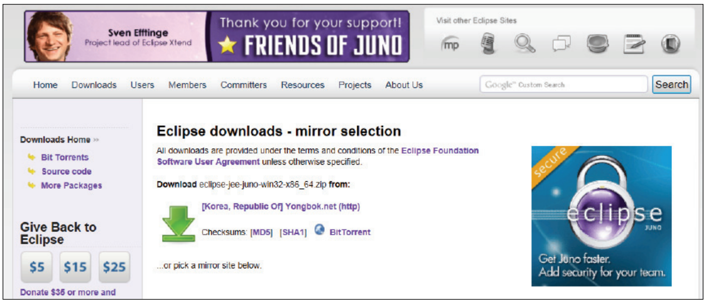
Eclipse Classic 4.2, 182 MB
Downloaded 108,439 Times Details Other Downloads
Windows 32 Bit
Windows 64 Bit

Eclipse IDE for Java Developers, 149 MB
Downloaded 56,133 Times Details
Windows 32 Bit
Windows 64 Bit

SpringSource Tool Suite Promoted Download
Complete IDE for enterprise Java, Spring, Groovy, Grails and the Cloud.
Download

02

Select mirror site to download Eclipse



Eclipse downloads - mirror selection

All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

Download eclipse-jee-juno-win32-x86_64.zip from:

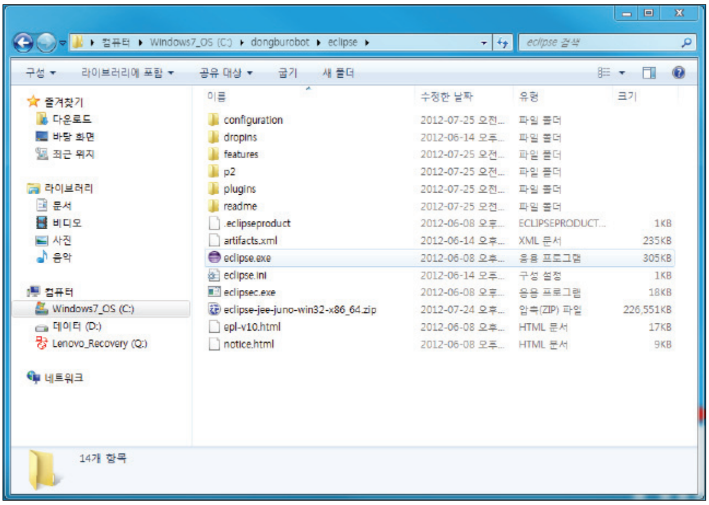
[Korea, Republic Of] Yongbok.net (http)

Checksums: [MD5] [SHA1] BitTorrent

...or pick a mirror site below.

03

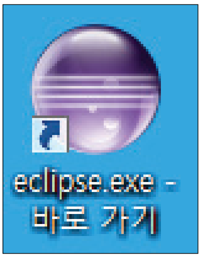
Decompres downloaded Eclipse (example C:₩dongburobot₩eclipse₩)
Warning : Do not decompress to path with Korean language.



이름	수정한 날짜	유형	크기
configuration	2012-07-25 오전...	파일 폴더	
dropins	2012-06-14 오후...	파일 폴더	
features	2012-07-25 오전...	파일 폴더	
p2	2012-07-25 오전...	파일 폴더	
plugins	2012-07-25 오전...	파일 폴더	
readme	2012-07-25 오전...	파일 폴더	
.eclipseproduct	2012-06-08 오후...	ECLIPSEPRODUCT...	1KB
artifacts.xml	2012-06-14 오후...	XML 문서	235KB
eclipse.exe	2012-06-08 오후...	응용 프로그램	305KB
eclipse.ini	2012-06-14 오후...	구성 설정	1KB
eclipsec.exe	2012-06-08 오후...	응용 프로그램	18KB
eclipse-jee-juno-win32-x86_64.zip	2012-07-24 오후...	압축(ZIP) 파일	226,551KB
epl-v10.html	2012-06-08 오후...	HTML 문서	17KB
notice.html	2012-06-08 오후...	HTML 문서	9KB

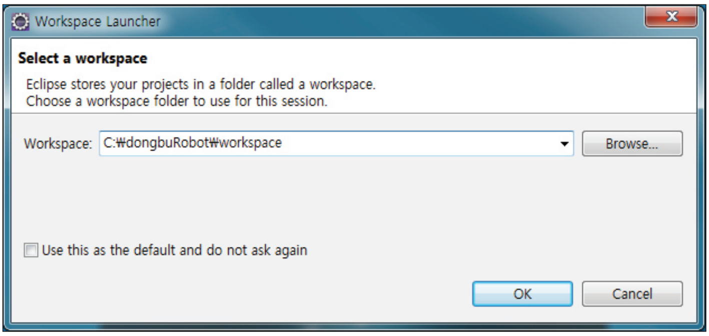
04

Create short cut icon on the desk top and run Eclipse.



05

Select Workspace and click OK.(Workspace refers to space for editing and saving source codes. Sample C:₩dongburobot₩workspace₩)



Workspace Launcher

Select a workspace

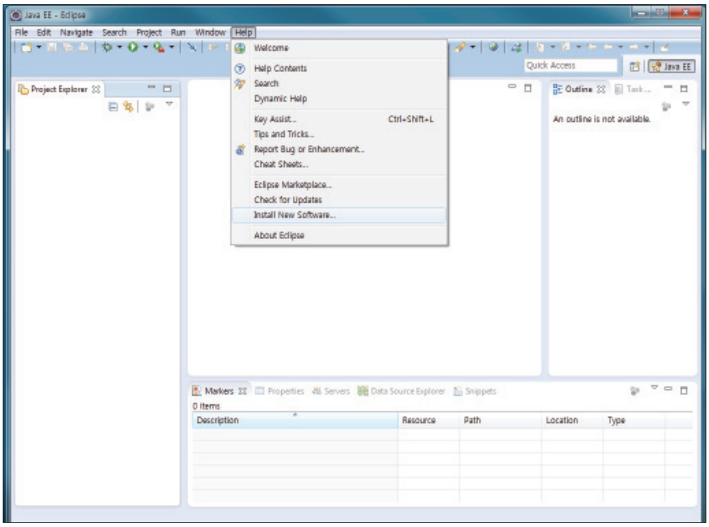
Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:₩dongbuRobot₩workspace

□ Use this as the default and do not ask again

06

Freom Eclipse select Help 〉Install New Softwar.

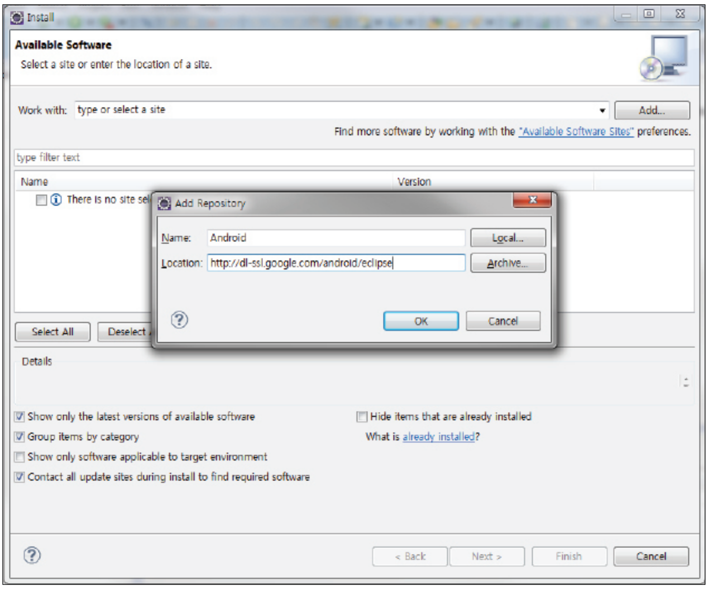


07

Click Add and add the following to the Add Repository window.

Name : Android

Location : http://dl-ssl.google.com/android/eclipse/



Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
□ There is no site sel	

Add Repository

Name: Android

Location: http://dl-ssl.google.com/android/eclipse

Details

☑ Show only the latest versions of available software

☑ Group items by category

□ Show only software applicable to target environment

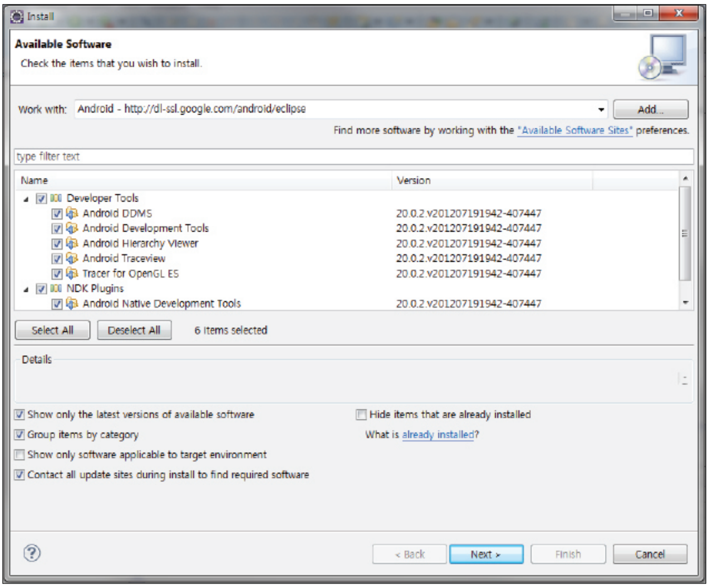
☑ Contact all update sites during install to find required software

□ Hide items that are already installed

What is already installed?

08

Following screen will appear when Rpository added in step 7 is selected. Select All and then click Next.



Install

Available Software

Check the items that you wish to install.

Work with: Android - http://dl-ssl.google.com/android/eclipse

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☑ Developer Tools	
☑ Android DDMS	20.0.2.v201207191942-407447
☑ Android Development Tools	20.0.2.v201207191942-407447
☑ Android Hierarchy Viewer	20.0.2.v201207191942-407447
☑ Android Traceview	20.0.2.v201207191942-407447
☑ Tracer for OpenGL ES	20.0.2.v201207191942-407447
☑ NDK Plugins	
☑ Android Native Development Tools	20.0.2.v201207191942-407447

6 items selected

Details

☑ Show only the latest versions of available software

☑ Group items by category

□ Show only software applicable to target environment

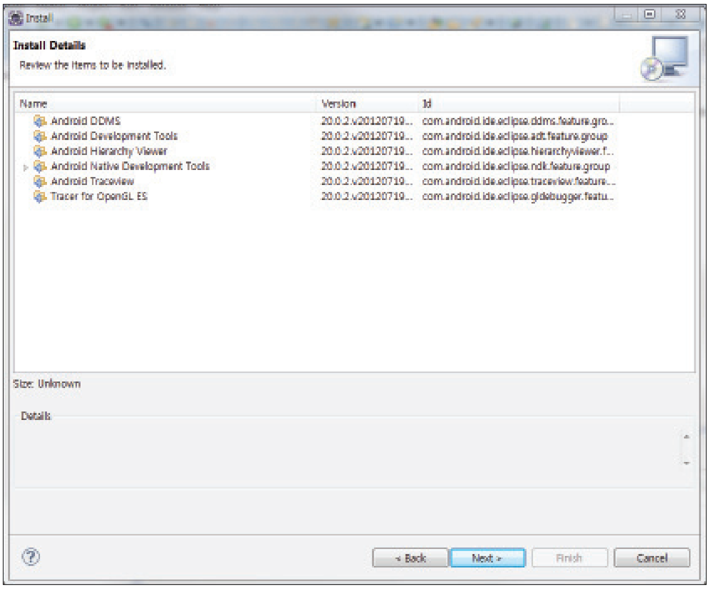
☑ Contact all update sites during install to find required software

□ Hide items that are already installed

What is already installed?

09

Check Eclipse Plugin list and click Next.



Install Details

Review the items to be installed.

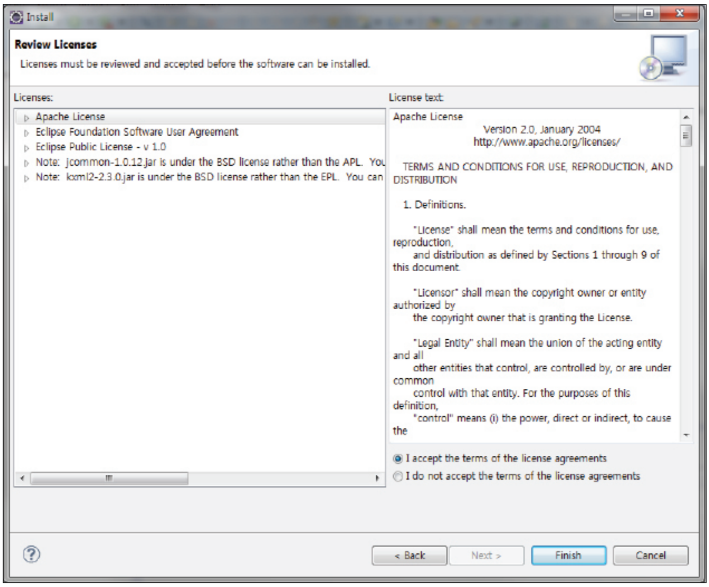
Name	Version	Id
Android DDMS	20.0.2.v20120719...	com.android.ide.eclipse.ddms.feature.gro...
Android Development Tools	20.0.2.v20120719...	com.android.ide.eclipse.adt.feature.group
Android Hierarchy Viewer	20.0.2.v20120719...	com.android.ide.eclipse.hierarchyviewer.f...
Android Native Development Tools	20.0.2.v20120719...	com.android.ide.eclipse.ndk.feature.group
Android Traceview	20.0.2.v20120719...	com.android.ide.eclipse.traceview.feature...
Tracer for OpenGL ES	20.0.2.v20120719...	com.android.ide.eclipse.gldebugger.featu...

Size: Unknown

Details

10

Accept license agreement and click Finish to start installation.



Review Licenses

Licenses must be reviewed and accepted before the software can be installed.

Licenses:

- Apache License
- Eclipse Foundation Software User Agreement
- Eclipse Public License - v 1.0
- Note: jcommon-1.0.12.jar is under the BSD license rather than the APL. You
- Note: kxml2-2.3.0.jar is under the BSD license rather than the EPL. You can

License text:

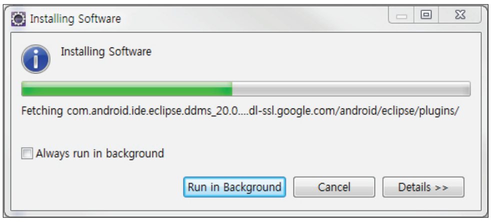
```
Apache License
                 Version 2.0, January 2004
              http://www.apache.org/licenses/

  TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND
DISTRIBUTION

  1. Definitions.

     "License" shall mean the terms and conditions for use,
reproduction,
     and distribution as defined by Sections 1 through 9 of
this document.

     "Licensor" shall mean the copyright owner or entity
authorized by
     the copyright owner that is granting the License.

     "Legal Entity" shall mean the union of the acting entity
and all
     other entities that control, are controlled by, or are under
common
     control with that entity. For the purposes of this
definition,
     "control" means (i) the power, direct or indirect, to cause
the
```

● I accept the terms of the license agreements

○ I do not accept the terms of the license agreements

11

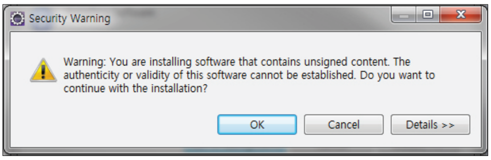
Following security warning may appear during installation. Click OK to continue.



Installing Software

Fetching com.android.ide.eclipse.ddms_20.0....dl-ssl.google.com/android/eclipse/plugins/

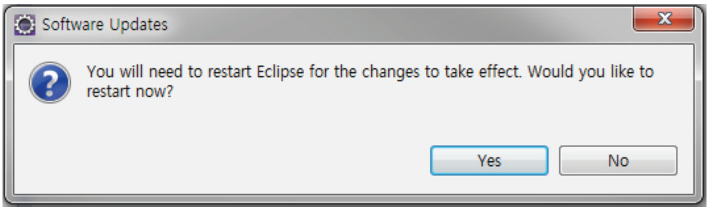
□ Always run in background



Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?

12

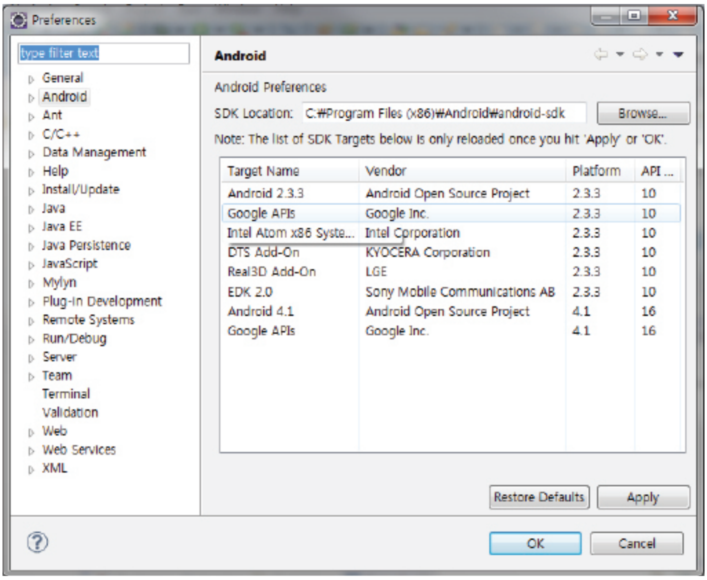
Restart Eclipse after completing Plugin installation.



You will need to restart Eclipse for the changes to take effect. Would you like to restart now?

13

From Eclispse, run Window 〉 Preferences and set SDK location in Android tab. SDK Location is the SDK installation path set in 2.2. Clip Apply button and then click OK to finish setup.



Android Preferences

SDK Location: C:₩Program Files (x86)₩Android₩android-sdk

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

Target Name	Vendor	Platform	API ...
Android 2.3.3	Android Open Source Project	2.3.3	10
Google APIs	Google Inc.	2.3.3	10
Intel Atom x86 Syste...	Intel Corporation	2.3.3	10
DTS Add-On	KYOCERA Corporation	2.3.3	10
Real3D Add-On	LGE	2.3.3	10
EDK 2.0	Sony Mobile Communications AB	2.3.3	10
Android 4.1	Android Open Source Project	4.1	16
Google APIs	Google Inc.	4.1	16

03 Android Robot (Genie) Programming

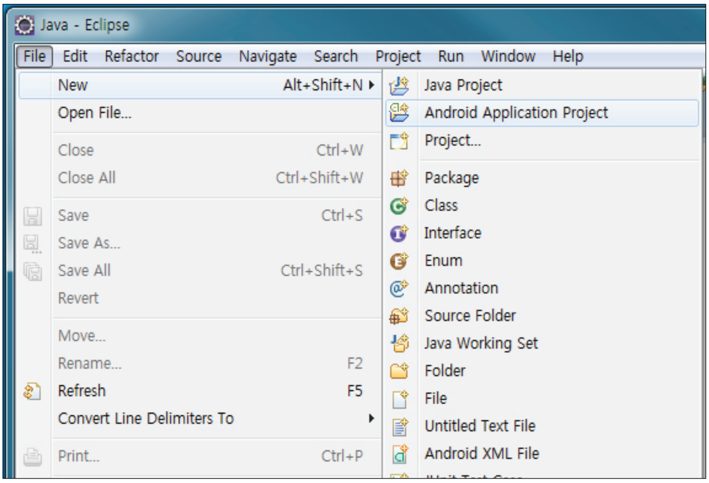
Development environment for Andriod programming was set up in Chapter 2. In this chapter, we will start creating actual Android based program.

Objective of this chapter is to create actual Android App that will work in the Android platform. In order to accomplish our objective, understanding of Android is required. We will learn the basic structure and features of Android while working our first App. However, due to the complexity of Android programming which is beyond the scope of this manual, we recommend you visit Android developers site (http://developer.android.com/index.html) frequently for extra information. Even though this manual cannot cover the whole of Andorid programming, examples provided will help you to understand the basic concepts invovled in working with Android.

3.1 Hello Genie Project Creation

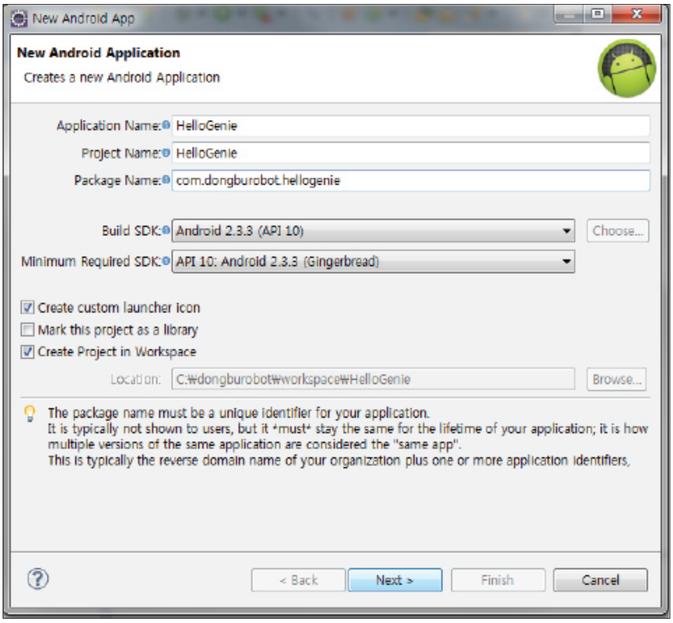
01

Run Eclipse, select File〉New〉Android Application Project.



02

Set App, project and package name. Build SDK must be set to Android 2.3.3 (API 10) since MID installed in Hovis Geine has Android version 2.3.3. Click Next after all names have been set.



