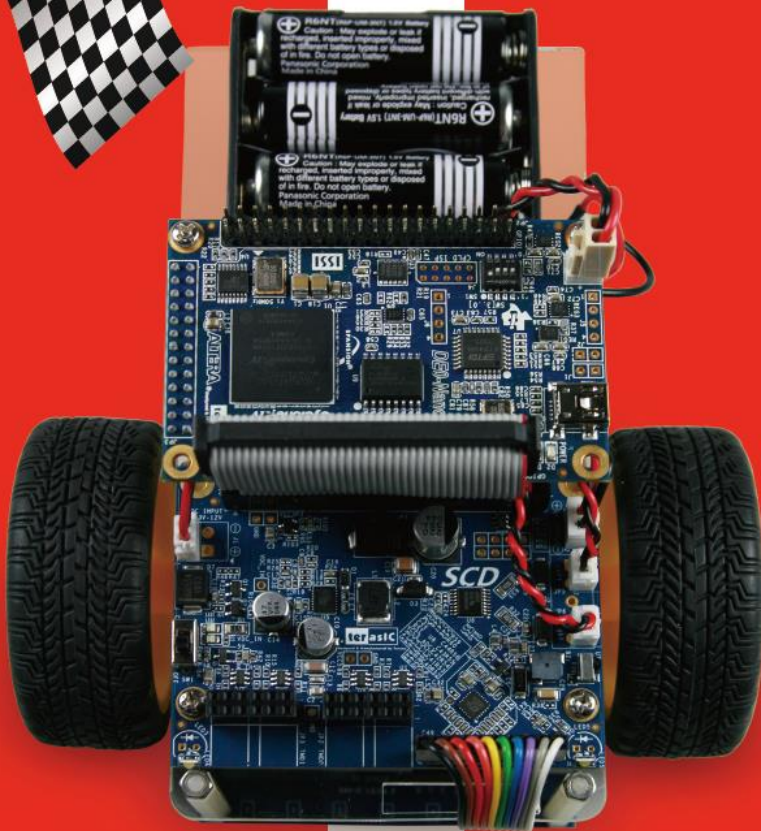
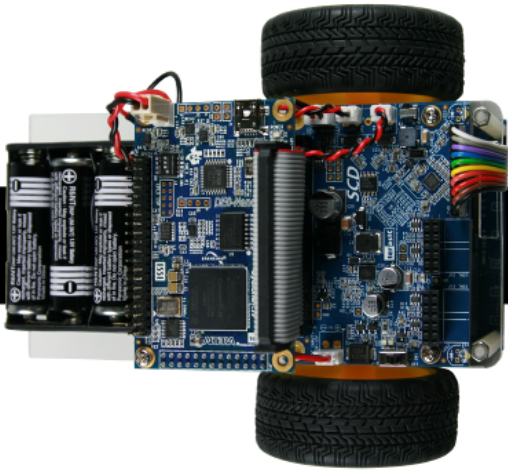




## Bluetooth Remote Control

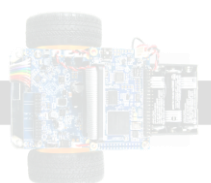


**terasic**  
www.terasic.com



## Abstraction

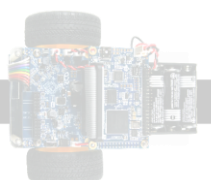
This application note provides details on how to achieve human-computer interaction (HCI) with Android device and Terasic A-Cute Car via Bluetooth using a module, Terasic BTS-TMD. The following instructions will help users easily learn more about UI design on device and the co-design of software and hardware on Terasic A-Cute Car. This enables users to develop customized system. Users can get started by scanning the QR code from any Android device and download Terasic A-Cute Car application.



# Content

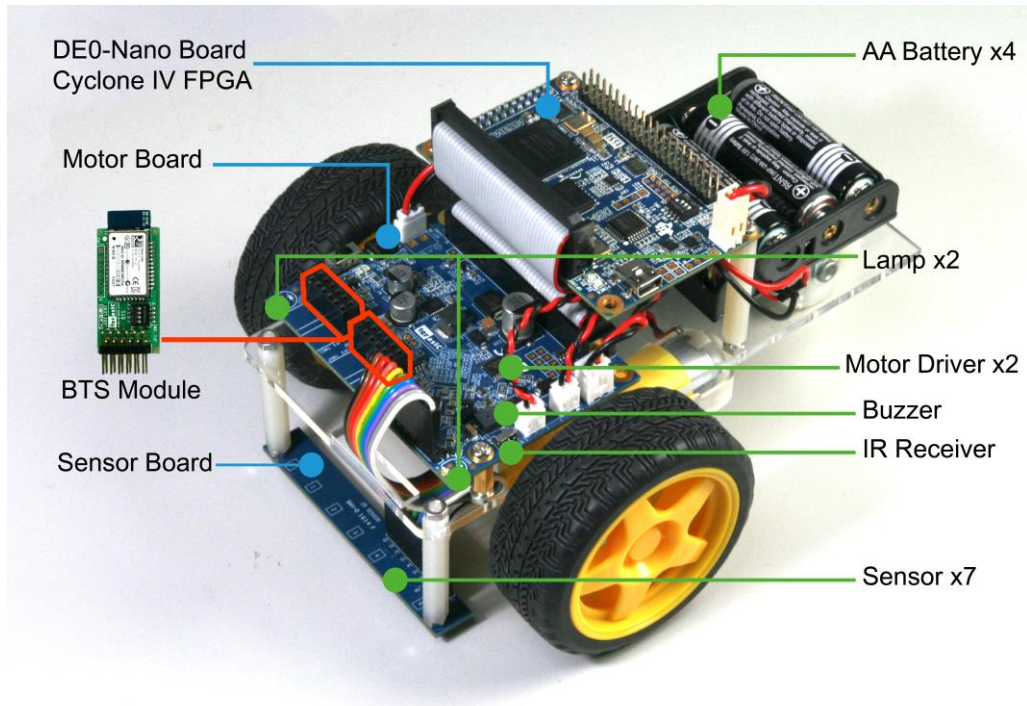
---

<u>Reference Design Overview</u> ..	2
<u>UART</u> .....	5
<u>LTC2308 VOLTAGE METER</u> ....	6
<u>PWM Controller</u> .....	7
<u>ANDROID APPLICATION</u> .....	8
<u>Demo Setup</u> .....	9
<u>Rebuild Project</u> .....	11
<u>Import Android Project</u> .....	11
<u>Update Demo File</u> .....	13

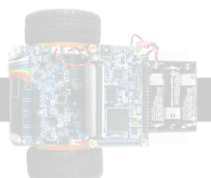


# Reference Design Overview

**Figure 1** shows the Terasic A-Cute Car. It consists of three components, DE0-Nano mainboard, SCD (Smart Car Daughter card), and sensor daughter card and optional BTS-TMD. The SCD includes a pair of lamps, buzzer, motor driver DRV8833, IR receiver, ADC chip LT2308, and