New Android App

New Android Application

Creates a new Android Application

Application Name: HelloGenie

Project Name: HelloGenie

Package Name: com.dongburobot.hellogenie

Build SDK: Android 2.3.3 (API 10)

Minimum Required SDK: API 10: Android 2.3.3 (Gingerbread)

☑ Create custom launcher icon

□ Mark this project as a library

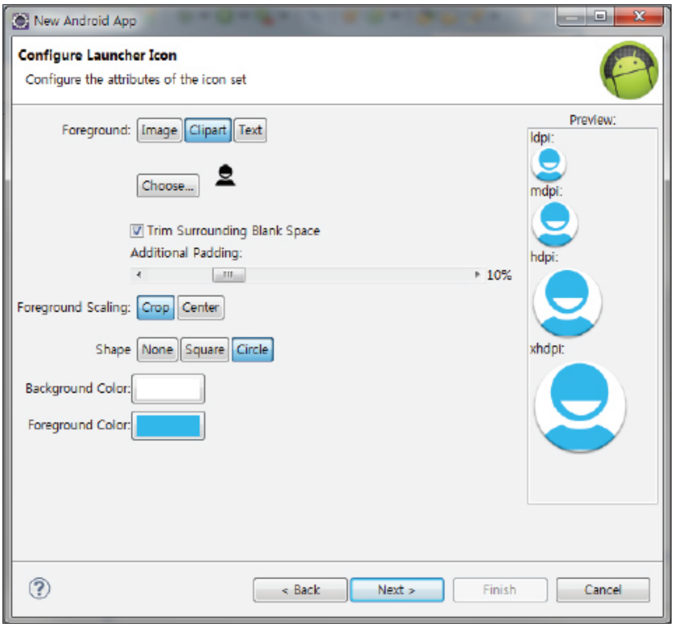
☑ Create Project in Workspace

Location: C:₩dongburobot₩workspace₩HelloGenie

The package name must be a unique identifier for your application.
It is typically not shown to users, but it *must* stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app".
This is typically the reverse domain name of your organization plus one or more application identifiers.

03

Select launch icon. Launch icon is a square icon representing the App in the smart phone. Click Next.



New Android App

Configure Launcher Icon

Configure the attributes of the icon set

Foreground: Image [✓ Clipart] Text

☑ Trim Surrounding Blank Space

Additional Padding: 10%

Foreground Scaling: [✓ Crop] Center

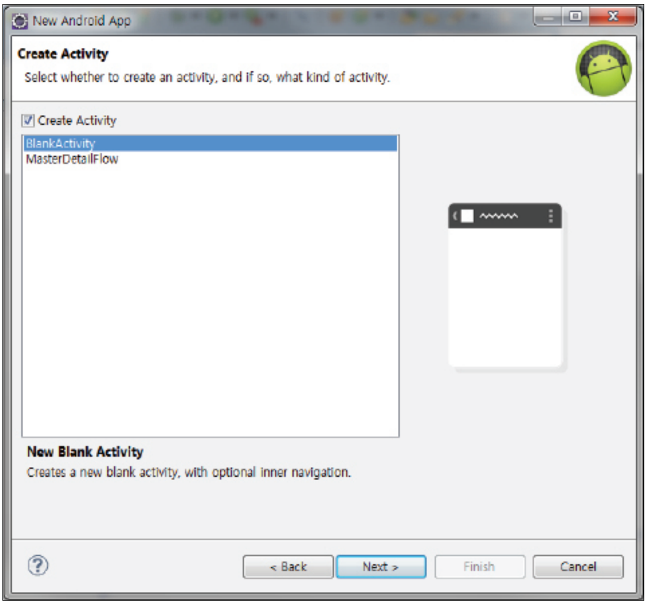
Shape: None Square [✓ Circle]

Background Color:

Foreground Color:

04

Create Activity. Activity refers to a single screen in the smart phone. For example, if you receive a call while you are in the midst of sending a txt message and the screen switches to the phone, one activity has been switched to another activity. We will discuss the Activity more in detail later.Select BlankActivity and click Next. (MasterDetailFlow is a newly added function for use in Tablet and will not be dealt with in this manual.)



New Android App

Create Activity

Select whether to create an activity, and if so, what kind of activity.

☑ Create Activity

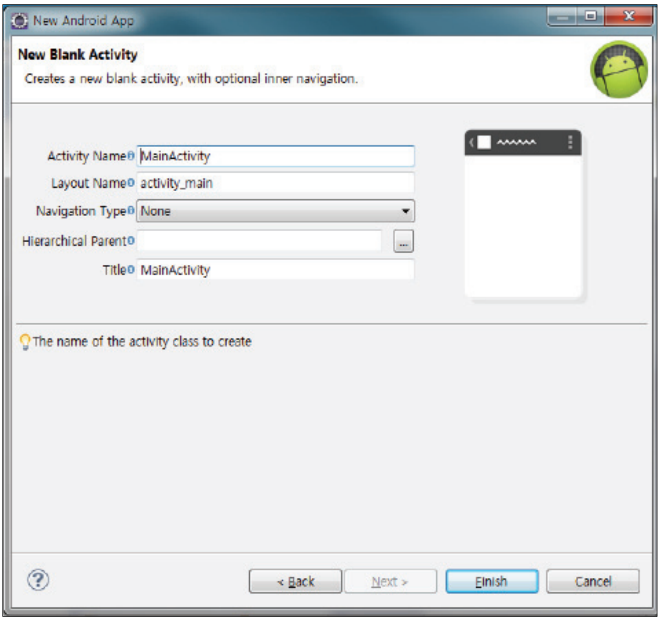
BlankActivity
MasterDetailFlow

New Blank Activity

Creates a new blank activity, with optional inner navigation.

05

Select name for the created BlankActivity. Default name will be used. Click Finish to complete new project creation.



New Android App

New Blank Activity

Creates a new blank activity, with optional inner navigation.

Activity Name: MainActivity
Layout Name: activity_main
Navigation Type: None
Hierarchical Parent:
Title: MainActivity

The name of the activity class to create

3.2 Running Hello Genie Project

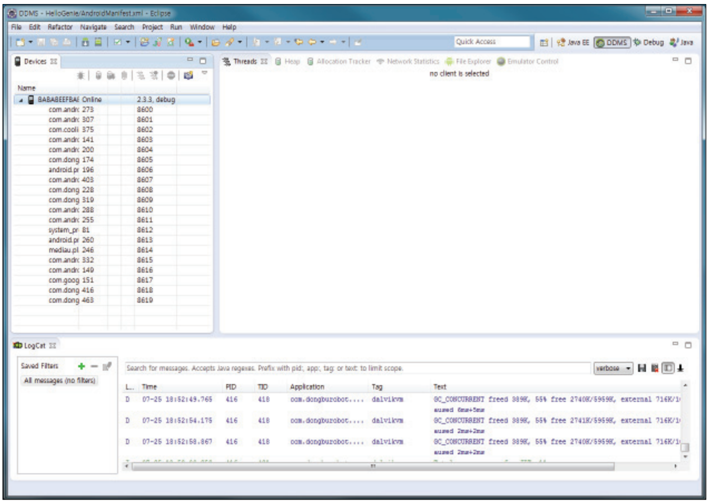
We will now run HelloGenie project. There are two ways to run the HelloGenie project created by Eclispse. Project could be run in the emulator or in the actual Andirod equpment. We will use the actual Android equipment MID to run the project.

01

USe USB jack to connect MID to PC

02

Use Eclipse DDMS to check the connection and to select the equipment.


DDMS - HelloGenie/AndroidManifest.xml - Eclipse
File Edit Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
DDMS
Debug
Java
Devices
Threads
Heap
Allocation Tracker
Network Statistics
File Explorer
Emulator Control
no client is selected
LogCat
Saved Filters
All messages (no filters)
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose

Name		
BABABEEFBAE Online		2.3.3, debug
com.andr	273	8600
com.andr	307	8601
com.cooli	375	8602
com.andr	141	8603
com.andr	200	8604
com.dong	174	8605
android.pr	196	8606
com.andr	403	8607
com.dong	228	8608
com.dong	319	8609
com.andr	288	8610
com.andr	255	8611
system_pr	81	8612
android.pr	260	8613
mediau.pl	246	8614
com.andr	332	8615
com.andr	149	8616
com.goog	151	8617
com.dong	416	8618
com.dong	463	8619

```
L..  Time                PID   TID   Application          Tag        Text
D    07-25 18:52:49.765  416   418   com.dongburobot....  dalvikvm   GC_CONCURRENT freed 389K, 55% free 2740K/5959K, external 716K/1
                                                                     paused 6ms+5ms
D    07-25 18:52:54.175  416   418   com.dongburobot....  dalvikvm   GC_CONCURRENT freed 389K, 55% free 2741K/5959K, external 716K/1
                                                                     paused 2ms+2ms
D    07-25 18:52:58.867  416   418   com.dongburobot....  dalvikvm   GC_CONCURRENT freed 389K, 55% free 2740K/5959K, external 716K/1
                                                                     paused 2ms+2ms
```

03

Press ▶ from the Eclipse tool bar to run the project and check the MID screen,

[TIP] If MID can not be found?

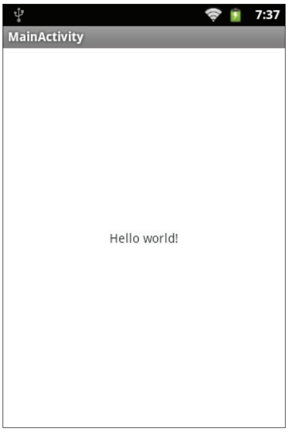
[Settings > Applications > Developer > USB debugging] Cancel USB debuggin and check again.

[TIP] If App does not run?

App will not run if xml source code is selected when button is clicked. In this case, *.xml.out file will be created. Delete this file and open any java file in /src and press the run button.

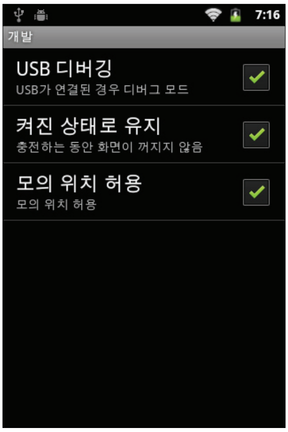
[TIP] If following error message appears?

Parser exception for /HelloGenie/ AndroidManifest.xml: In the text, markup follwoing the root element must be in correct format. It looks to be ADT Plugin bug. Refer to the source codes in the next section and edit the codes.



MainActivity

Hello world!



개발

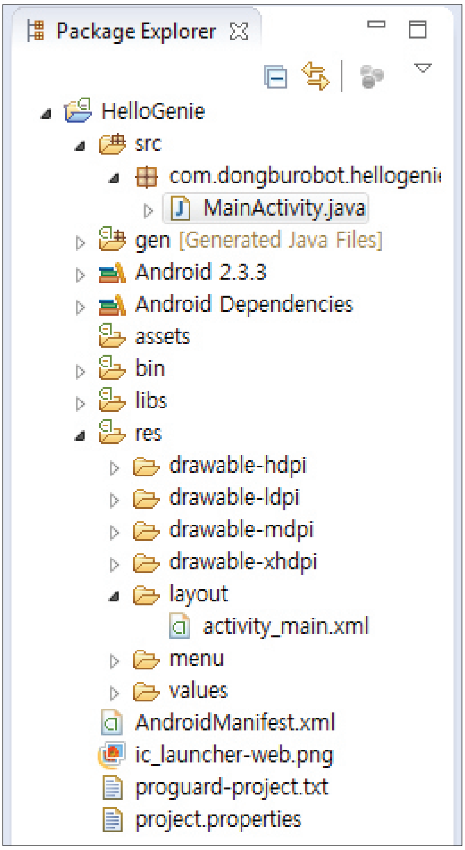
USB 디버깅 ☑
USB가 연결된 경우 디버그 모드

켜진 상태로 유지 ☑
충전하는 동안 화면이 꺼지지 않음

모의 위치 허용 ☑
모의 위치 허용

3.3 Hello Genie Project Basic Structure

We created and ran our first Android App HelloGenie in the actual Android equipment MID. We will now check the HelloGenie project files. Basic structure of HelloGenie can be found in the Eclispse Package Explorer window.



```
Package Explorer
HelloGenie
  src
    com.dongburobot.hellogeni
      MainActivity.java
  gen [Generated Java Files]
  Android 2.3.3
  Android Dependencies
  assets
  bin
  libs
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    layout
      activity_main.xml
    menu
    values
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

/src

Directory containing App source. Currently, MainActivity selected when creating the project is included as default.

/gen

Files in this directory are created automatically. Files contain data related to resounce and environment.

/bin

Directory containing files that were created during project build.

/libs

Directory containing external libraries that is included in the project. In the future, this directory will be used to include the libaries providing robot services.

/res

Directory containing project resources.

/res/drawable

Directory containing project image files.

/res/layout

Directory containing files defining screen UI(User Interface). One Activity represents one screen.One Activity will use one xml file contained in this directory.

/res/menu

Directory containing xml file defining the screen shown when menu button is pressed in the Andriod smart phone.

/res/values

Directory containing xml files designating other resources (text string or style).

AndroidManifest.xml

This file defines the project component structure and basic properties.
Each project has one AndroidMnaifest.xml file.

HelloGenie project structure contains many directories. Among the directories, understanding of /src, /res, and AndroidManifest.xml file is required to understand App operation.

(1) AndroidManifest.xml

Looking at AndroidManifest.xml file first. Each App includes single AndroidMaifest.xml file. AndroidManifest.xml file defines App component structure, App rights and properties. In other words, this file contains essential information about the App. AndroidManifest.xml file is shown below.

```
 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

〈manifest〉 tag is the higest tag of the AndroidManifest.xml file. This tag contains the basic information about the App. Package in the 2nd line contains the package name for the application. Lines 3~4, android:versionCode, android:versionName is for the version number and name of the App. Thes two properties can change when APP is upgraded.

〈uses–sdk〉 and 〈application〉 tags are found below 〈manifest〉 tag. 〈uses–sdk〉 tag designates operational environment of the App. android:minSdkVersion in 7th line designates minimum Android version App can operate in. android:targetSdkVersion in 8th line designates the API level of the App. Set this line to API Level 10 which is for the Android 2.3.3 Gingerbread version. MID installed in the robot contains Android version 2.3.3.

〈application〉 tag declares App components. Android has four core App components; Activity, Service, Broadcast receiver, and Content Provider (Refe to the website for detailed information on components). Since current HelloGenie project is composed on only one Activity, AdroidManifest.xml inlcudes one 〈activity〉 tag.

android:icon in the 11th line within 〈application〉 tag selects icons to show in the Android terminal. Depending on the resolution of the Android terminal, porperty value "@drawable/ic_launcher" refers to the image file ic_launcher in /res/drawable–hdpi/, … , /res/drawable–mdpi directory. (@ is a keyword that refers to existing resource). android:label in 12th line designates name of the App, property value "@string/app_name" refers to app_name text string designated in /res/values/strings.xml. 13 th line designates the App style. "@style/AppTheme" referst to the App Theme designated in /res/values/styles.xml.

Lines 14~21 decalres Activity contained in the project. android:name in line 15 designates the name of the class that implements the Activity. Name of the class for this Activity is com.dongburobot.hellogenie.MainActivity but since package name designated in the 〈manifest〉 tag can be skipped, value can be set to MainActivity. 16th line designates external name for the Activity.

Lines 17~20 contains 〈intent–filter〉. Intent filter specifies the types of intents Activity can respond to. Intent here refers to an object that contains necessary information to operate one of the core components. Intent is used to operate single or multiple core components. To be more specific, when HelloGeine App is run from the launcher (home screen), android.intent.action.MAIN Action and android.intent.category.LAUNCHER with Category property intent are sent to HelloGenie. Received intent will run the .MainActivity which has 2 filter properties.
AndroidManifest.xml file is an important file defining App component structure. Each App always includes one AndroidManifest.xml file. Looking at the HelloGenie App through the AndroidManifest.xml, 〈uses–sdk〉 tag shows that App runs in Android 2.3.3, and 〈application〉 tag shows one Activity is included. 〈activity〉 MainActivity is activated when intent which is generated when user clicks an icon from the launcher (home screen) is received by the intent filter.

(2) MainActivity.java and activity_main.xml

One Activity refers to one screen. User use the screen (Activity) to interact with the smart phone. In other words, since singe App is composed of multiple screens, it could be thought of as loose collection of multiple Activity.

When user runs an App, there is a sigle main Activity that is presented first. This Activity defined in the AndroidManifest.xml is MainActivity in our HelloGenie project.
To interact with the user, MainActivity contains activity_main.xml and MainActivity.java, We will examine the MainActivity.java file first.

```
package com.dongburobot.hellogenie;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

```

```
 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

All android Activity are created by inheriting android.app.Acitivity class. Our MainActivity class was also created by inheriting Activitiy class. When Activity class is inherited, many methods can be overridden depending on the Activity lifecycle. For your reference, Activity lifecycle is very important and must be understood properly. Refer to http://developer.android.com/guide/components/activities.html for more information. Lifecyclye will be discussed more in detail at later stage.

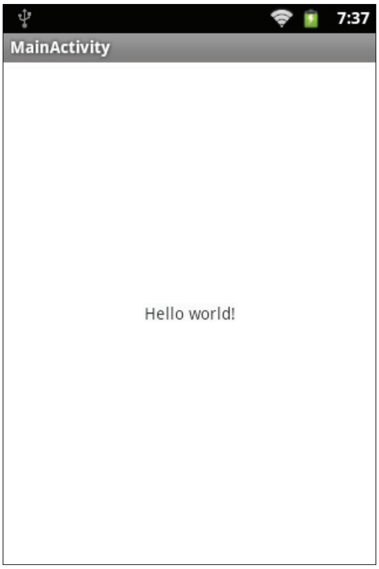
onCreate() method in 10th line is the first method to be executed when MainActivity is first generated. Following line is in the 12th line of onCreate().

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

setContentView() in this line assigns user interface defined in the activity_main.xml to the MainActivity. R.Layout.activity_main in setContentView() refers to res/layout/activity_main.xml.

```
〈RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    xmlns:tools=”http://schemas.android.com/tools”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent” 〉

    〈TextView
        android:layout_width=”wrap_content”
        android:layout_height=”wrap_content”
        android:layout_centerHorizontal=”true”
        android:layout_centerVertical=”true”
        android:text=”@string/hello_world” /〉
〈/RelativeLayout〉
```

Listing3. activity_main.xml

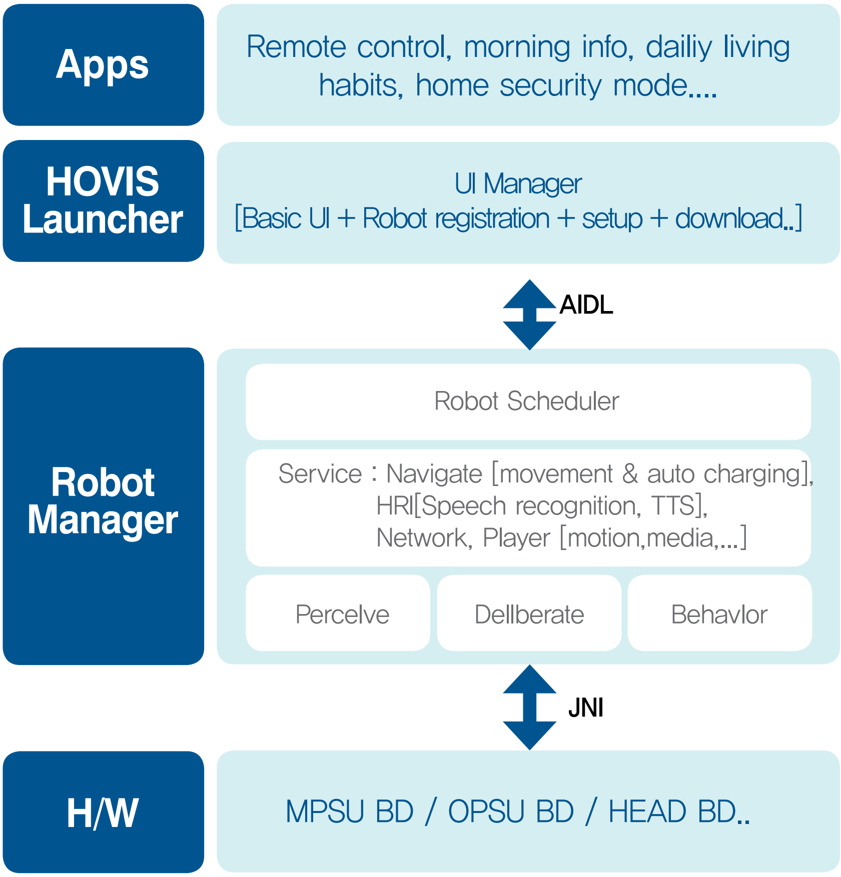


activity_main.xml is composed of RelativeLayout and lower element TextView. Relative Layout is a layout that places multiple views relative to each other. TextView is a view that ouputs text to the screen. Picture on the left shows the actual user interface defined in the Activitiy_main.xml. XML layout file will be discussed more in detail later. Refer to http://developer.android.com/guide/topics/ui/declaring-layout.html for more information.

04 Starting HOVIS Genie Robot Programming

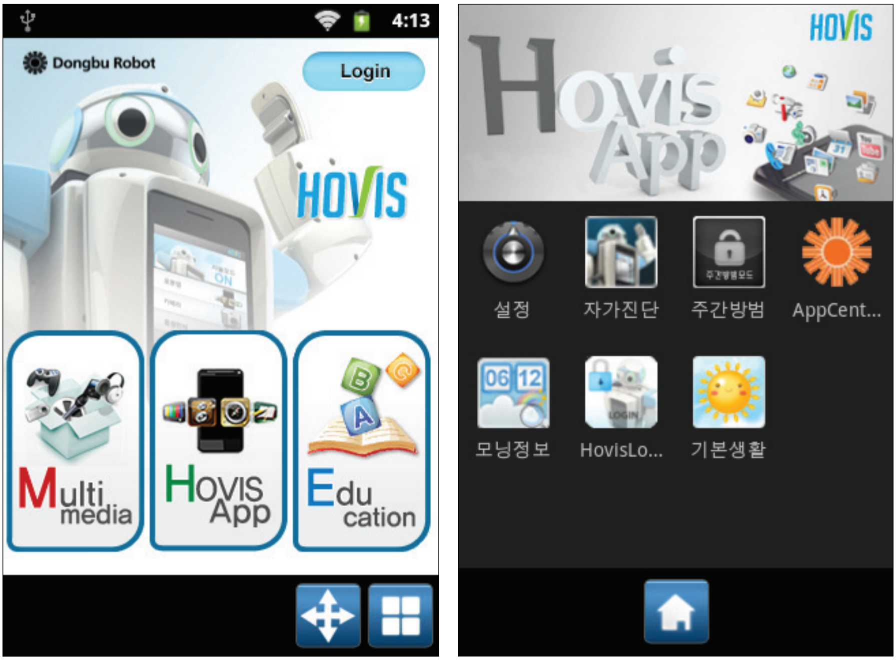
We created Android programming environment in Chapter 2. In Chapter 3, we learned about the Android project structure while creating and and running our first Android App. In this chapter, we will take our first step in programming Hovis Genie .

Understanding of the Hovis Genie S/W structure is required before starting Hovis Genie programming. Through the understanding of robot operation and by using the available services, it becomes possible to develope robot Apps. [Diagram 4-1] shows the overall structure of Hovis Genie S/W within the Android system.


Apps
Remote control, morning info, dailiy living habits, home security mode....
HOVIS Launcher
UI Manager
[Basic UI + Robot registration + setup + download..]
AIDL
Robot Manager
Robot Scheduler
Service : Navigate [movement & auto charging], HRI[Speech recognition, TTS], Network, Player [motion,media,...]
Percelve
Dellberate
Behavlor
JNI
H/W
MPSU BD / OPSU BD / HEAD BD..

[Diagram 4-1] Hovis Genie S/W

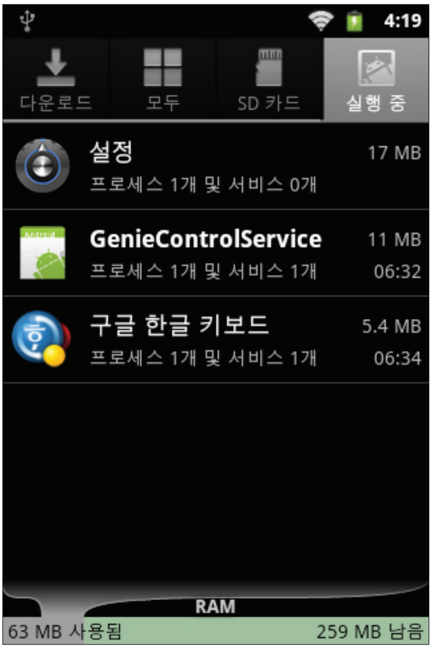
Hovis Genie S/W structure within the Android system can be divided into application programs Apps, Hovis Launcher, and Robot Manager. Among the 3, upper level application programs Apps and Hovis Launcher refers to robot Apps and home screen and can be easily recognized as shown in [Diagram 4–2].



[Diagram 4–2] Hovis Launcher and Robot App

Robot Manager cannot be seen as it runs internally within the system but it contains Robot Scheduler and service which plays an important part of robot operation. Robot Scheduler is responsible for taking care of robot events and depending on the robot status, autonomous mode, charging mode, and other modes, robot scheduler decides and executes couse of action. Service abstracts robot sensors, motors, and other hardware and provides robot service so that they can be easily used.

Service also provides API (Application Programing Interface) so that robot App we create can easily control the robot.Hovis Launcher (home screen) and robot App above the Robot Manager uses the Robot Manger service to execute robot related tasks.



설정 17 MB
프로세스 1개 및 서비스 0개

GenieControlService 11 MB
프로세스 1개 및 서비스 1개 06:32

구글 한글 키보드 5.4 MB
프로세스 1개 및 서비스 1개 06:34

63 MB 사용됨 259 MB 남음

[Diagram 4–3] Robot Manager level service is GenieControlService

As shown in [Diagram 4–3] Service runs under the name GenieControlService in the Android. GenieControlService [Diagram 4–1] forms the Robot Manager level and includes Robot Scheduler and Service. GenieControlService which includes im–portant modules is required for robot control and always runs in the background during robot operation. GenieControlService is able to restore itself when problem occurs due to unexpected system error. To summarize, GenieControlService is re–quired to access robot H/W from the App. In this chapter, we will look at the ways in which App we create accesses the robot service GenieControlService. Correct understanding of this chapter is required as example used in this chapter leads to robot programming using the Service in Chapter 5.

4.1 GenieApiDemo Project Creation

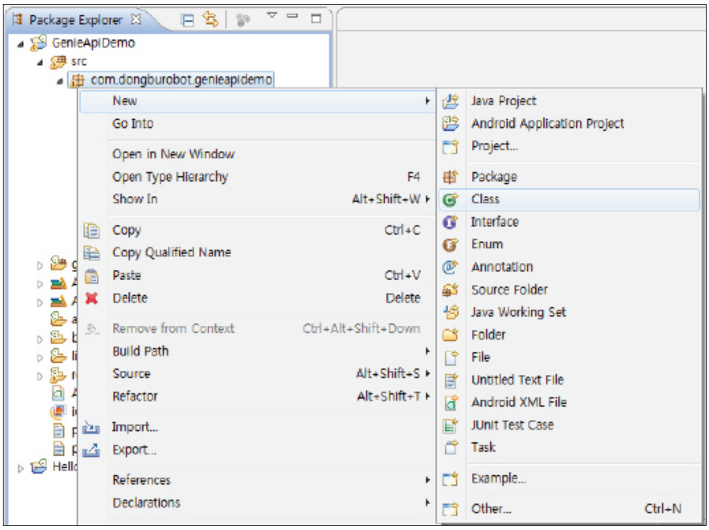
New project will be created for the example in ths chapter. From Eclipse, run the following window File 〉 New 〉 Android Application Project and enter the following to creat new project. (Refer to Chapter 3)

- 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)

After GenieApiDemo project is created, add GenieApiDemoApplication.java and BaseActivity.java. Also, to use GenieControlService, add HovisService.jar to the libs folder.

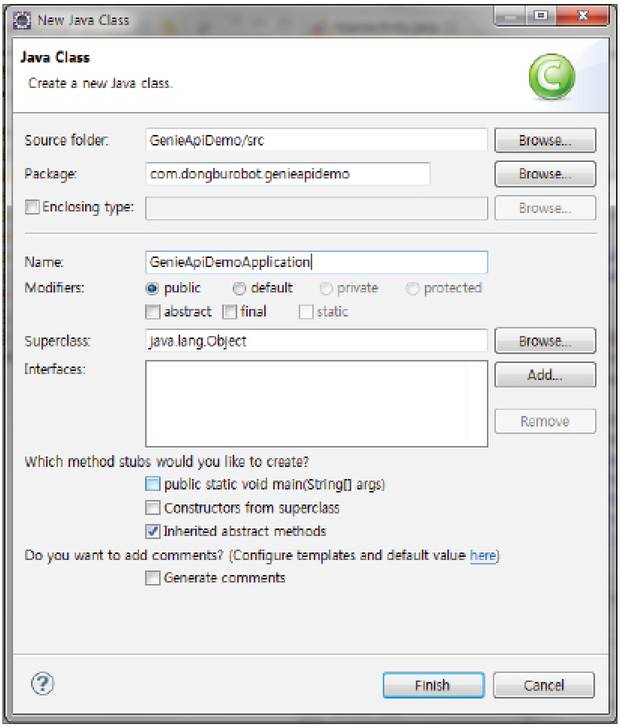
01

Right click from src/com.dongburobot.genieapi and then click New 〉 Class



02

Add GenieApiDemoApplication.java.



Java Class

Create a new Java class.

Source folder: GenieApiDemo/src [Browse...]

Package: com.dongburobot.genieapidemo [Browse...]

□ Enclosing type: [Browse...]

Name: GenieApiDemoApplication

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object [Browse...]

Interfaces: [Add...] [Remove]

Which method stubs would you like to create?

□ public static void main(String[] args)

□ Constructors from superclass

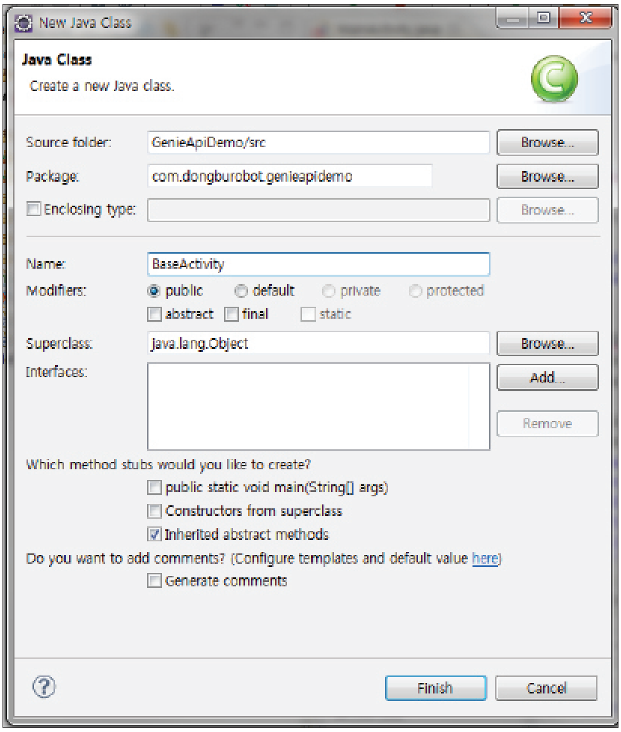
☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

03

Add BaseActivity.java using same method.



Java Class

Create a new Java class.

Source folder: GenieApiDemo/src [Browse...]

Package: com.dongburobot.genieapidemo [Browse...]

□ Enclosing type: [Browse...]

Name: BaseActivity

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object [Browse...]

Interfaces: [Add...] [Remove]

Which method stubs would you like to create?

□ public static void main(String[] args)

□ Constructors from superclass

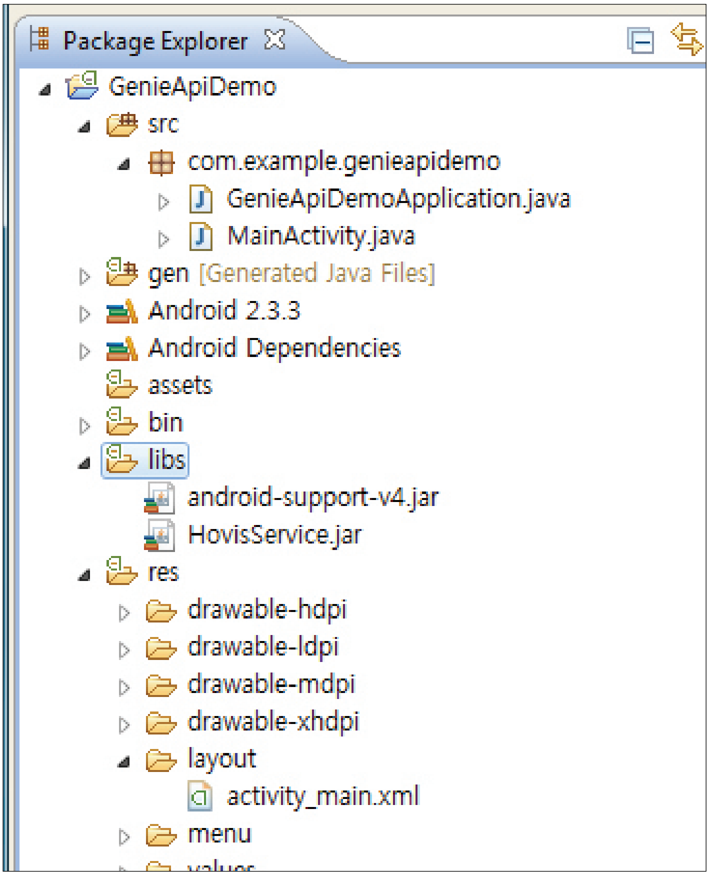
☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

04

Drag and add HovisService.jar file to libs folder. HovisService.jar can be downloaded from Dongbu Robot website.
(http://www.dongburobot.com/jsp/cms/view.jsp?code=100122)



```
Package Explorer ☒
◢ GenieApiDemo
    ◢ src
        ◢ com.example.genieapidemo
            ▷ GenieApiDemoApplication.java
            ▷ MainActivity.java
    ▷ gen [Generated Java Files]
    ▷ Android 2.3.3
    ▷ Android Dependencies
      assets
    ▷ bin
    ◢ libs
          android-support-v4.jar
          HovisService.jar
    ◢ res
        ▷ drawable-hdpi
        ▷ drawable-ldpi
        ▷ drawable-mdpi
        ▷ drawable-xhdpi
        ◢ layout
              activity_main.xml
        ▷ menu
```

HovisService.jar file in libs folder is a collection of robot related API declarations contained in GenieControlService distributed as single file. Make sure to include this file, otherwise project will not be built properly.

※Warning – Method for adding HovisService.jar maybe different depending on the version of the ADT(Android Development Tools) installed in Eclipse. Latest version of Eclipse and ADT Plugin is recommended.

4.2 GenieApiDemoApplication Class

GenieApiDemoApplication class is created by inheriting Application class. Application class is a class defined in android.app.Application class that expresses Android App as abstraction. Application class is used when overall state needs to be maintained while the App is running. In other words, it is useful for delcaring global variable which can be used for the whole App . Function of GenieApiDemoApplication in our GenieApiDemo project is to declare global global variable to bind the service (HovisGenieServic) provided by GenieControlService so that service can be used while the App is running.

GenieApiDemoApplication class source code is as follows.

```
package com.dongburobot.genieapidemo;

import android.app.ActivityManager;
import android.app.ActivityManager.RunningServiceInfo;
import android.app.Application;
import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.IBinder;
import android.util.Log;

import com.dongburobot.HovisGenieServiceInterfaces.IHovisGenieService;

public class GenieApiDemoApplication extends Application {

    // HovisGenieService service binding
    public IHovisGenieService mBinder = null;

    private boolean mIsBinded;
    private Intent mIntent;

    public ServiceConnection mConnection = new ServiceConnection() {

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

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

    protected boolean isServiceRunning() {
        ActivityManager manager = (ActivityManager)getSystemService(ACTIVITY_SERVICE);

        for (RunningServiceInfo serviceInfo: manager.getRunningServices(Integer.MAX_VALUE)) {
            if (serviceInfo.service.getClassNam e().equals(“com.dongburobot.geniecontrol.HovisGenieService”)) {
                return true;
            }
        }
        }
        return false;

    }

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

        return false;
    }

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

        mIntent = null;
    }

    // Application Binding
   "GenieControlService" when a robot app starts and finishes.
    @Override
    public void onCreate() {
        super.onCreate();

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

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

        disconnectService();
    }
}
```

[Listing. 4-1] GenieApiDemoApplication.java

We will begin creating GenieApiDemoApplication class.
Open the GenieApiDemoApplication.java file and edit the file so that GenieApiDemo Application class will inherit Application class.

public class GenieApiDemoApplication extends Application { … }

When 'extends Application' is added, red underline will be added to GenieApiDemo Application. Presssing Ctrl+Shift+O from above the underline at this point will automatically add the required package and method to be Overridden.

TIP. **Press 'Ctrl+Shift+O' to add package not added from Eclipse.**

Additionally, 'Ctrl+Space' is the function key for automatic code completion during code writing.

Methods to be redefined are onCreate() and onTerminate(). onCreate() is the first method to be called when App is executed. Method connect Service() will be added since it connects the HovisGenieService provided by GenieContorlService when GenieApiDemo App starts to execute. onTerminate() is called when App completes execution. disconnectService() method is addes since service will be disconnected when App completes execution.

```
...
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() and disconnectService() connects and disconnects service. Declare service interface objects before defining these methods.

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


public class GenieApiDemoApplication extends Application {
public IHovisGenieService mBinder = null; // Service interface object
…
    …
    …
}
```

mBinder is an important IHovisGenieService object since all robot service will performed through mBinder. IHovisGenieService class is an interface for accessing the service HovisGenieService which is a service within GenieControlService. IHovisGenieService is defined in Hovis Service.jar. Namespace of the IHovisGenieService class package defined in HovisService.jar file is com.dongburobot.HovisGenieServiceInterfaces.IHovisGenieService which needs to be imported to use IHovisGenieService.

IHovisGenieService is created using AIDL(Android Interface Definition Language) and it is in charge of IPC(Inter Process Communication). Since we will be focusing on using the robot service using service binding, IPC and IDL will not be discussed. Refer to Andoid developer website for information concerning IPC related AIDL.

– http://developer.android.com/guide/components/aidl.html

Refer to [Listing. 4–1] and add connectService(), disconnectService(), and isServiceRunning(). Variables and objects in line19~20 should be added as well.

Since GenieApiDemoApplication class was created by inheriting basic Android Application class, this has to be noted in the AndroidManifest.xml. Add the following to the 11th line. android:name=”GenieApiDemoApplication”

```
〈manifest xmlns:android=”http://schemas.android.com/apk/res/android”
    package=”com.dongburobot.genieapidemo”
    android:versionCode=”1”
    android:versionName=”1.0” 〉

    〈uses–sdk
        android:minSdkVersion=”10”
        android:targetSdkVersion=”10” /〉

    〈application
        android:name=”.GenieApiDemoApplication”
        android:icon=”@drawable/ic_launcher”
        android:label=”@string/appa_name”
        android:theme=”@style/AppTheme” 〉
        〈activity
            android:name=”.MainActivity”
            android:label=”@string/title_activity_main” 〉
            〈intent–filter〉
                〈action android:name=”android.intent.action.MAIN” /〉
                〈category android:name=”android.intent.category.LAUNCHER” /〉
            〈/intent–filter〉
        〈/activity〉
    〈/application〉

〈/manifest〉
```

[Listing 4–2] AndroidManifest.xml

(1) Robot Service Connection

Our App and the robot service become connected when the App is first executed. When the App starts, onCreate() in GenieApiDemoApplication is called and connectService() is called within onCreate() to create service binding. Robot service connected by connectService() will remain connected until the App ends.

isServiceRunning() is used within connectService() to check if the service is operating in the backround and binds to robot service through bindService(). True or false is returned depending on the result of the binding.

```
@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;
    }
}
```

When connectService() is executed, isServiceRunning() in line 33 is called to check if the robot service (GenieServiceControl) is running in the MID. isServiceRunning() will return true if the robot service is running in the background and return false otherwise. If the robot service is running in the background, bindService() is called to actually bind to the robot service. binsService() is defined in the android.content, and the original format is as follows

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

Parameter

service	Defines the subject of the service connection. Intent will designate component name matching the intent filter defined in th service.
conn	Designates object to receive information when service starts or ends.
flags	Service binding option, BIND_AUTO_CREATE, BIND_DEBUG_UNBIND, BIND_NOT_FOREGROUND, BIND_ABOVE_CLIENT, BIND_ALLOW_OOM_MANAGE–MENT, or BIND_WAIVE_PRIORITY.

Return Value

True if service binding succeeds and false otherwise.

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

Intent is the first argument of bindService(). Intent is an object that one App component sends to run another App component. Intent sent by our App to GenieControlService through bindService() is same as sending a message stating robot services provided by GenieControlService will be used. Intent is defined as followin in connectService().

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

GenieContorlService의 AndroidManifest.xml 중에서.

```
〈intent–filter〉
〈action android:name=”com.dongburobot.HovisGenieServiceInterfaces.IHovisGenieService”/〉
〈/intent–filter〉
```

Argument IHovisGenieService.class.getName() returns robot service interface class name com.dongburobot.HovisGenieServiceInterfaces.IHovisGenieService to the Intent constructor. Also, Intent object having this class name is assigned to the mIntent.The reason mIntent has robot interface class name is due to the fact intent filter defined in the robot service is as follows

2nd argument of the bindService() is a ServiceConnection class object that receives information about the service when the service starts or ends. ServiceConnection object mConnection is declared in lines 22~31.

```
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() is called back (method is called automatically by the system)when service binding is complete and service interface connected to mBinder is assigned. In contrast, onServiceDisconnected() is called back when the service is disconnected and mBinder is set to null as service is no longer valid. In other words, mConnection manages mBinder depending on the robot service connection status.

Last or 3rd argument of bindService() is service binding option. BIND_AUTO_CREATE creates service automatically as long as binding is valid. There are other options available bu they are rarely used. When bindService() is called, depending on the service connection status and two redefined methods in defined ServiceConnection class object mConnection it is assigned as interface to IHovisGenieService object mBinder. When connectService() is all completed, App is ready to use the service.

(2) Robot Service Cancellation

Service cancellation is done by calling disconnectService()calling within onTerminate() in GenieApiDemoApplication class. onTerminate() is called automatically when App ends. Service ends together with App.

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

     disconnectService();
}
```

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

    mIntent = null;
}
```

When disconnectService() is called, service bind state is checked through mIsBinded. If the service is in bound state mConnection is checked to see if it is null and unbindService() is called. unbindService orginal form is as follows.

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

Parameter

conn	ServiceConnection object applied to bindService() during service connection.

```
unbindService(mConnection);
```

unbindService() cancels the connected service. unbindService() argument is ServiceConnection object which was used as 2nd argument of bindService() during service connection.

When unbindService() is called, service connection is cancelled and at the same time redefined onServiceDisconnected() is called to mConnection sent to the argument. Also, mBinder is set to null to complete the service connection cancellation.

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

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

It is important to get into a habit of cancelling the service properly as it can prevent improper robot system behavior.

4.3 Creating Base Activity for Robot System

In Android, one screen is composed of one Activity. During App creation, creating one Activity is same as creating one screen. During regular smart phone App creation. screen is created by inheriting Activity class defined in android.app.Activity.

Robot App can also be create screen by inheriting Actity class. However, unlike regular smart phone App, following two items must be considered before creating Acitivity.

Item to Consider

1. **Synchronization of control since there is only one robot**
2. **On/Off control required on App screen**

First, Synchronization of control is required since there is only one robot. If control synchroniztion is not guranteed, precise robot control cannot be guranteed. There is a potential danger since Activity manager may not gurantee control synchroniztion. The danger lies with the fact that screen may disappear but the Activity representing the screen did not end.

For example, Supposing robotforward Activity was in the midst of commanding the robot to move forward when low battery level switches robot to the charging mode and the screen also switches to ChargingActivity. If the robotforward Activity does not end and stays in the stack continuing to issue move forward command, this could affect the automatic charging which must be performed by ChargingActivity. This type of situation may occur very rarely and the developer could also forsee such a situation and try to write very precise codes to minimize the danger but the best method would be to stop such situation from occuring at all.

The robot App we will be creating will end the Activity when that Activity disappears from the screen. Method that is always called when Activity disappears from the screen is onPause(). Therefore, Activity must be guranteed to end at onPause().

```
package com.dongburobot.genieapidemo;

import android.app.Activity;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;

public class BaseActivity extends Activity {

    protected GenieApiDemoApplication myRobotApp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Screen setup
        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);

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

onCreate() in 13th line of BaseActivity is called when Activity is created. Required parts for starting Activity are created in onCreate(). Screen setup codes are required. This falls under Item to Consider #2 .

```
// Screen Setup (Item to Consider 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);
```

Codes in line17~23 are screen setup codes. Codes in line17~18 are to set current screen to FULL SCREEN. Codes in line19~22 shows screen even where there is a lock, does not show Android key input screen, screen is always On, uses flag for maintaining On state to setup screen. Code in line 23 removes screen title bar. There ar many more flags used for screen setup. Refer to Android developers site for more information.

onPause() in line 29 is called at the time Activity disappears from the screen. As mentioned previously, to sychronize robot control, our robot App does not leave Activity in the stack. Use finish() in line 32 to end the Activity completely. This falls under “Item to Consider 1”.

```
@Override
protected void onPause() {
    super.onPause();
    // Item to Consider 1
    this.finish();
```

GenieApplicationDemoApplication object is decalred in line 10. myRobotApp assigns global application object of the current process returned by getApplicationContext().
Codes in Line 25 within onCreate(), objects and variables included in GenieApiDemoApplication class can be used by all Activity created by inheriting BaseActivity. This sets the base for using robot service by from all Activity.

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

This ends the basic setup for starting robot programming. This chapter discussed method of using robot service to control the robot. Next chapter will discuss functions provided by robot service.

05 HOVIS Genie API (Application Programing Interface)

Service for controlling Hovis Genie is performed by GenieContorlService running in the back–ground in MID. User is able to use the robot service by using GenieControlService to bind the service.

Previous chapter discussed binding to use the robot service. This chapter will discuss functions provided by Hovis Genie API. Functions provided by Hovie Genie API are as follows.

Drivetrain Function

Drivetrain function is collection of functions related to Hovis Genie omniwheel drivetrain. These functions are used for configuring the drivetrain wheel and wheel operation. Actual robot movement is performed by the navigation function.

Navigation Function

Navigation function is responsible for actual movement of Hovis Genie. These functions abstract Omniwheel drive train so that Genie can move following the x,y, theta coordinates.

Sensor Function

Sensor related function provides control interface for various sensors (distance sensor, ground detection sensor, touch sensor, and etc).

TTS(Text–to–Speech) Function

TTS function converts text string input to robot speech.

Sound(sound effect) Function

Sound function can output various sound effects depending on the status of the robot.

Multimedia(Audio & Video) Function

Provides play/stop function for audio video files (MP3, MP4, AVI …) by the robot.

Motion & Servo Motor Related Function

Provides robot motion related functions.

5.1 Drivetrain Function

dmel_drive_servo_on

```
boolean dmel_drive_servo_on(ComponentName cn)
                             throws android.os.RemoteException
```

Robot drivetrain (DRS–0102) “Servo On” 3 servo motors
Parameters:
cn – Component nameReturns:
Return true when successful, false when fail.
Throws:
android.os.RemoteException

dmel_drive_set_movable

```
void dmel_drive_set_movable(ComponentName cn,
                           boolean flag)
                             throws android.os.RemoteException
```

Setup robot drivetrain operational state
Parameters:
cn – Component name
flag – Operational true, Not operational false
Throws:
android.os.RemoteException

dmel_drive_get_movable

```
boolean dmel_drive_get_movable(ComponentName cn)
                                throws android.os.RemoteException
```

Return robot drivetrain operational state.
Parameters:
cn – Component name
Returns:
Operational true, Not operationa false return
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
```

Setup horizontal speed of the robot drivetrain
Parameters:
cn – Component name
vel – Horizontal velocit (unit : m/s)
ang – Compensated angle from horizontal plane (unit : rad value + CCW, – 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
```

Set rotational vlecocity for drivetrain.
Parameters:
cn – Component name
vel – rotational velocity (rad/s)
Returns:
Success true, fail false return
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
```

Drive robot drivetrain.
Parameters:
cn – Component name
lin – horizontal velocity (m/s)
ang – rotational velocity (rad/s) +는 CCW, -는 CW
horizontal_ang – upto set value (+ CCW, – CW) horizontal movement with compensated angle.
flag – true horizontal movement, move according to horizontal_ang value, false move according to angular movement ang value
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
```

Set maxium robot drivetrain horizontal velocity.

Parameters:

cn – Component name

vel – maximum horizontal velocity (m/s)

Returns:

Setup success true, fail false

Throws:

android.os.RemoteExceptio

dmel_param_get_max_lin_velocity

```
double dmel_param_get_max_lin_velocity(ComponentName cn)
                           throws android.os.RemoteException
```

Returm robot drivetrain maximum velocity.

Parameters:

cn – Component name

Returns:

Return maximum horizontal velociy (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
```

Set robot drivetrain maximum angular velocity.

Parameters:

cn – Component name

vel – maximum angular velocity (rad/s)

Returns:

setup success true, fail false

Throws:

android.os.RemoteException

dmel_param_get_max_ang_velocity

```
double dmel_param_get_max_ang_velocity(ComponentName cn)
                                       throws android.os.RemoteException
```

Return robot drivetrain maximum angular velocity

Parameters:

cn – Component name

Returns:

Return maximum angular velocity (rad/s)

Throws:

android.os.RemoteException

android.os.RemoteException

5.2 Navigation Function

dmel_robot_set_pose

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

Set current robot position by x, ,y, theta. If current robot position is set to (0, 0, 0),current position becomes standard.

Parameters:

cn – Component name

x – (meter)

y – (meter)

theta – (Radian –PI ~ +PI)

Returns:

success true, fail false return

Throws:

android.os.RemoteException

dmel_robot_get_pose

```
double[] dmel_robot_get_pose(ComponentName cn)
                            throws android.os.RemoteException
```

Return current position based on starting position

Parameters:

cn – Component name

Returns:

Stadard starting position return each x, y, theta(Radian –PI ~ +PI) value to double[0] ~ double[2]

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
```

From the robot starting position, move to (x, y, theta) position.

Parameters:

cn – Component name

x – standard starting position x (meter)

y – standard starting position y (meter)

theta – Robot Heading from starting point standard robot heading theta (Radian –PI ~ +PI)

Throws:

android.os.RemoteException

dmel_navigate_prepare

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

Perform initialization for robot navigation. When initialized, current position is set to (0,0,0), current Heading is set to 0 degrees.

Parameters:

cn – Component name

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
```

Add robot waypoint. based on standard starting position set in the robot (0, 0, 0) set waypoint.. (dmel_navigate_prepare) (reference) robot moves to the set waypoint through dmel_navigate_start().

Parameters:

cn – Component name

x – standard starting position x (meter)

y – standard starting position y (meter)

theta – Robot Heading from starting point standard robot heading theta (Radian –PI ~ +PI)

rflag – when true, robot will face theta degree direction when reaching destination

time – robot waiting time after reaching destination (sec)

Returns:

Throws:

android.os.RemoteException

dmel_navigate_start

boolean dmel_navigate_start(ComponentName cn)
throws android.os.RemoteException

Run robot navigation. Robot navigation will visit previously set goal_pose one by one.

Parameters:

cn – Component name

Returns:

Success true,fail false return

Throws:

android.os.RemoteException

dmel_navigate_stop

boolean dmel_navigate_stop(ComponentName cn)
throws android.os.RemoteException

Stop robot navigation.

Parameters:

cn – Component name

Returns:

Success true, fail false return

Throws:

android.os.RemoteException

5.3 Sensor Function

dmel_robot_get_obs

double[] dmel_robot_get_obs(ComponentName cn)
throws android.os.RemoteException

Return front detection PSD sensor value. From front to clockwise direction 0, 1, 2, 3, 4

Parameters:

cn – Component name

Returns:

Retun 5 front detection sensor values to double[0] ~ double[4]

Throws:

android.os.RemoteException

dmel_robot_get_cliff

```
double[] dmel_robot_get_cliff(ComponentName cn)
                                   throws android.os.RemoteException
```

Return ground detection PSD sensor value. From front to clockwise direction 0, 1, 2
Parameters:
cn – Component name
Returns:
Return 3 ground detection sensor values to double[0] ~ double[2]
Throws:
android.os.RemoteException

dmel_hri_get_head_touch_info

```
boolean dmel_hri_get_head_touch_info(ComponentName cn)
                                           throws android.os.RemoteException
```

Return head touch detction.
Parameters:
cn – Component name
Returns:
Head touched true, not touched false return
Throws:
android.os.RemoteException

dmel_hri_get_hand_touch_info

```
boolean[] dmel_hri_get_hand_touch_info(ComponentName cn)
                                           throws android.os.RemoteException
```

Return hand (boths hands) touch detection.
Parameters:
cn – Component name
Returns:
touched true, not touched false return. boolean[2] (boolean[0] : left hand touch, boolean[1] : right hand touch)
Throws:
android.os.RemoteException

5.4 TTS(Text–to–Speech) Function

dmel_tts_speak

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

Play TTS(Text–to–Speech).

Parameters:

cn – Component name

text – txt string to be changed to speech String ex)”Hello. Iam Hovie Genie.”

Throws:

android.os.RemoteException

dmel_tts_stop

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

Stop TTS(Text–to–Speech).

Parameters:

cn – Component name

Throws:

android.os.RemoteException

5.5 Sound(sound effect) Function

dmel_sound_play_intro

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

Play sound effect. (Intro sound effect)

Parameters:

cn – Component name

Throws:

android.os.RemoteException

dmel_sound_play_chimes

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

Play sound effect. (Chime sound effect)
Parameters:
cn – Component name
Throws:
android.os.RemoteException

dmel_sound_play_dingdong

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

Play sound effect. (Ding dong sound effect)
Parameters:
cn – Component name
Throws:
android.os.RemoteException

dmel_sound_play_horn

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

Play sound effect. (Horn sound effect)
Parameters:
cn – Component name
Throws:
android.os.RemoteException

dmel_sound_play_fail

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

Play sound effect. (Fail sound effect)
Parameters:
cn – Component 이름
Throws:
android.os.RemoteException

[Speech Recognition Related Function]

dmel_voice_recognition_start

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

Input speech for speech recognition.
Parameters:
cn – Component name
Throws:
android.os.RemoteException

dmel_voice_recognition_stop

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

End speech input for speech recognition.
Parameters:
cn – Component name
Throws:
android.os.RemoteException

dmel_voice_recognition_set_mode

```
void dmel_voice_recognition_set_mode(ComponentName cn,
                              boolean is_auto)
                              throws android.os.RemoteException
```

Change mode to speech recognition mode
Parameters:
cn – Component name
is_auto – true automatically repeat speech recognition from start to end.
Throws:
android.os.RemoteException

dmel_voice_recognition_get_mode

```
boolean dmel_voice_recognition_get_mode(ComponentName cn)
                                   throws android.os.RemoteException
```

Return speech recognition possibility.
Parameters:
cn – Component name
Returns:
Speech recognition possible true, not possible false
Throws:
android.os.RemoteException

5.6 Multimedia(Audio & Video) Function

dmel_audio_set_volume

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

Adjust volume. Minimum volume : 0, Maximum volume : 1
Parameters:
cn – Component name
volume – Volume size setup (0~1)
Throws:
android.os.RemoteException

dmel_audio_get_volume

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

Return current volume size. Minimum volume : 0, Maximum volume : 1
Parameters:
cn – Component name
Returns:
Return volume size (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
```

Play MP3 file. (default path set to /sdcard/dongbu/)
When actual path is /sdcard/dongbu/audio/test.mp3
mBinder.dmel_audio_play_mp3(getComponentName() , “audio/test,mp3”);
Parameters:
cn – Component name
path – MP3file path (include file name, except default path)
Throws:
android.os.RemoteException

dmel_audio_stop_mp3

```
void dmel_audio_stop_mp3(ComponentName cn)
                        throws android.os.RemoteException
```

Stop MP3 play
Parameters:
cn – Component name
Throws:
android.os.RemoteException

dmel_video_play_mp4

```
void dmel_video_play_mp4(ComponentName cn,
                       java.lang.String path)
                         throws android.os.RemoteException
```

Play MP4 file. (Default path set to /sdcard/dongbu/)
When actual path is /sdcard/dongbu/video/test.mp3
mBinder.dmel_video_play_mp4(getComponentName() , “video/test.mp3”);
Parameters:
cn – Component name
path – MP4file path (include file name, except default path)
Throws:
android.os.RemoteException

5.7 Motion & Servo Motor Related Function

dmel_motion_play

```
void dmel_motion_play(ComponentName cn,
                    java.lang.String motion)
                     throws android.os.RemoteException
```

Play robot motion file. (Default path set to /sdcard/dongbu/motion/)
Parameters:
cn – Component name
motion – Motion file name 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
```

Play robot motion file. Media Sync option and repeat setup possible.
(Path set to /sdcard/dongbu/motion/)
Parameters:
cn – Component name
motion – Motion file name ex) "01n_shake_head.dmt"
repeat – Continuously repeat until stop command received
sync – Sync with media. (Takes priority over repeat)
Throws:
android.os.RemoteException

dmel_motion_stop

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

Stop current motion
Parameters:
cn – Component name
Throws:
android.os.RemoteException

dmel_motion_servo_on

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

Set "Servo ON" to upper body servo motor with ID value
Parameters:
cn – Component name
id – id motor ID (0: right shoulder motor, 1: right upper arm motor, 2: right forearm motor, 3: left shoulder motor, 4: left upper arm motor, 5: left forearm motor, 18: head motor, 19: waist motor)
Returns:
Success true, fail false return
Throws:
android.os.RemoteException

dmel_motion_servo_off

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

Set “Servo OFF” to upper body servo motor with ID value

Parameters:

cn – Component name

id – id motor ID (0: right shoulder motor, 1: right upper arm motor, 2: right forearm motor, 3: left shoulder motor, 4: left upper arm motor, 5: left forearm motor, 18: head motor, 19: waist motor)

Returns:

Success true, fail false return

Throws:

android.os.RemoteException

dmel_motion_set_servo

```
boolean dmel_motion_set_servo(ComponentName cn,
                              int on)
                              throws android.os.RemoteException
```

Set all upper body servo motor to “Servo ON”.

Parameters:

cn – Component name

on – Servo ON : 1, Servo OFF : 0

Returns:

Success true, fail false return

Throws:

android.os.RemoteException

dmel_motion_get_position

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

Return current position value of the upper body servo motor with Servo ID

Parameters:

cn – Component name

id – motor ID (0: right shoulder motor, 1: right upper arm motor, 2: right forearm motor, 3: left shoulder motor, 4: left upper arm motor, 5: left forearm motor, 18: head motor, 19: waist motor)

Returns:

Returns current position value of applicable motor

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
```

Adjust 0 point value of the upper body motor with Servo ID

Parameters:

cn – Component name

id – motor ID (0: right shoulder motor, 1: right upper arm motor, 2: right forearm motor, 3: left shoulder motor, 4: left upper arm motor, 5: left forearm motor, 18: head motor, 19: waist motor)

val – 0 point adjust value (–127 ~ 127)

Returns:

succesful 0 point adjustment true, fail false return

Throws:

android.os.RemoteException

5.8 Head LED Control Realted Function

dmel_hri_set_brow_beam

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

Setup forehead LED.

Parameters:

cn – Component name

brightness – 0 : off, 1 : weak light, 2 : medium light, 3 : bright light

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
```

Setup eye LED.

Parameters:

cn – Component name

eyeCode – 0 : maintain current state, 1 : round and round, 2 : eyebrow blink, 3 : eyebrow stop, 4 : light all, 5 : all blink, 6 : left-right effect, 7 :up-down effect, 8 : off

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
```

Setup ear LED.

Parameters:

cn – Component name

earCode – 0 : maintain current state, 1 : On, 2 : Off, 3 : blink

Throws:

android.os.RemoteException

dmel_hri_set_mouth_led

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

Setup mouth LED.

Parameters:

cn – Component name

mouthCode – 0 : maintain current state, 1 : middle ON, 2 : three ON, 3 : middle blink, 4 : all three blink, 5 : talk

Throws:

android.os.RemoteException

06 Hovis Genie Basic Example

In Chapter 6, we will start robot programming usng the robot service in ernest. In Chapter 4, we created GenieApiDemo project to do basic work to use robot service. In this section, we will add various robot control examples and complete GenieApiDemo project.

There are total of 7 screens combined in GenieApiDemo. In other words, GenieApiDemo is an App with 7 Activity combined. File to be added and edited for each section are shown int the table blow [table 6-1] which shows 7 Activity contained in GenieApiDemo.

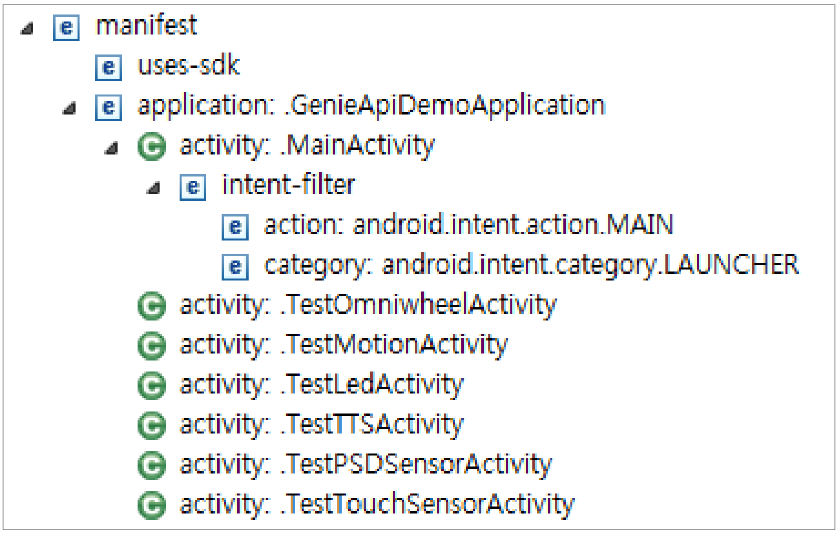
Section	Content	UI layout	Source code
6.1.	Main screen	activity_main.xml	MainActivity.java
6.2.	Drivetrain control	activity_testomniwheel.xml	TestOmniwheelActivity.java
6.3.	Motion control	activity_testmotion.xml	TestMotionActivity.java
6.4.	Head LED control	activity_testled.xml	TestLedActivity.java
6.5.	TTS control	activity_testtts.xml	TestTTSActivity.java
6.6.	PSD sensor control	activity_testpsdsensor.xml	TestPSDSensorActivity.java
6.7.	Touch sensor control	activity_testtouchsensor.xml	TestTouchSensorActivity.java

[Table 6-1] GenieApiDemo screen format

Writing example found in each section adds an Activity to the App thereby changing the structure of the App. These changes have to be declared and added in the AndroidMainfest.xml. Open the AndroidManifest.xml and add the remaining 6 Activitiy excluding the Activity already added when project was created by referring to the [Listing 6-1] lines 22~27.

```
〈manifest xmlns:android=”http://schemas.android.com/apk/res/android”
    package=”com.dongburobot.genieapidemo”
    android:versionCode=”1”
    android:versionName=”1.0” 〉

    〈uses–sdk
        android:minSdkVersion=”10”
        android:targetSdkVersion=”10” /〉

    〈application
        android:name=”.GenieApiDemoApplication”
        android:icon=”@drawable/ic_launcher”
        android:label=”@string/app_name”
        android:theme=”@style/AppTheme” 〉
        〈activity
            android:name=”.MainActivity”〉
            〈intent–filter〉
                〈action android:name=”android.intent.action.MAIN” /〉
                〈category android:name=”android.intent.category.LAUNCHER” /〉
            〈/intent–filter〉
        〈/activity〉
        〈activity android:name=”.TestOmniwheelActivity” /〉
        〈activity android:name=”.TestMotionActivity” /〉
        〈activity android:name=”.TestLedActivity” /〉
        〈activity android:name=”.TestTTSActivity” /〉
        〈activity android:name=”.TestPSDSensorActivity” /〉
        〈activity android:name=”.TestTouchSensorActivity” /〉
    〈/application〉

〈/manifest〉
```

[Listing 4–2] BaseActivity.java



```
◢ e manifest
    e uses-sdk
  ◢ e application: .GenieApiDemoApplication
    ◢ G activity: .MainActivity
      ◢ e intent-filter
          e action: android.intent.action.MAIN
          e category: android.intent.category.LAUNCHER
      G activity: .TestOmniwheelActivity
      G activity: .TestMotionActivity
      G activity: .TestLedActivity
      G activity: .TestTTSActivity
      G activity: .TestPSDSensorActivity
      G activity: .TestTouchSensorActivity
```

[Diagram 6-1] AndroidManifest.xml structure

[Diagram 6-1] shows the strucre of GenieApiDemo App seen through AndroidManifest.xml. GenieApiDemo 〈application〉(=App) is enabled by GenieApiDemoApplication.java and has 7 〈activity〉. Each 〈activity〉 is enabled by MainActivity.java, …, TestTouchSensorActivity.java. Among the 〈activity〉, MainActivity has 〈intent-filter〉 containing LAUNCHER category and MAIN action which shows that the screen is the first screen to appear when App is started from the launcher.

We already edited to the structure of the App from the AndroidManifest.xml so it is no longer necessary to make any changes to AndroidManifest.xml each time new example is added. Not paying enough attention to the App structure in AndroidManifest.xml is a common mistake made by many App developers. Care should be taken to avoid this type mistake that can cause App error.

Starting from the next page, explanation will be based on the examples. Refer to the Android developer website or to other Android sources for more indepth information.

6.1 Main Screen Composition

In Android, one screen is composed of one Activity and this Actviviy normally contains XML and Java file which composes UI layout. Files contained in the Main screen are activity_main.xml and MainActivity.java

There is no need to add 2 files mentioned above as both activity_main.xml and MainActivity.java files were automatically added in Chapter 4 when GenieApiDemo was created.

GenieApiDemo 〉 res 〉 layout folder in Eclipse Package Explorer contains all UI layout including the activity_main.xml .

Find and double click on the activity_main.xml and add the source as shown in [Listing 6-2].

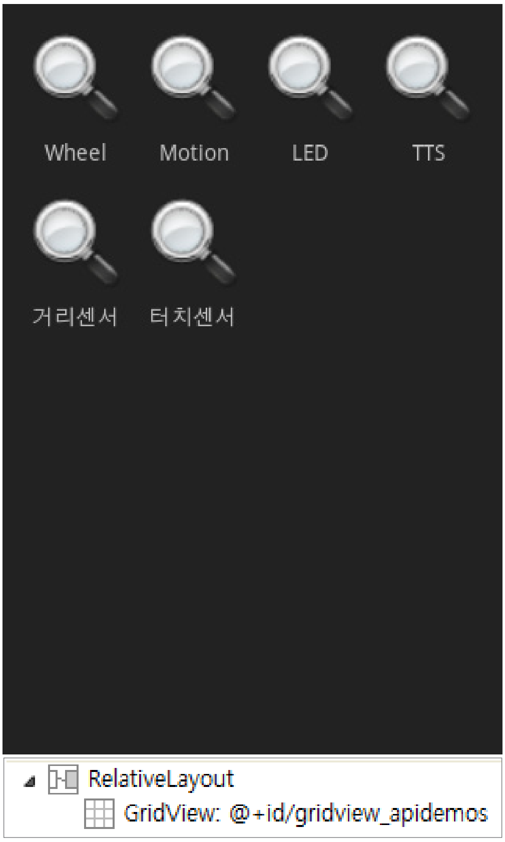
```
〈RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    xmlns:tools=”http://schemas.android.com/tools”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent” 〉

        〈GridView
            android:id=”@+id/gridview_apidemos”
            android:layout_width=”match_parent”
            android:layout_height=”match_parent”
            android:padding=”10dp”
            android:verticalSpacing=”10dp”
            android:horizontalSpacing=”10dp”
            android:numColumns=”auto_fit”
            android:columnWidth=”60dp”
            android:stretchMode=”columnWidth”
            android:gravity=”center”
            android:background=”#222222”
        /〉

〈/RelativeLayout〉
```

[Listing 6-2] activity_main.xml

activity_main.xml structure has 〈RelativeLayout〉 as the highest tag with 〈GridView〉 below.

RelativeLayout is a layout which places the widgets on the screen relative to other and GridViw displays widgets in in a two–dimensional, scrollable grid.

[Diagram 6–2] shows the screen showing UI created by activity_main.xml.



[Diagram 6–2] Main screen

〈GridView〉 takes up the whole screen. Assigned value "match_parent" in size properties located in line 8~9 android:layout_width and android:layout_height makes the size same as the parent screen and the parent screen 〈ReleativeLayout〉 takes up the whole screen.

〈GridView〉 taking up the whole screen includes the lower widgets in grid formation. Other properties of 〈GridView〉 are used to setup how lower widgets will be placed in the screen. Presently, widget placement is setup as android:numColumns="auto_fit". Size of the lower widgets determine how many widgets can be placed on each line.

➜ Important properties of ReleativeLayout

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

➜ Important properties of GridView

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

Lower widges in 〈GridView〉 are composed of image icons and texts. Another XML file is required to define the widgets. Add /res/layout에 demoitems.xml file and edit the source code as follows.

```
〈?xml version=”1.0” encoding=”utf-8”?〉
〈LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    android:layout_width=”fill_parent”
    android:layout_height=”fill_parent”
    android:orientation=”vertical”
    android:gravity=”center”〉

    〈ImageView
        android:layout_width=”65dp”
        android:layout_height=”65dp”
        android:padding=”6dp”
        android:gravity=”center”
        android:id=”@+id/demo_icon”/〉
    〈TextView
        android:layout_width=”65dp”
        android:layout_height=”30dp”
        android:gravity=”center”
        android:ellipsize=”end”
        android:singleLine=”true”
        android:id=”@+id/demo_name”/〉

〈/LinearLayout〉
```

[Listing 6-3] demoitems.xml

demoitems.xml is composed of 〈ImageView〉 and 〈TextView〉. This represents the image icons and the texts in [Diagram 6–2]. This only expresses the widget structure expressed in 〈GridView〉. Actual data source are not in 〈GridView〉. Receiving the actual data source occurs in MainActivity.java. Open the MainActivity.java and write the source codes as shown in [Listing 6–4].

1) Inhertiting BaseActivity Class (Line 18)

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

BaseActivity is a class created in Chapter 4. It is a redefined Activity class taking into account special properties of robot programming. BaseActivity uses the whole MID screen and prevents Activity from piling in Android stack and also allows use of global varialble mBinder to use service.

2) Declare MainActivity class member variable (Lines19~23)

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

mMainGrid is a variable for assigning GridView declared in activity_main.xml. mMainAdapter is an object of Inner class MainAdapter which will be written in (4). It is used for selecting data source which will be assigned to GridView. Lower widgets in GridView are composed of image icons and texts. Data concerning these two will be assigned to mDemoImgList and mDemoNameList.

3) Create onCreate() (Lines 26~50)

Contains source codes for using member variable declared in (2).
In line 28, setContentView() is used to activity_main.xml to MainActivity. In lines 30~44, Data to be output to lower widgets in GridView are assigned to LinkedList. Lines 46~48 assigns Adapter to GridView. Adapter is used for supplying multiple data source.Line 49 contains code that makes it possible for widgets assigned to GridView to be clicked.

Reference : Assingning resouces in Android (R.java)

R.drawable. filename → /res/drawable–*dpi image resource

R.layout. filename→ /res/layout access layout XML file

R.id.ID → From each layout file access delcared widget using ID

4) Create MainActivity Inner class MainAdapter (Lines 97~126)

This is the most importand section of this example code. MainAdapter is our own Adapter class inherited from BaseAdapter. MainAdapter is used for providing data source to GridView. Since MainAdapter was inherited from BaseAdapter, getView(), getCount(), getItem(), getItemId() has to be overridden.

getView(), getCount() implemented in MainAdapter are called when Adapter applied widget is sent to output. getItem() and getItemId() must be implemented but explanation will be skpped since they do not play any role.

getView() method is called each time GridView draws lower widget on the screen. When MainActivity was created. actual data image and text to be output were already assigned to GridView by onCreate(). getView() uses this data.

getCount() returns actual number or data that will be ouput to GridView. Since our example has 6 image and text, 6 will be returned. Returned 6 calls getView() 6 times resulting in 6 icons and texts in GridView.

Adapter will always be used in source codes such as List UI that require multiple data source.

5) Inherit OnItemClickListener interface multilple times (Line 18)

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

Inherit OnItemClickListener multiple times to process the touch event when Grid–View icon is touched. Input ‘implements OnItemClickListener’, place cursor on top of

‘MainActivity ‘ and add onItemClick().

6) Create onItemClick() (Lines 58~95)

onItemClick() is a method that processes icon touch event in GridView.
one of the onItemClick() arguments int position shows the location of the touched icon in GridView. Top left position value is 0 and the value increases by 1. In our example, Wheel icon returns 0, touch sensor icon returns 5.

onItemClick() uses retuned position value to switch the screen to particular Activity screen. Coed for starting other Activity is as follows.

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

When switched to new Activity by startActivity(), MainActivity calls onPause() function. However, as function is not defined, Pause() function from the parent class BaseAcitivity is called. When onPause() from BaseActivity is called, Activity is ended by finish(). (Refer to Chapter 4).

```
 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;
```

```
    public LinkedList〈Integer〉 mDemoImgList;

    private GridView mMainGrid;
    private MainAdapter mMainAdapter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mDemoNameList = new LinkedList〈String〉();
        mDemoImgList = new LinkedList〈Integer〉();
        mDemoNameList.add(“Wheel”);
        mDemoNameList.add(“Motion”);
        mDemoNameList.add(“LED”);
        mDemoNameList.add(“TTS”);
        mDemoNameList.add(“Distance sensor”);
        mDemoNameList.add(“Touch sensor”);

        mDemoImgList.add(R.drawable.icon1);
        mDemoImgList.add(R.drawable.icon1);
        mDemoImgList.add(R.drawable.icon1);
        mDemoImgList.add(R.drawable.icon1);
        mDemoImgList.add(R.drawable.icon1);
        mDemoImgList.add(R.drawable.icon1);

        mMainGrid = (GridView) findViewById(R.id.gridview_apidemos);
        mMainAdapter = new MainAdapter();
        mMainGrid.setAdapter(mMainAdapter);
        mMainGrid.setOnItemClickListener(this);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
```

```
}

public void onItemClick(AdapterView〈?〉 parent, View v, int position, long id) {

    Intent intent;

    switch (position) {
    case 0:
        // wheel(omni wheel) go to control Activity
        intent = new Intent(this, TestOmniwheelActivity.class);
        startActivity(intent);
        break;
    case 1:
        // go to motion control  Activity
        intent = new Intent(this, TestMotionActivity.class);
        startActivity(intent);
        break;
    case 2:
        // go to head  LED control Activity
        intent = new Intent(this, TestLedActivity.class);
        startActivity(intent);
        break;
    case 3:
        // go to TTS Activity
        intent = new Intent(this, TestTTSActivity.class);
        startActivity(intent);
        break;
    case 4:
        // go to PSD Activity
        intent = new Intent(this, TestPSDSensorActivity.class);
        startActivity(intent);
        break;
    case 5:
        // go to Touch Activity
        intent = new Intent(this, TestTouchSensorActivity.class);
        startActivity(intent);
        default:
            break;
        }
    }

    public class MainAdapter extends BaseAdapter {

        public int getCount() {
            return mDemoImgList.size();
        }

        public Object getItem(int position) {
            return null;
        }

        public long getItemId(int position) {
            return position;
        }

        public View getView(int position, View convertView, ViewGroup parent) {
            if (convertView == null) {
                LayoutInflater li =  (LayoutInflater) getApplicationContext()
                        .getSystemService(LAYOUT_INFLATER_SERVICE);
                convertView = li.inflate(R.layout.demoitems, null);
            }

            ImageView icon = (ImageView) convertView.findViewById(R.id.demo_icon);
            TextView name = (TextView) convertView.findViewById(R.id.demo_name);

            icon.setBackgroundResource(mDemoImgList.get(position));
            name.setText(mDemoNameList.get(position));

            return convertView;
        }
    }
}
```

[Listing 6-4] MainActivity.java

6.2 Drivetrain Control

Drivetrain control example is for testing the Hovis Genie omni wheel control. In this example program, robot will move forward 5ocm and come back to the original position. Drivetrain control Activity is comprised of activity_testomniwheel.xml and TestOmni wheelActivity.java. Add each file to /res/layout and /src/com/dongburobot/geniea. First we will create UI for the drivetrain control Activitiy . Open activitiy_ testomniwheel.xml file and edit the source code as following.

```
〈RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    xmlns:tools=”http://schemas.android.com/tools”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent” 〉

    〈!— Test Title—〉
    〈TextView
        android:id=”@+id/tv_title”
        android:layout_width=”match_parent”
        android:layout_height=”85dp”
        android:text=”Omniwheel test”
        /〉

    〈!— Robot Image --〉
    〈ImageView
        android:id=”@+id/iv_image”
        android:layout_width=”match_parent”
        android:layout_height=”320dp”
        android:layout_below=”@id/tv_title”
        android:contentDescription=”이미지”
        /〉

    〈!-- Start and End Buton —〉
    〈LinearLayout
        android:layout_width=”match_parent”
        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="Start"
33:              /〉
34:          〈Button
35:              android:id="@+id/btn_quit"
36:              android:layout_width="160dp"
37:              android:layout_height="wrap_content"
38:              android:text="End"
39:              /〉
40:      〈/LinearLayout〉
41:
42: 〈/RelativeLayout〉
```

[Listing 6–5] activity_testomniwheel.xml

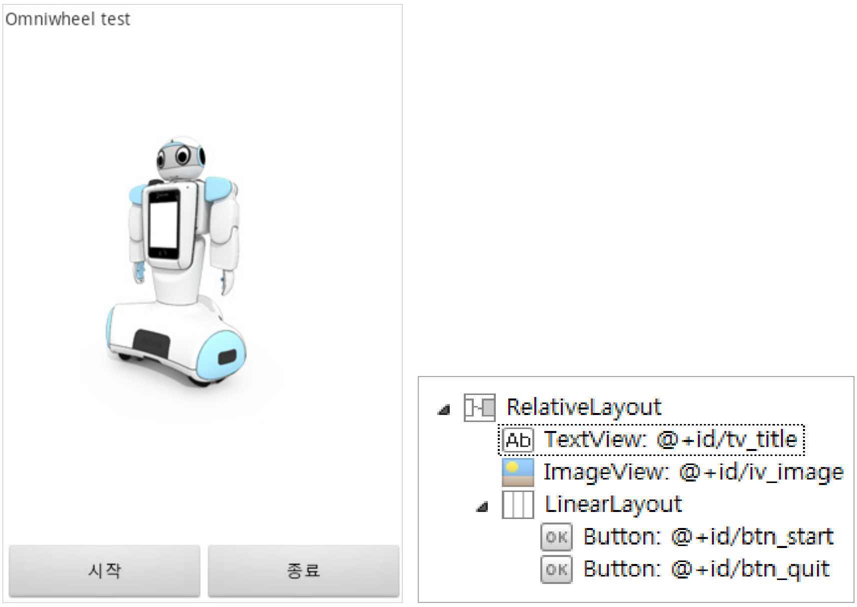
〈RelativeLayout〉 is the highest tag in activity_testomniwheel.xml structure and contains 〈TextView〉, 〈ImageView〉, 〈LinearLayout〉. 〈LinearLayout〉 contains two 〈Button〉 widgets.

When 〈RelativeLayout〉 is used, each widget must include property assigning relative position. If the property assigning relative positionis not assigned, widgets will be placed with top left side of the parent as standard reference. This part was skipped in the previous example since 〈RelativeLayout〉 had only one 〈GridView〉 widget. However when there are multiple widgets as in this example, missing property is a cause for error.

Among the 〈RelativeLayout〉 widgets, 〈TextView〉 does not have property and will be placed at top left. 〈ImageView〉 property is in line 19. This line places 〈ImageView〉 below 〈TextView〉 whic has tv_title as id. 〈LinearLayout〉 proper is in line 27. 〈LinearLayout〉 will be paced at lowest section of 〈RelativeLayout〉

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

Screen UI is shown below.



- RelativeLayout
  - TextView: @+id/tv_title
  - ImageView: @+id/iv_image
  - LinearLayout
    - Button: @+id/btn_start
    - Button: @+id/btn_quit

[Diagram 6-3] Drivetrain control

from rc 〉 com.dongburobot.genieapi in Eclipse Package Explorer, select TestOmniwheelActivity.java file and add the source codes in [Listing 6-6]. Writing method is as follows. This example requires wheel_1.png. Copy this file to res/drawable-mdpi. (refer to www.dongburobot.com archives)

1) Inherit BaseActivity

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

2) Declare widget related member variable

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

3) Create onCreate()

onCreate() connects layout XML file and sets widget resource to each member variable. setContentView() in line 21 expresses UI defined in activity_testomniwheel.xml to Activity screen. Lines 23~25 is used for robot image [Diagram 6-2] setup. Lines 27~30 contain codes used to setup touch event to the buttons.

4) Inherit OnClickListener interface multiple times and then create onClick()

This example has two buttons, 'Start' and 'End' . Write source codes to process touch event when button is pressed. First inherit OnClickListener interface multiple times.

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

Add 'implements OnClickListener' and then place mouse cursor over'TestOmniwheel Activity' to add onClick().

5) Creat moveOmniWheel(), stopOmniWheel()

moveOmniWheel() is a function that makes robot move forward 50cm and then return to the original position. stopOmniWheel() is a code that stops the robot.

6) Creat onPause()

onPause() function is called as soon as TestOmniwheelActivity disappears from the screen. super.onPause() in line 44 calls onPause() from the BaseActivity() which is parent class of TestOmniwheelActivity. BaseActivity onPause() calls finish() to end the Activity completely so that Activity will not remain in Android stack.

7) Create onKeyDown()

As MainActivity was also created by inherting BaseActivity. MainActivity will not stay in Android stack when icon is clicked in GridView and screen switches to another example. Normally in most of the Apps, when Activity switches from A -> B Activity, A Activity remains saved in stack so that activity can switch back to A Activity when back button is pressed. When MainActivity is not saved in Android memory stack. Pressing the back button will end the App.
onKeyDown() in lines 34-40 contain codes for switching screen to MainActivity when Back button is pressed. Due to these codes, App will switch to main screen instead of ending the App when back button is pressed.

TestOmniwheelActivity.java source codes are as follows.

```
package com.dongburobot.genieapidemo;

import android.content.Intent;
import android.os.Bundle;
import android.os.RemoteException;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

public class TestOmniwheelActivity extends BaseActivity
                                                implements OnClickListener {

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

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_testomniwheel);

        mIvWheelImg = (ImageView)findViewById(R.id.iv_image);
        mIvWheelImg.setImageResource(R.drawable.wheel_1);
        mIvWheelImg.setScaleType(ImageView.ScaleType.FIT_XY);

        mBtnStart = (Button)findViewById(R.id.btn_start);
        mBtnQuit = (Button)findViewById(R.id.btn_quit);
        mBtnStart.setOnClickListener(this);
        mBtnQuit.setOnClickListener(this);
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            Intent intent = new Intent(this, MainActivity.class);
            startActivity(intent);
        }
        return super.onKeyDown(keyCode, event);
    }

    @Override
    protected void onPause() {
        super.onPause();
    }

    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.btn_start:
            moveOmniwheel();
            break;
        case R.id.btn_quit:
            stopOmniwheel();
            break;
        default:
            break;
        }
    }

    private void moveOmniwheel() {
        if (myRobotApp.mBinder == null)
            return;

        try {
            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());
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

    private void stopOmniwheel() {
        if (myRobotApp.mBinder == null)
            return;

        try {
            myRobotApp.mBinder.dmel_navigate_stop(getComponentName());
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

[Listing 6-6] TestOmniwheelActivity.java

The most importan section in this example are the moveOmni wheel() and stopOmniwheel() functinons which are called when Start and End button is pressed. These two functions use the robot service to control the drivetrain. moveOmniwheel() makes the robot move forward 50cm and then return to the original position. stopOmniwheel() function stops robot movement.

moveOmniwheel() uses robot service. robot service becomes bound when GenieApiDemo App is executed. Bound service can call robot API through the IHovisGenieService object mBinder. mBinder is declared in GenieApiApplication as follows.

GenieApiDemoApplication.java LIne 14:

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

GenieApiApplication object is need to approach mBinder defined in GenieApiDemoApplication class. This object is declared in BaseActivity.

BaseActivity.java Line 10:

```
protected GenieApiDemoApplication myRobotApp;
```

Since TestOmniwheelActivity is a child class of BaseActivity, myRobotApp can be used without declaration Therefore, robot service can be approaced using myRobotApp.mBinder.

LIne 61 of moveOmniwheel() checks to see if myRobotApp.mBinder object is null. Since robot service operates externally, binding may become disconnected even though service was bound at the start of the App. This code prevents the error that would be caused by disconnected binding.
Lines 64~72 are codes for using robot service. Since robot service commuicates with remote process, RemoteException may occur. try–catch statement has to be written for robot service exception handling.

Within the section surrounded by try–catch statement there is a section that calls actual robot service API function. Each robot service function is as follows.

dmel_robot_set_pose

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

Set current robot position with x, ,y, theta. If current robot postion is set to (0, 0, 0). current location becomes standard.

Parameters:

cn – Component name
x – (meter)
y – (meter)
theta – (Radian –PI ~ +PI)

Returns:

Success true, fail false return

Throws:

android.os.RemoteException

dmel_navigate_prepare

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

Perform initialization for robot naviation. When initialized, current robot position is set to (0, 0, 0), current Heading Heading is set to 0 degrees.

Parameters:

cn – Component name

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
```

Add robot waypoint. Add robot waypoint. based on standard starting position set in the robot (0, 0, 0) set waypoint. (dmel_navigate_prepare) (reference) robot moves to the set waypoint through dmel_navigate_start().

Parameters:

cn – Component name
x – Standard starting position x (meter)
y – Standard strting position y (meter)
theta – Robot Heading from starting point standard robot heading theta (Radian –PI ~ +PI)
rflag – when true, robot will face theta degree direction when reaching destination
time – robot waiting time after reaching destination (sec)

Returns:

Throws:

android.os.RemoteException

dmel_navigate_start

```
boolean dmel_navigate_start(ComponentName cn)
                             throws android.os.RemoteException
```

Run robot navigation. Robot navigation will visit previously set goal_pose one by one.

Parameters:

cn – Component name

Returns:

Success true, Fail false return

Throws:

android.os.RemoteException

dmel_robot_set_pose() function sets current robot position by x, y, theta. Since position was set as 0, 0, 0, current robot positon on the coordinate grid is set as x coordinate 0, y coordinate 0, and the robot direction is set to 0 degrees. Argument unit for x,y is in m, theta value unit is in radian.

dmel_navigate_prepare() is a function that perfoms initialization for robot naviation. This function is as sam as dmel_robot_set_pose(). (From the example above, one of the two lines can be deleted)
dmel_navigate_add_goal_pose() is used to set the robot waypoint.
dmel_navigate_add_goal_pose() simply adds the waypoint and actual robot movement starts when dmel_navigate_start() function is called.
Among the dmel_navigate_add_goal_pose() arguments, except for the first argument ComponentName, other arguments refer to x m, y m, theta(radian).From the code, 0.5, 0, 0 refers to the direction robot was facing before moving 50cm forward. Fifth argument of the function sets true or false to whether the robot will face the direction set in theta in third argument once the robot arrives at the destination. Sixth argument refers to the wait time (s) at the destination. Codes in our example is true,1 which means robot will face 0 degree direction and wait 1s after it arrives at the destination.
dmel_navigate_add_goal_pose() arguments are 0, 0, 0, true, 1. First three arguments refer to the original postion. Since 4th argument is true, robot will turn and face forward direction once it arrives back at the original position. Fifth argument will make robot wait 1s after arriving at original position. Robot will start to move once dmel_navigate_start() function is called.

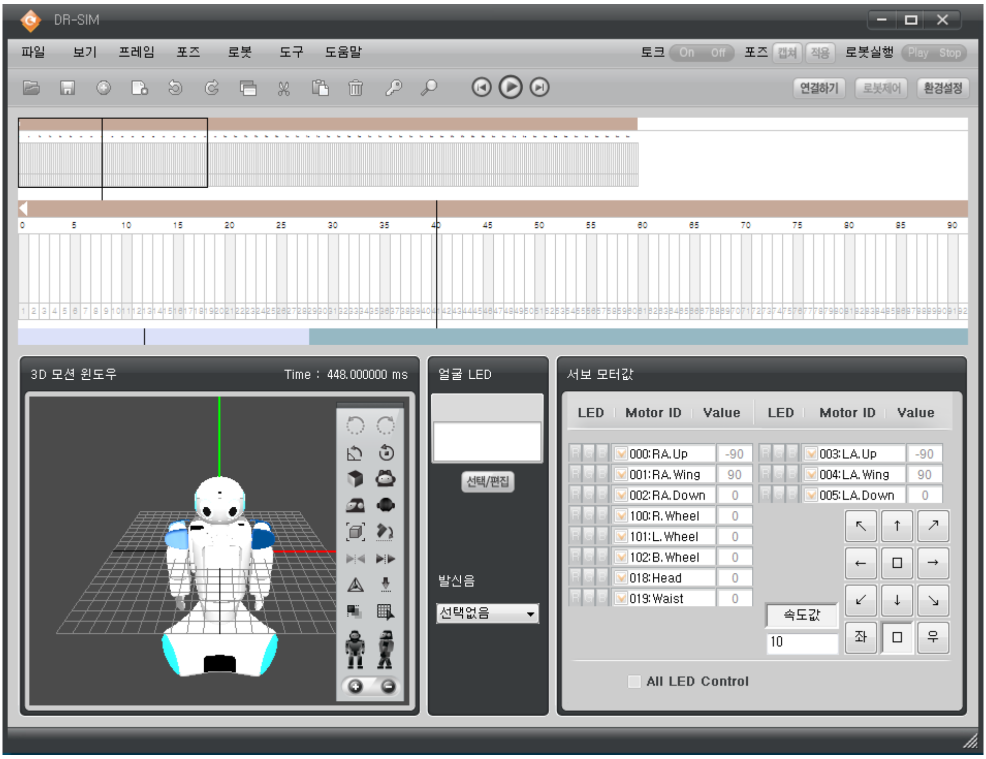
```
myRobotApp.mBinder.dmel_robot_set_pose(getComponentName(), 0, 0, 0);
myRobotApp.mBinder.dmel_navigate_prepare(getComponentName());
myRobotApp.mBinder.dmel_navigate_add_goal_pose(getComponentName(), 0.5, 0, 0, true, 1);
myRobotApp.mBinder.dmel_navigate_add_goal_pose(getComponentName(), 0, 0, 0, true,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

6.3 Motion Control

This example will make Hovis Genie peform motion. Motion control refers to controlling robot motion by controlling robot drivetrain and servomotors making up the robot joints. Hovis Genie has total of 11 motors. There are 3 motors in each arm, waist motor, head motor, and 3 omni wheel drivetrain motors.

Motion control requires control of each motor but controlling each motor indivually makes control difficult and hard to create motion efficiently. Dongbu Robot provides robot motion editor tool DR-SIM to make robot motion creation more easy and efficient. Users are able to edit Hovis Genie motion using DR-SIM.

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



Time : 448.000000 ms

LED	Motor ID	Value	LED	Motor ID	Value
R G B	000:RA.Up	-90	R G B	003:LA.Up	-90
R G B	001:RA.Wing	90	R G B	004:LA.Wing	90
R G B	002:RA.Down	0	R G B	005:LA.Down	0
R G B	100:R.Wheel	0			
R G B	101:L.Wheel	0			
R G B	102:B.Wheel	0			
R G B	018:Head	0			
R G B	019:Waist	0			

속도값: 10

[Diagram 6-4] DR-SIM motion editor program

DR–SIM creates robot motion based on virtual 3D motor and provides timeline to edit robot motion based on time. Motion files created by DR–SIM are used to control Hovis Geni motion. Motion files have extension DMT.(Refer to DR–SIM manual)

Hovis Genie motion files are saved in SD card installed in MID. Robot motion file pathe is /dongbu/motion.

Access to robot motion folder is possible by using MID as USB memory device. From MID Settings 〉 Application 〉 Developer 〉 USB debuggig. Uncheck USB debuggin and connect MID to PC to access SD card in MID. Access /dongbu/motion.

MID motion folder already contains variety of motions. These motions are basic motions. Motion list is seen below.

File Name	Content
01m_turn_l.dmt	Turn from same spot (R)
01m_turn_r.dmt	Turn from same spot (L)
01n_adbomen0.dmt	Touch stomach with one hand
01n_adbomen1.dmt	Touch stomach with both hands
01n_aging.dmt	Motor Aging
01n_bend_weist.dmt	Bend waist
01n_body.dmt	Body wash
01n_clap.dmt	Clap from chest level
01n_command1.dmt	Conduct1
01n_command2.dmt	Conduct 2
01n_command3.dmt	Conduct 3
01n_command4.dmt	Conduct 4
01n_dance.dmt	Dance
01n_face.dmt	Wash face
01n_front_hand.dmt	Face forward, look around
01n_goodjob.dmt	One arm forward(Best!)

File Name	Content
01n_gymnastics.dmt	Exercise
01n_hide.dmt	Cove face with both hands
01n_hurrah0.dmt	Walk around with both arms lifted
01n_hurrah1.dmt	Lift both arms and shake arms L/R: opposite direction
01n_hurrah2.dmt	Lift both arms and shake both arms L/R: same direction
01n_kiss0.dmt	Kiss1 (put one hand to mouth and then remove)
01n_kiss1.dmt	Kiss2 (Put both hands to mouth and then remove)
01n_love.dmt	Love you (Lift both arema and make heart [circle]
01n_muscle	Show off muscle
01n_pos1	Pose 1
01n_scrape.dmt	Rub eye (move one hand l/f in front of the eye)
01n_scratch	Scratch head
01n_self_msg.dmt	Self massage
01n_shake_head0.dmt	Lift one hand and shake
01n_shake_head1.dmt	Open and Shake one's arm(without raising)
01n_shake_head2.dmt	Shake both arms front and back altenately
01n_shake_head.dmt	Fast head shake
01n_sidebyside.dmt	One arm forward
01n_sorrow.dmt	Cry (Move both hands L.R in front of the eyes
01n_stare0.dmt	Look around from same spot
01n_stare1.dmt	Look around with one hand on head
01n_stare2.dmt	Look around with one hand beside the ear
01n_stretching.dmt	Stretch
01n_tekwuando.dmt	Takwondo
01n_tekwuando_wide.dmt	Taekwondo (MID landscape)

File Name	Content
01n_tooth.dmt	Brush teeth
01n_turn_360_0.dmt	Rotate 360 degrees from same spot (once)
01n_turn_360_1.dmt	Rotate 360 degrees from same spot (repeat)
01n_turn_head_l.dmt	Turn head (R)
01n_turn_head_r.dmt	Turn head (L)
01n_turn_love.dmt	Rotate I love you(Lift both arms and make heart (circle))
01n_turn_shake.dmt	Fold both arms at chest and shake wheel L/R(Like shaking body)
01n_up_hand.dmt	LIft one arm
01n_watch_clock.dmt	Look at wrist watch (Bend elbow and move handclose to face)
01n_wave.dmt	Wave both arms
basicpose.dmt	Basic pose
mid_rotate.dmt	Rotate MID from portrait to landscape
motion_highwalk.dmt	Walk fast
motion_slow_walk.dmt	Walk slow
motion_standby.dmt	Standby
motion_walk.dmt	Walk normal

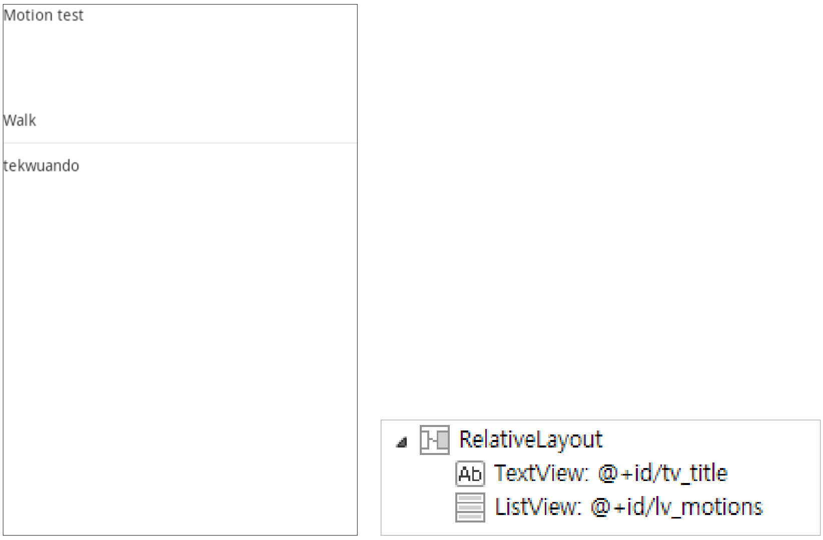
In this example we will use the installed basic motions to run robot motion. Motion control is comprised of activity_testmotion.xml and TestMotionActivity.java. Add two files to the GenieApiDemo project. Edit activity_testmotion.xml as following to create UI for motion control.

```
〈?xml version=”1.0” encoding=”utf-8”?〉
〈RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent” 〉

    〈!— Test Title —〉
    〈TextView
```

```
        android:id=”@+id/tv_title”
        android:layout_width=”match_parent”
        android:layout_height=”85dp”
        android:text=”Motion test”
        /〉

    〈!— Motion list —〉
    〈ListView
        android:id=”@+id/lv_motions”
        android:layout_width=”match_parent”
        android:layout_height=”wrap_content”
        android:layout_below=”@id/tv_title”
        /〉

〈/RelativeLayout〉
```

[Listing 6-7] activity_testmotion.xml

activity_testmotion.xml has 〈RelativeLayout〉 as the highest tag. 〈RelativeLayout〉 contains 〈TextView〉 exoresing test title and 〈ListView〉 to express motion list. UI expressed in screen is shown in [Diagram 6-5].



Motion test

Walk

tekwuando

RelativeLayout

TextView: @+id/tv_title

ListView: @+id/lv_motions

[Diagram 6-5] Motion control

In TestMotionAcitivity.java, Adapter is central to the ListView list selection.Adapter will be used in this example as we used Adapter in Chapter 6.1 to receive multiple data sources when creating GridView. Excpet in Chapter 6.1, In order to receive multiple data sources comprising of image and text, we defined lower widgets below demoitems.xml and created MainAdapter. In this example since we will only be using one Text View such as 'Walk', 'taekwondo', standard ArrayAdapter provided by Android will be used. To define lower widget comprising of single TextView, we will add listitems.xml to /res/layout and add the source.

```
〈?xml version=”1.0” encoding=”utf-8”?〉
〈LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent”
    android:orientation=”vertical” 〉

    〈TextView
        android:id=”@+id/tv_name”
        android:layout_width=”wrap_content”
        android:layout_height=”40dp”
        android:gravity=”center_vertical”
        /〉

〈/LinearLayout〉
```

[Listing 6-7] listitems.xml

In listitems.xml, horizontal shows size of the text string assigned to 〈TextView〉, and vertical 40 dp TextView. Widget apllicable to 'Walk' and 'taekwondo' in [Diagram 6-4]

Android size unit

px – pixel

dp – deviece–independent pixel

Android has many units for expressing the size but dp (dpi) is used most commonly used as widgets defined by px may show up differently depending on the resolution of the device. Use of dp is recommended since widget defined dp will show same size ratio regardless of the resoultion.

Write TestMotionActivity.java in following order referring to [Listing 6–8].

1) Inherit BaseActivity

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

2) Declare widget related member variable

```
 private ListView mMotionListView;
```

3) Create onCreate() – ListView and Adapter

First code that will be executed when TestMotionActivity first appears in the screen. setContentView() function in line 24 calls UI defined in activity_testmotion.xml. In line 26, resource is assigned to member variable mMotionListView through ListView ID.

Line 28~30 assigns text string to ArrayList. Lines 32~35 are the core codes that create ArrayAdapter object adapter. ArrayAdapter construtor and relavant codes are as follows.

```
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);
```

Second argument of the above code is the resource for the previousle declared listitems.xml. Third argument is the TextView ID declared in listitems.xml. Fourth argument is the list string that will be assigned in lines 28~30. These argument values are used by the adapter to maintains data structure and value of the data that will be supplied to ListView. In line 36, data is assigned to ListView by attaching adatpter to the listView. In line 37, Touch event is registered to ListView.

4) Inherit onItemClickListener intreface multiple times and then create onItemClick()

Function processing user touch event. ListView position is counted from 0. In this example 'Walk' is, 'taekwondo' is 1.

```
 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);
```

```

        mMotionListView = (ListView)findViewById(R.id.lv_motions);

        ArrayList〈String〉 arrayList = new ArrayList〈String〉();
        arrayList.add(“Walk”);
        arrayList.add(“tekwuando”);

        ArrayAdapter〈String〉 adapter = new ArrayAdapter〈String〉(this,
                R.layout.listitems,
                R.id.tv_name,
                arrayList);
        mMotionListView.setAdapter(adapter);
        mMotionListView.setOnItemClickListener(this);
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            Intent intent = new Intent(this, MainActivity.class);
            startActivity(intent);
        }
        return super.onKeyDown(keyCode, event);
    }

    public void onItemClick(AdapterView〈?〉 parent, View v, int position, long id) {
        if (myRobotApp.mBinder == null)
            return;

        try {
            myRobotApp.mBinder.dmel_motion_stop(getComponentName());

            switch (position) {
            case 0:
                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() runs when user touches an item in the list. When the item in the list is touched, myRobotApp.mBinder will be in line 50 for service connection error. If no error is found, command stopping current motion will be sent and new motion will start.

```
myRobotApp.mBinder.dmel_motion_stop(getComponentName());
...
myRobotApp.mBinder.dmel_motion_play(getComponentName(), File name);
```

dmel_motion_play

```
void dmel_motion_play(ComponentName cn,
                      java.lang.String motion)
                       throws android.os.RemoteException
```

Play robot motion . (default path /sdcard/dongbu/motion/)
Parameters:
cn – Component name
motion – Motion file name ex) “01n_shake_head.dmt”
Throws:
android.os.RemoteException

dmel_motion_stop

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

Stop current motion
Parameters:
cn – Component name
Throws:
android.os.RemoteException

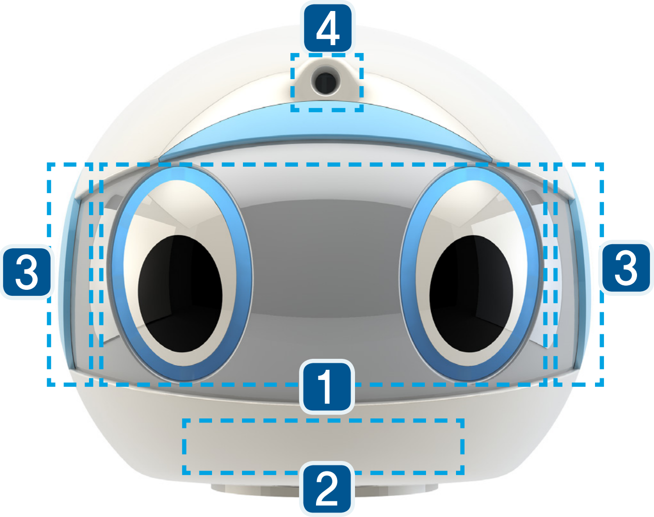
Codes in Lines 56~66 executes motion based on position of the touch on the list. According to the code, motion_walk.dmt motion will be executed when ‘Walk’ is selected and 01n_tekwuando_wide.dmt motion when ‘taekwondo’ is selected.

Word of caution, motion file must exist in SD card /dongbu/motion. Try adding other motions to the list or add your own motion created by DR-SIM.

6.4 Head LED Control

In this chapter, we will learn about controlling Hovis Genie head LEDs. Hovis Geniehas LED contol board installed in the head which can turn on/off the LEDs on the eye, mouth, ear, and forehead. LED composition is as shown in the diagram.

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



[Diagram 6-6] Hovis Genie Head LED position
(1-eye, 2-mouth, 3-ear, 4-forehead)

Eye LEDs are located in position 1. Each eye has 8 red and 8 blue LEDs for total of 16 LEDs for each eye. Red and blue LEDs can be set separately and purple color effect can be created by turning on both red and blue LEDs at the same time.

Mouth LEDs are located in position 2. There are 3 LEDs at the mouth. Ear LEDs are located at position 3. There are 4 LEDs located at ear. Ear LEDs can only be controlled as group of 4. Forehead LEDs are located at position 4. Brightness of forehead LED can be adjusted. Mouth, ear, and forehead LEDs have single color.

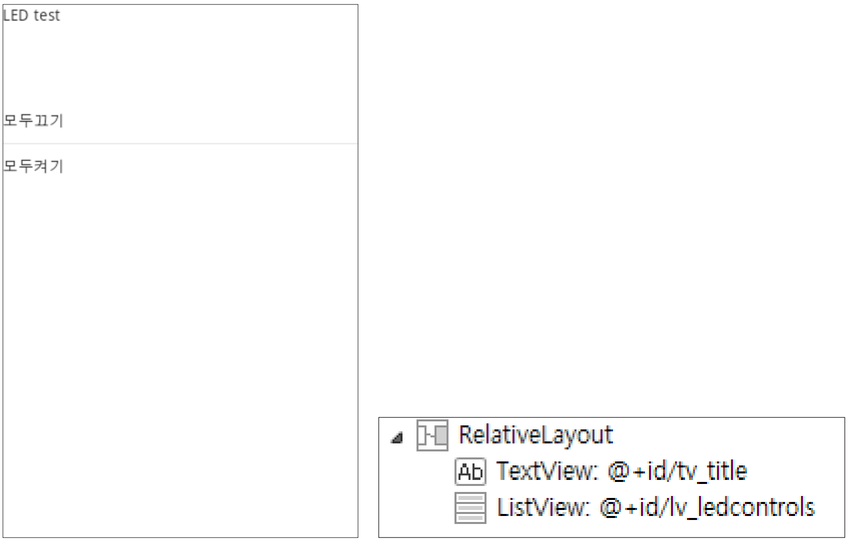
To control the Head LEDs more efficiently, robot service provides functions for modules instead of control for each individual LED. These functions are very useful for expressing robot emotions and visual effects through LEDs.

Add activity_testled.xml and TestLedActivity.java required for Head LED control to the GenieApiDemo project and open activity_testled.xml to create UI for the example.

```
〈?xml version=”1.0” encoding=”utf–8”?〉
〈RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent” 〉

    〈!–– Test Title ––〉
    〈TextView
        android:id=”@+id/tv_title”
        android:layout_width=”match_parent”
        android:layout_height=”85dp”
        android:text=”LED test”
        /〉

    〈!–– LED control list ––〉
    〈ListView
        android:id=”@+id/lv_ledcontrols”
        android:layout_width=”match_parent”
        android:layout_height=”wrap_content”
        android:layout_below=”@id/tv_title”
        /〉

〈/RelativeLayout〉
```

[Listing 6–7] activity_testmotion.xml

Head LED control UI is very similar to Motion control UI in Chapter 6.3 Screen is same as [Diagram 6–7].



LED test

모두끄기

모두켜기

RelativeLayout

TextView: @+id/tv_title

ListView: @+id/lv_ledcontrols

[Diagram 6–7] Head LED control

Create lower widget which will become an item in the ListView. , Add text string datat from onCreate() and create Adapter. Register Adapter to the ListView to make it appear on the screen. Set touch event to the ListView amd jnherit onItemClickListener interface to create onItemClick().

istitems.xml assigning lower widgets in ListView was already created in Chapter 6.3 so it will be skipped in this chapter. TestLedActivity.java is also similar. Refer to Chapger 6.3.

```
package com.dongburobot.genieapidemo;

import java.util.ArrayList;

import android.content.Intent;
import android.os.Bundle;
import android.os.RemoteException;
import android.view.KeyEvent;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
```

```
public class TestLedActivity extends BaseActivity implements OnItemClickListener {

    private ListView mLedControlListView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_testled);

        mLedControlListView = (ListView)findViewById(R.id.lv_ledcontrols);

        ArrayList<String> arrayList = new ArrayList<String>();
        arrayList.add(“All off”);
        arrayList.add(“All on”);

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

    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            Intent intent = new Intent(this, MainActivity.class);
            startActivity(intent);
        }
        return super.onKeyDown(keyCode, event);
    }

    public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
        turnAllLedOff();
        try {
            Thread.sleep(300);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        switch (position) {
        case 0:
            turnAllLedOff();
            break;
        case 1:
            turnAllLedOn();
            break;
        default:
            break;
        }

    }


    // beam 0 : Off, 1 : On weak, 2 : On medium, 3 : On strong
    // eyeCode – 0 : Maintain status, 1 : Round & Round, 2 : Blink eyebrow, 3 : Stop eyebrow,
    //            4 : All on, 5 : All blink, 6 : L/R effect, 7 : Top/bottom effect, 8 : Off

    // earCode – 0 : Maintain status, 1 : On, 2 : Off, 3 : Blink
    // colorCode – 1 : RED LED, 2 : BLUE LED, 3 : PURPLE (RED + BLUE) LED
    // mouthCode – 0 : Maintain status, 1 : Middle ON, 2 : 3 ON, 3 : Middle blink, 4 : All 3 blink, 5 : Talk, 6: Off
    private void turnAllLedOff() {
        if (myRobotApp.mBinder == null)
            return;

        try {
            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);
        } catch (RemoteException e) {
            e.printStackTrace();
        }

    }

    private void turnAllLedOn() {
        if (myRobotApp.mBinder == null)
            return;

        try {
            myRobotApp.mBinder.dmel_hri_set_mouth_led(getComponentName(), 2);
           myRobotApp.mBinder.dmel_hri_set_brow_beam(getComponentName(), 2);
            myRobotApp.mBinder.dmel_hri_set_ear_led(getComponentName(), 1);
            myRobotApp.mBinder.dmel_hri_set_eye_led(getComponentName(), 4, 2);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

[Listing 6-10] TestLedActivity.java

In this example, ListView items are 'All off' and 'All on'. Each touch will call turnAllLedOff() and turnAllLedOn() in lines 57~65 of onItemClick() to turn on/off all HEad LEDs. turnAllLedOff() and turnAllLedOn() calls following 4 functions. These LED functions have some parameters that control robot's eye, mouth, ear and forehead LEDs. The functions are as followings.

dmel_hri_set_brow_beam

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

Setup forehead LED.
Parameters:
cn – Component name
brightness – 0 : Off, 1 : Weak On, 2 : Medium On, 3 : Bright On
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
```

Setup eye LED.
Parameters:
cn – Component name
eyeCode – 0 : maintain status, 1 : Round & Round, 2 : Eyebrow blink, 3 : Eyebrow stop, 4 : All On, 5 : All Blink, 6 : L/R effect, 7 : Top/bottom effect, 8 : Off
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
```

Ear LED setup.
Parameters:
cn – Component name
earCode – 0 : maintain status, 1 : On, 2 : Off, 3 : Blink
Throws:
android.os.RemoteException

dmel_hri_set_mouth_led

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

Mouth LED setup.
Parameters:
cn – Component name
mouthCode – 0 : maintain status, 1 : Middle ON, 2 : 3 ON, 3 : Middle blink, 4 : 3 blink, 5 : Talk
Throws:
android.os.RemoteException

Mouth, forehead, and ear LEDs are single colored and robot service function does not have argument for color section.Eye LED control function dmel_hri_set_eye_led() has argument of selecting colors since eye LEDs are comprised of 8 red LEDs and 8 blue LEDs for each ye. (Puple color effect appears when both Red and Blue LEDs are on). Since controlling 32 eye LEDs individually become very complicated, LEDs are controlled by playing pre defined effects.

Code values for each function argument is as follows [Table 6–3].

Code Vlaue	Eye LED	Eye Color	Ear	Mouth	Forehead
0	Maintain status		Maintain status	Maintain status	Off
1	Round & Round	Red	On	Middle ON	Weak
2	Blink eyebrow	Blue	Off	3 ON	Medium
3	Stope eyebrow	Purple	Blink	Middle blink	Strong
4	All On			3 blink	
5	All Blink			Talk	
6	L/R effect			Off	
7	Up/Down effect				
8	Off				

Examine turnAllLedOff() in lines 82~85 which turn off all LEDs using [6–3]as reference.

```
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)
```

In line82, code value assigned as dmel_hri_set_mouth_led() function argument value is 6. In [Table6–2] code 6 for mouth is off, therefore all three mouth LEDs will be turned off. Code value for hri_set_ear_led() in line 83 is 2. This code value will turn off all ear LEDs. dmel_hri_set_brow_beam() in next line is forehead LED brightness function and the code value 0 will turn off the LED. dmel_hri_set_eye_led() argument in last line had eye LED code 8, color code value 1. Eye LED code value 8 turns off the LEDs, therefore Eye color value of 1 has no effect. assign any color to the Eye LEDs even though all Eye ELDs will be turned off.
Analyze turnAllLedOn() in linese 97~100 using [Table 6–2] and you will notice that it will turn on all LEDs.

```
myRobotApp.mBinder.dmel_hri_set_mouth_led(getComponentName(), 2);
myRobotApp.mBinder.dmel_hri_set_brow_beam(getComponentName(),2);myRobotApp.
mBinder.dmel_hri_set_ear_led(getComponentName(),  1);  myRobotApp.mBinder.dmel_
hri_set_eye_led(getComponentName(), 4, 2);
```

Try adding other items to the screen list to experiment with different effcts. If middle mouth LED blinks and ear and mouth LEDs can't be controlled. This signfies LED control board error. Board will be automatically restored in robot autonmous mode.

6.5 TTS Control

TTS refers tp Text–to–Speech and changes text to robot voice. For example, When robot receives text input “Hello” it will output “Hello” by voice. TTS function is used in majority of the robots as it allows robot to output voice like human.

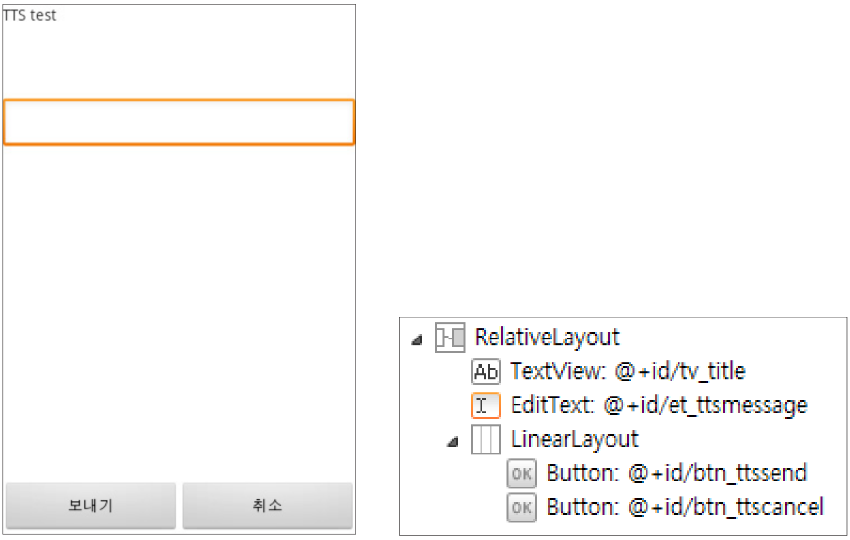
Hovis Genie has Korean TTS engine installed. This example will use Korean TTS to test robot voice function.

Add activity_testtts.xml for creating UI in TTS control Activity in /res/layout and add matching TestTTSActivity.java in /src/com/dongburobot/genieapidemo. Open activity_testtts.xml file and edit the code using [Listing 6–11] as reference.

```
〈?xml version=”1.0” encoding=”utf–8”?〉
〈RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent” 〉

    〈!— Test title ––〉
    〈TextView
        android:id=”@+id/tv_title”
        android:layout_width=”match_parent”
        android:layout_height=”85dp”
        android:text=”LED test”
        /〉

    〈!— LED control list ––〉
    〈EditText
        android:id=”@+id/et_ttsmessage”
        android:layout_width=”match_parent”
        android:layout_height=”wrap_content”
        android:hint=”Input your message please..”
        android:maxLength=”140”
        android:maxLength=”140”
        android:layout_below=”@id/tv_title”
        /〉

        〈!— Start and end button —〉
        〈!— Start and end button —〉
    〈LinearLayout
        android:layout_width=”match_parent”
        android:layout_height=”wrap_content”
        android:layout_alignParentBottom=”true”〉
        〈Button
            android:id=”@+id/btn_ttssend”
            android:layout_width=”160dp”
            android:layout_height=”wrap_content”
            android:text=”send”
            /〉
        〈Button
            android:id=”@+id/btn_ttscancel”
            android:layout_width=”160dp”
            android:layout_height=”wrap_content”
            android:text=”cancle”
            /〉
    〈/LinearLayout〉
〈/RelativeLayout〉
```

[Listing 6–7] activity_testmotion.xml

〈RelativieLayout〉 is the highest tag in TTS contrl screen layout. Place each widget in relative position to other widgets.In case of 〈TextView〉 which does not have relative position property, it will be placed at top left of the screen.〈EditText〉 will be placed below 〈TextView〉 as assigned by the property in line 21. 〈LinearLayout〉 will be placed at bottom left as assigned by the property in line 28. 〈LinearLayout〉 contains two lower 〈Button〉 widgets. These 〈Button〉 widgets position is influenced by the 〈LinearLayout〉 and they will be placed at bottom of the screen. [Diagram 6–8] shows structure with widgets.



- RelativeLayout
  - TextView: @+id/tv_title
  - EditText: @+id/et_ttsmessage
  - LinearLayout
    - Button: @+id/btn_ttssend
    - Button: @+id/btn_ttscancel

[Diagram 6–8] TTS Control Screen

In TTS control screen, user input is done through 〈EditText〉. 〈EditText〉 widget shown in [Diagram 6–8]is surrounded by gold colored box. Touching the 〈EditText〉will display keyboard screen to input text. Pressing Back button will make the keyboard disappear. 〈EditText〉 input can be up to 140 characters as sepecified byt the android:maxLength=”140” propety in line 20. If 〈EditText〉 text input goes over to next line due to large amount of text, android:layout_height=”wrap_content” will increase the size of 〈EditTex〉 by size of the content.

UI for example created. Open TestTTSActivity.java and edit the source code as follows.

1) Inherit BaseActivity

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

2) Declare widget related member variable

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

3) Create onCreate()

onCreate() loads layout XML, assigns resource to each widget member variable, and sets up touch event to each button.

4) Inherit OnClickListener interface multiple times and then create onClick()

This example has one 'Send' and one 'Cancel' button. Create codes to process touch event when button is pressed. First inherit OnClickListener interface multiple times.

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

Add 'implements OnClickListener' and place mouse cursor on top of 'TestTTSActivity' to add onClick().

5) Create playTTS(), stopTTS()

Create methods playTTS() and stopTTS() for playing and cancelling TTS. These two methods are called by onClick() when usere presses 'Send' or 'Cancel' button.

6) Create onBackPressed()

In previous examples Back button was processed using onKeyDown() overdrive. ofonKeyDown() is called regardless of the key pressed and and Back button press is determined and processed within the function. onBackPressed() is called only when Back button is pressed. Robot App is designed so that only one Activity can be maintained and prevents other Activity from being saved in the stack and Back button press can end the App. onBackPressed() prevents Back button press from ending the App and switches to main screen when button is pressed.

```
package com.dongburobot.genieapidemo;

import android.content.Intent;
import android.os.Bundle;
import android.os.RemoteException;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class TestTTSActivity extends BaseActivity implements OnClickListener {

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

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_testtts);

        mTTSEditText = (EditText)findViewById(R.id.et_ttsmessage);

        mBtnTTSSend = (Button)findViewById(R.id.btn_ttssend);
        mBtnTTSCancel = (Button)findViewById(R.id.btn_ttscancel);
        mBtnTTSSend.setOnClickListener(this);
        mBtnTTSCancel.setOnClickListener(this);
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

```

```
public void onClick(View v) {
    switch (v.getId()) {
    case R.id.btn_ttssend:
        playTTS(mTTSEditText.getText().toString());
        break;
    case R.id.btn_ttscancel:
        stopTTS();
        break;

    default:
        break;
    }
}

private void playTTS(String msg) {
    stopTTS();

    if (myRobotApp.mBinder == null)
        return;

    try {
        myRobotApp.mBinder.dmel_tts_speak(getComponentName(), msg);
    } catch (RemoteException e) {
        e.printStackTrace();
    }
    return;
}

private void stopTTS() {
    if (myRobotApp.mBinder == null)
        return;

    try {
        myRobotApp.mBinder.dmel_tts_stop(getComponentName());
        Thread.sleep(200);
    } catch (RemoteException e) {
```

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

[Listing 6-12] TestTTSActivity.java

Core of this example is in lines 51~79 where playTTS() and stopTTS() is called by onClick().

playTTS() receives String as argument. In the example, when the user presses 'Send' button, Text string entered in EditText will be converted to String format and entered as argument. Line 40.

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

This is sent to TTS related robot service function dmel_tts_speak() in line 58 where tex is coverted to voice.

```
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
```

Play TTS(Text-to-Speech).
Parameters:
cn – Component name
text – text string to be converted to voice String ex)"Hello. I am Hovis Genie."
Throws:
android.os.RemoteException

In line 52, stopTTS() is executed to gurantee precise function of TTS. TTS command priority lies with which ever TTS reserved the robot resource fist. Therefore, if TTS is already running, Robot service will ignore the incoming TTS command.

stopTTS() in lines65~78 stops current TTS command. stopTTS() executes dmel_tts_stop() internally. dmel_tts_stop() stops TTS but some dealy is possible. Therefore, to insure safe operation Thread.sleep(200) can be used to add little bit of delay.

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

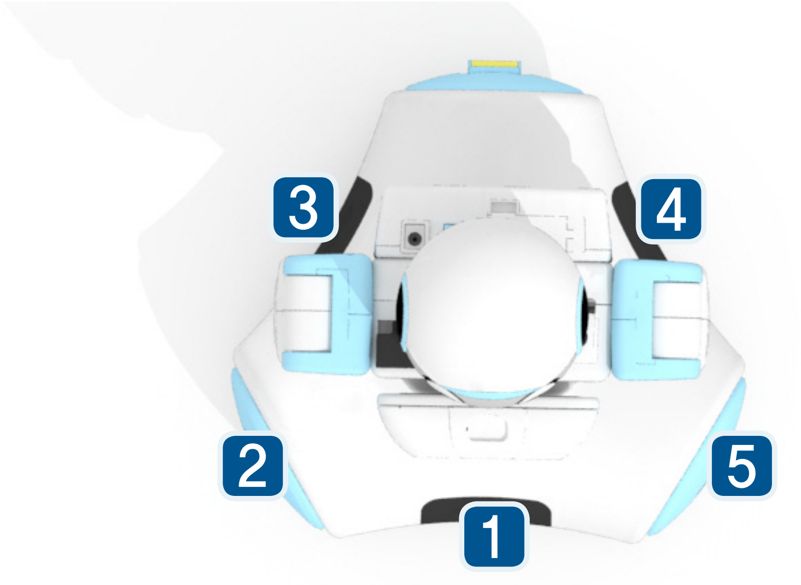
dmel_tts_stop

```
void dmel_tts_stop(ComponentName cn)
                    throws android.os.RemoteException
Stop TTS(Text-to-Speech).
Parameters:
cn - Component name
Throws:
android.os.RemoteException
```

We used playTTS() and stopTTS() which use robot service functions to control TTS. TTS has high usability and can make robot look intelligent. However, TTS voice will not sound like human. To improve pronounciation, text could be written according to how each word sounds.

6.6 PSD Sensor Control

Hovis Genie has 5 PSD(Position Sensitive Device) sensors to recognize surrounding environment. PSD sensor has maximum detection rang of 80cm to detech obstacles. Robot can use the PSD sensors to avoid obstacles or used to create interesting movment strategies.



[Diagram 6-9] Distance sensor placement

Hovis Genie has five 1~5 PSD sensors counted clockwise direction starting from the front. This example will receive PSD values in real time and output them to the screen.

activity_testpsdsensor.xml is the PSD sensor control UI file and applicable source isTestPSDActivity.java. Add the applicable file to the GenieApiDemo project and open the activity_testpsdsensor.xml file to edit the codes as following.

```
〈RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    xmlns:tools=”http://schemas.android.com/tools”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent” 〉

    〈!— Text Title —〉
    〈TextView
        android:id=”@+id/tv_title”
        android:layout_width=”match_parent”
        android:layout_height=”85dp”
        android:text=”PSD Sensor test”
        /〉

    〈LinearLayout
        android:layout_width=”match_parent”
        android:layout_height=”wrap_content”
        android:layout_below=”@id/tv_title”
        android:orientation=”vertical”
        〉
        〈!— #1 PSD ——〉
        〈LinearLayout
            android:layout_width=”match_parent”
            android:layout_height=”wrap_content”
            〉
            〈TextView
                android:layout_width=”70dp”
                android:layout_height=”wrap_content”
                android:text=”Front”
                /〉
            〈TextView
                android:id=”@+id/tv_psd1”
                android:layout_width=”40dp”
                android:layout_height=”wrap_content”
                /〉
            〈SeekBar
                android:id=”@+id/seekbar_psd1”
            android:layout_width=”200dp”
            android:layout_height=”wrap_content”
            android:max=”80”
            /〉
〈/LinearLayout〉

〈!— #2 PSD —〉
    〈LinearLayout
    android:layout_width=”match_parent”
    android:layout_height=”wrap_content”
    〉
    〈TextView
        android:layout_width=”70dp”
        android:layout_height=”wrap_content”
        android:text=”Front right”
        /〉
    〈TextView
        android:id=”@+id/tv_psd2”
        android:layout_width=”40dp”
        android:layout_height=”wrap_content”
        /〉
    〈SeekBar
        android:id=”@+id/seekbar_psd2”
        android:layout_width=”200dp”
        android:layout_height=”wrap_content”
        android:max=”80”
        /〉
〈/LinearLayout〉

    〈!— #3 PSD —〉
    〈LinearLayout
    android:layout_width=”match_parent”
    android:layout_height=”wrap_content”
    〉
    〈TextView
```

```
            android:layout_width=”70dp”
            android:layout_height=”wrap_content”
            android:text=”Back right”
            />
        〈TextView
            android:id=”@+id/tv_psd3”
            android:layout_width=”40dp”
}

        〈SeekBar
            android:id=”@+id/seekbar_psd3”
            android:layout_width=”200dp”
            android:layout_height=”wrap_content”
            android:max=”80”
            /〉
    〈/LinearLayout〉

        〈!— #4 PSD —〉
        〈LinearLayout
        android:layout_width=”match_parent”
        android:layout_height=”wrap_content”
        〉
        〈TextView
            android:layout_width=”70dp”
            android:layout_height=”wrap_content”
            android:text=”Back left”
            /〉
        〈TextView
            android:id=”@+id/tv_psd4”
            android:layout_width=”40dp”
            android:layout_height=”wrap_content”
            /〉
        〈SeekBar
            android:id=”@+id/seekbar_psd4”
            android:layout_width=”200dp”
            android:layout_height=”wrap_content”
            android:max="80"
            />
    〈/LinearLayout〉

        〈!— #5 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〉

〈!-- Start and Stop button —〉
〈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=”Start”
146:              /〉
147:          〈Button
148:              android:id=”@+id/btn_stop”
149:              android:layout_width=”160dp”
150:              android:layout_height=”wrap_content”
151:             android:text=”End”
152:              /〉
153:      〈/LinearLayout〉
154:
155: 〈/RelativeLayout〉
```

[Lisiting 6–11] activity_testpsdsensor.xml

This PSD control UI source code is quite long but structure is simple due to repeating widgets.
Highest tag is 〈RelativeLayout〉. Tags below 〈RelatvieLayout〉 are 〈TextViw〉 in lines 7~12, , 〈LinearLayout〉 in lines14~134, and 〈LinearLayout〉 in lines 137~153. 〈TextView〉 outputs “PSD Sensor test”, 〈LinearLayout〉 expresses 5 PSD information, 〈LinearLayout〉 expresses ‘Start’ and ‘end’ button.

Tags contained in 〈RelativeLayout〉 expresses widget position in relative terms. 〈TextView〉 in lines 7~12 does not have layout proprety and it is placed top left of the parent tag. 〈LinearLayout〉 in lines 14~134 is placed below 〈TextView〉 according to the property in line 17. In accordance with the property in line 140s, 〈LinearLayout〉 in lines 137~135 is placed at lowest section of parent 〈RelativeLayout〉.

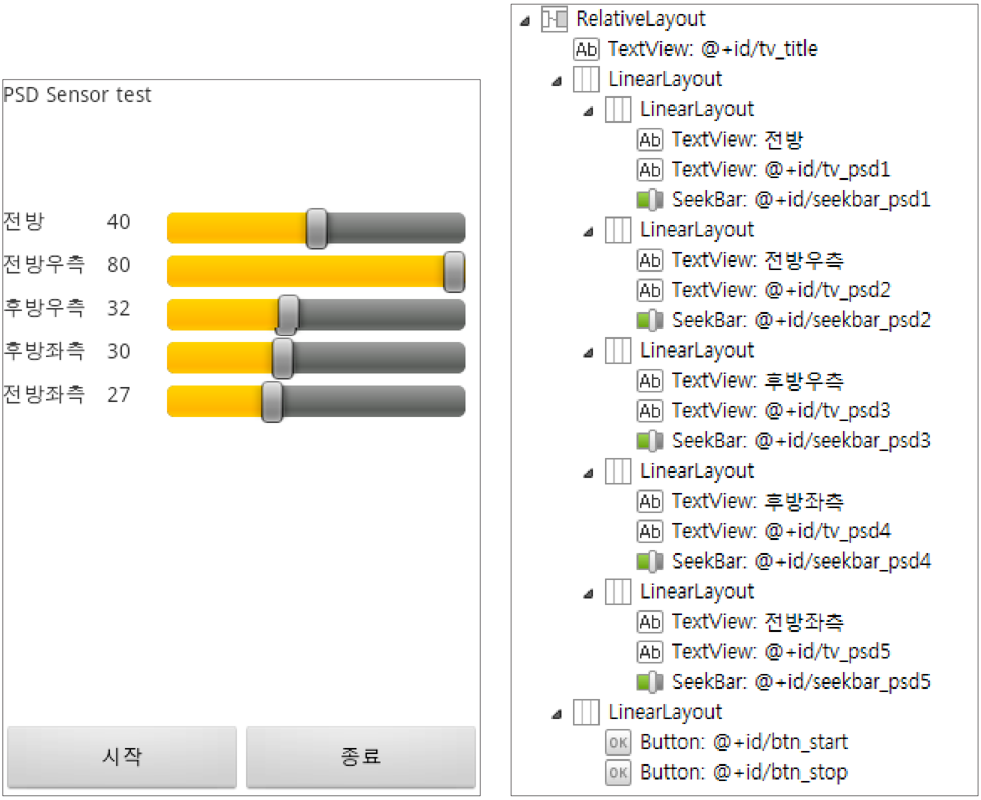
〈LinearLayout〉 in lines14~134 which is lower tag of 〈RelativeLayout〉 includes 5 〈LinearLayout〉 which outpus #1~5 PSD sensor information. 5 〈LinearLayout〉 is placed verically. This is determined by the parent tag 〈LinearLayout〉 property in line 18.

5 〈LinearLayout〉 has 3 widgets, 〈TextView〉, 〈TextView〉, and 〈SeekBar〉 as lower tags used for actual PSD information output.Each 〈LinearLayout〉 does not have vertical or horizontal position setup, therefore included 3 widgets are placed horizontally.

As 5 〈LinearLayout〉 with included 3 widgets are identical except for the ID. we will look at 〈LinearLayout〉 only. 〈LinearLayout〉 in lines 21~41 outputs #1 PSD information. First lower tag 〈TextView〉 expresses sensor placement postion by text. 2nd 〈TextView〉 outputs measured distance by PS sensor in cm unit. 〈SeekBar〉 expresses distance value in graphics. 〈SeekBar〉 has property android:max="80"which sets 〈SeekBar〉 range from 0~80. 80 is the maximum distance PSD sensor can measure.

〈LinearLayout〉 is located at the bottom. 〈LinearLayout〉 in lines 137~153 contains two 〈Button〉 widgets. 〈LinearLayout〉 does not have vertical or horizontal property, therefore buttons are placed horizontally.

UI is shown in [Diagram 6-10].



PSD Sensor test

전방 40

전방우측 80

후방우측 32

후방좌측 30

전방좌측 27

```
RelativeLayout
    TextView: @+id/tv_title
    LinearLayout
        LinearLayout
            TextView: 전방
            TextView: @+id/tv_psd1
            SeekBar: @+id/seekbar_psd1
        LinearLayout
            TextView: 전방우측
            TextView: @+id/tv_psd2
            SeekBar: @+id/seekbar_psd2
        LinearLayout
            TextView: 후방우측
            TextView: @+id/tv_psd3
            SeekBar: @+id/seekbar_psd3
        LinearLayout
            TextView: 후방좌측
            TextView: @+id/tv_psd4
            SeekBar: @+id/seekbar_psd4
        LinearLayout
            TextView: 전방좌측
            TextView: @+id/tv_psd5
            SeekBar: @+id/seekbar_psd5
    LinearLayout
        Button: @+id/btn_start
        Button: @+id/btn_stop
```

[Diagram 6-10] PSD sensor control screen

Open TestPSDActivity.java and start programming to control PSD senspr. Guide for writing source code is shown below.

1) Inherit BaseActivity

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

2) Widget related member variable declaration and Handler declaration

Refer to lines17~23 and declare widget and handler object as member variable. mTvPSD is a TextView object variable which is used to putput each PSD value to TextView, mSbPSD is a object variable used to output distance value to SeekBar. mBtnStart are mBtnStop button objects for controlling buttons.

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

Handler is the core subject of this example. Handler receives PSD sensor value in real time and updates them on the screen.

Handler and Thread has similar function. Within the prcocess, Thread executes an independnt job in using parallel process. There is a need for thread function in this example since PSD sensor value has to be read continuously and updated on the screen in real time and also user touch input has to be processed.

Unfortunately, Thread has a major defect that prevents us from using it in our example. Thread is not stable in GUI(Graphic User Interface) and causes error in majority of cases when Thread is used for screen related work in Android. Handler is used to prevent this kind of error by thread. Handler contains Thread and Thread Queue internally. Refer to Android developers site for more detailed information concerning Handler. For now, it is enough know that Handler will be always used for UI related work.

3) Create onCreate()

onCreate() loads layout XML file from lines 28~45. assigns resource to each widget member variables. and sets up touch event for each button.

Most important sectio of onCreate() is Handler in lines 47~71. Registered Handler starts and ends by message. This will be explained further after coding.

4) Inherit OnClickListener interface multiple times and create onClick()

This example has one"Start" button and one"End" button. "Start" starts PSD sensor value reading and "End" ends the process. Inherit onClickListener interface multiple times and create onClick().

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

Add 'implements OnClickListener' from Eclipse and place the mouse cursor over 'TestPSDActivity' to create onClick(). Depending on the pressed button, onClick() will send message to the Handler or end.

5) Create onPause()

onPause() is a method that is called when Activity ends. In this PSD control Activity, codes will be added to end Handler process when Activity ends.

6) Create onBackPressed()

Robot App is designed to maintain only one activity and prevent other Activity from being saved in the stack and APP could end when Back button is pressed. onBackPressed() prevents Activity from ending when Back button is pressed and switches screen to main screen.

```
package com.dongburobot.genieapidemo;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.RemoteException;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;

public class TestPSDSensorActivity extends BaseActivity implements OnClickListener {

    private TextView[] mTvPSD = new TextView[5];
    private SeekBar[] mSbPSD = new SeekBar[5];

    private Button mBtnStart;
    private Button mBtnStop;

    private Handler mHandler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_testpsdsensor);

        mTvPSD[0] = (TextView)findViewById(R.id.tv_psd1);
        mTvPSD[1] = (TextView)findViewById(R.id.tv_psd2);
        mTvPSD[2] = (TextView)findViewById(R.id.tv_psd3);
        mTvPSD[3] = (TextView)findViewById(R.id.tv_psd4);
        mTvPSD[4] = (TextView)findViewById(R.id.tv_psd5);

        mSbPSD[0] = (SeekBar)findViewById(R.id.seekbar_psd1);
        mSbPSD[1] = (SeekBar)findViewById(R.id.seekbar_psd2);
        mSbPSD[2] = (SeekBar)findViewById(R.id.seekbar_psd3);
        mSbPSD[3] = (SeekBar)findViewById(R.id.seekbar_psd4);
        mSbPSD[4] = (SeekBar)findViewById(R.id.seekbar_psd5);

        mBtnStart = (Button)findViewById(R.id.btn_start);
        mBtnStop = (Button)findViewById(R.id.btn_stop);
        mBtnStart.setOnClickListener(this);
        mBtnStop.setOnClickListener(this);

        mHandler = new Handler() {
            @SuppressLint(“HandlerLeak”)
            public void handleMessage(android.os.Message msg) {
                if (myRobotApp.mBinder == null)
                    return;

                double[] psdValue = null;

                try {
                    psdValue =
                    myRobotApp.mBinder.dmel_robot_get_obs(getComponentName());
                } catch (RemoteException e) {
                    e.printStackTrace();
                }

                //UI Return
                for (int i = 0; i 〈 5; i++) {
                    if ( (int)(psdValue[i]*100) != 100 ) {
                        mTvPSD[i].setText( String.valueOf((int)(psdValue[i]*100)) );
                        mSbPSD[i].setProgress((int)(psdValue[i]*100));
                    }
                }
                mHandler.sendEmptyMessageDelayed(0, 100);
            };
        };
    }

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

        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    @Override
    protected void onPause() {
        if (mHandler != null) {
            if (mHandler.hasMessages(0))
                mHandler.removeMessages(0);

            mHandler = null;
        }

        super.onPause();
    }

    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.btn_start:
            mHandler.sendEmptyMessage(0);
            break;

        case R.id.btn_stop:
            if (mHandler.hasMessages(0))
                mHandler.removeMessages(0);
            break;

        default:
            break;
        }
    }
}
```

[Listing 6-14] TestPSDActivity.java

onCreate() is called whenTestPSDActivity appears in the screeen.Handler is registered in onCreate()

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

In the example, “Start” button is used to send message to the registered Handler.”Start” button is defined within onClick() line 96 and is as shown below.

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

mHandler.sendEmptyMeesage() sends message to the registered Handler. sendEmptyMessage() argument value 0 distinguishes each message when multiple messages are sent to the Handler. In this example, there is only one message so it does not have much meaning in our example. When there are multiple messages, msg.what is used from handleMessage() to distinquish the messages.There are other ways of sending mesaage other than sendEmptyMessage(). Refer to the follwoing link (http://developer.android.com/reference/android/os/Handler.html)

When message is sent by ‘Start’ button, handleMessage() is executed. Service binding is checked in line50. In line 56, robot service function dmel_robot_get_obs() that returns PSD sensor value is called.봇 서비스 함수 를 호출합니다.

dmel_robot_get_obs

```
double[ ] dmel_robot_get_obs(ComponentName cn)
                    throws android.os.RemoteException
```

Return front detection PSD sensor value. From front to clockwise direction 0, 1, 2, 3, 4
Parameters:
cn – Component name
Returns:
Retun 5 front detection sensor values to double[0] ~ double[4]
Throws:
android.os.RemoteException

dmel_robot_get_obs() returns double format line with size 5. #11~5 PSD sensor values are assingned to lines 0~4. Values returned to each line has m unit. TextView and SeekBar that outputs each PSD sensor value uses cm unit for distance value. Lines 62~67 converts m to cm.

In line 68 of handleMessage(), message is sent to Handler once again to repeat the same routine. This is to continuously update the PSD sensor values to the screen.

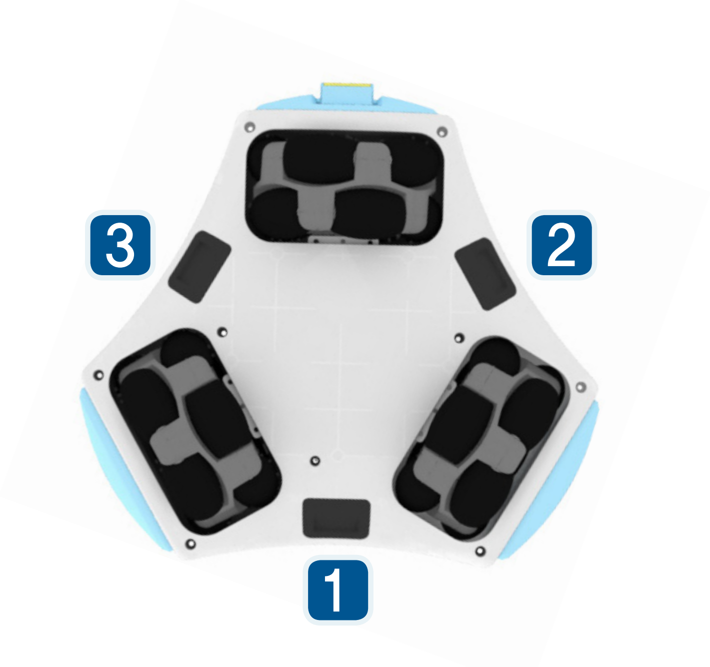
```
68: mHandler.sendEmptyMessageDelayed(0, 100);
```

When Handler object is assigned to mHandler, method handleMessage() has to be redefined. handleMessage() is called when Handler receives a message and has android.os.Message msg as argument to distinquish the message.

In the example, “Start” button is used to send message to the registered Handler.”Start” button is defined within onClick() line 96 and is as shown below.

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

Line 100 checks if message distinguished as 0 is bens sent to the Handler. If message exists, applicable message is deleted in line 101. After, message will no longer be sent to the Handler and handleMessage() will not be called which will end PSD value reading and screen update. Hovis Genie has 3 more PSD sensors to detect the ground. Lower sensor locations are shown below.



[Diagram 6-11] Lower PSD sensor position

Lower detection sensor robot service is dmel_robot_get_cliff()

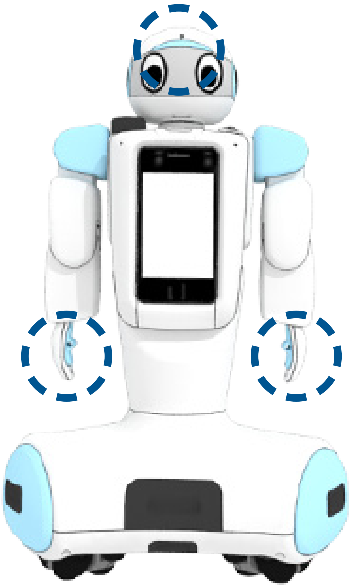
dmel_robot_get_cliff

double[] dmel_robot_get_cliff(ComponentName cn)
throws android.os.RemoteException
Return ground detection PSD sensor value. From front to clockwise direction 0,1,2
Parameters:
cn - Component name
Returns:
Return 3 ground detection sensor values to double[0] ~ double[2
Throws:
android.os.RemoteException

Use of lower sensors are almost identical to the example in this chapter and will be let to the user to experiment with. Add GenieApiDemo XML and Java file to create new Activity to experiment with the lower sensors. Another idesa is to create new icons and txt to the Gridview in MainActiviy. Don't forget to add new Activity to AndroidManifest.xml.

6.7 Touch Sensor Control

In this example, we will learn about Hovis Genie touch sensor control. Touch sensors can be used for interaction between the user and the robot. Genie has total of 3 touch sensors located at head and at palm of both hands.



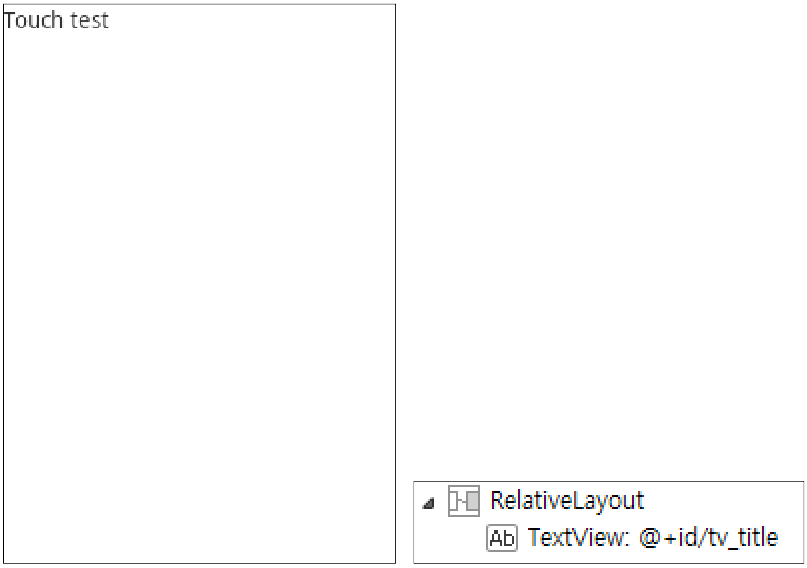
[Diagram 6-12] Touch sensor locaations (both palms, head)

Add activity_testtouchsensor.xml for creating UI in touch sensor Activity to /res/layout. Also, add related TestTouchSensorActivity.java to /src/com/dongburobot/genieapidemo.

This example will express touch to the palms and head using TTS. UI structure is very simple as touch is not expressed to UI. Set 〈RelativeLayout〉 as highest tag and put 'Touch test' in 〈TextView〉 to show current Activiy is touch sensor example. Edit activity_testtouchsensor.xml codes as following.

```
〈?xml version=”1.0” encoding=”utf-8”?〉
〈RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent” 〉

    〈!-- Test Title --〉
    〈TextView
        android:id=”@+id/tv_title”
        android:layout_width=”match_parent”
        android:layout_height=”85dp”
        android:text=”Touch test”
        /〉

〈/RelativeLayout〉
```

[Listing 6-15] activity_testtouchsensor.xml



[Diagram 6-13] Touch sensor control screen

Next, pen TestTouchSensorActivity.java file and write source codes as following.

1) Inherit BaseActivity

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

2) Declare Handler member variable object

```
private Handler mHandler;
```

3) Create onCreate()

onCreate() uses setContentView() to load XML file. Since this example does not change the screen when UI is executed there is no need to assign widget member variable for resouces declared in activity_testtouchsensor.xml.
As we used Handler to received PSD sensor value in real time in previous Chapter, we will again use Handler to observe the sensors in real time to determine the touch status of each sensor.Assign Handler object to mHandler and override handleMessage().

4) Create onPause() To gurantee Handler closing when Activity ends, delete any messages in Handler.

5) Create onBackPressed()

Pressing the Back button may end the App since robot App is designed to maintain only one Activity and prevents other Activity from being saved in the Stack. onBackPressed() prevents Back button from ending the App and switches the screen to main screen.

```
package com.dongburobot.genieapidemo;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.RemoteException;

public class TestTouchSensorActivity extends BaseActivity {

    private Handler mHandler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_testtouchsensor);

        mHandler = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);

                if (myRobotApp.mBinder == null)
                    return;

                boolean[] isHandTouched = null;
                boolean isHeadTouched;
                boolean isNothingTouched = false;

                try {
                    isHandTouched =
        myRobotApp.mBinder.dmel_hri_get_hand_touch_info(getComponentName());
                    isHeadTouched =
        myRobotApp.mBinder.dmel_hri_get_head_touch_info(getComponentName());

                    // Simultaneous touch not considered.
                    if (isHandTouched[0]) {
                        myRobotApp.mBinder.dmel_tts_speak(getComponentName(),
                                                         "Left hand touched");
                    } else if (isHandTouched[1]) {
                        myRobotApp.mBinder.dmel_tts_speak(getComponentName(),
                                                         "Right hand touched");
                    } else if (isHeadTouched) {
                        myRobotApp.mBinder.dmel_tts_speak(getComponentName(),
                                                         "Head touched");
                    } else {
                        isNothingTouched = true;
                    }
                }
                catch (RemoteException e) {
                    e.printStackTrace();
                }

                if (isNothingTouched)
                    mHandler.sendEmptyMessageDelayed(0, 100);
                else
                    mHandler.sendEmptyMessageDelayed(0, 3000);
            }
        };

        mHandler.sendEmptyMessage(1000);
    }

    @Override
    protected void onPause() {
        super.onPause();

        if (mHandler != null) {
            if (mHandler.hasMessages(0))
                mHandler.removeMessages(0);

            mHandler = null;
        }
    }

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

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

[Listing 6–16] TestTouchSensorActivity.java

TestTouchSensorActivity starts when touch sensor icon is touched from MainActivity. onCreate() which is called at this time assings Handler oject and defines handleMessage(). In line 56, message is sent to the Handler,thereby executing defined handleMessage().
Functions dmel_hri_get_head_touch_info() and dmel_hri_get_hand_touch_info() in lines 31~32 of handleMessage() returns touch status of head and both palm sensors. Original format of each function is as follows.

dmel_hri_get_head_touch_info

```
boolean dmel_hri_get_head_touch_info(ComponentName cn)
                                  throws android.os.RemoteException
```

Retun head touch status.
Parameters:
cn – Component name
Returns:
Head touched true, not touched false return
Throws:
android.os.RemoteException

dmel_hri_get_hand_touch_info

```
boolean[] dmel_hri_get_hand_touch_info(ComponentName cn)
                                  throws android.os.RemoteException
```

Return hand touch (both hands) detection .
Parameters:
cn – Component name
Returns:
Touched true, not touched false return. boolean[2] (boolean[0] : left hand touch, boolean[1] : right hand touch)
Throws:
android.os.RemoteException

In lines35~43, isHandTouched[] and isHeadTouched with returned values from the fuctions are checked and touch status expressed by TTS. These codes do not process simultaneous touches. When sensors are touched simutaneously, output to the TTS follows following sequence. left hand, right hand, and head. If robot does not detect any touches,isNothingTouched is set to true by line 42. This variable is used in the if statement in lines 49~52. If isNothingTouched is true true, message is sent to the Handler 100ms later to read the sensor value again for real time sensor touch detection.

If isNothingTouched is false, it means one of the sensors detected a touch. In this case, depending on the touched sensor, appropriate output is sent to TTS. 3000ms time is line 52 gurantees time for TTS to finish output before sending message to the Handler again.

Have fun learning the fundamentals of robot programming
with Dongbu Robot Hovis Genie and Apps.

Dongbu Robot

Dongbu Robot Co.,Ltd www.dongburobot.com

HQ : 11th Fl, Bucheon Techno Prk Bldg, 401, Yakdae-dong,Wonmi-gu, Bucheon-si, Gyeonggi-do, S Korea
TEL : 032-329-5551 (ext 311), FAX : 032-329-5569, E-MAIL : robotsales@dongbu.com
Factory : 6 Gil 27, 4 San dan, Jiksan-eup, Sebouk-gu, Cheonan-ski, Chungcheonnam-do, S Korea
TEL : 041-590-1700, FAX : 041-590-1701

93550
9 788998 873073
ISBN 978-89-98873-07-3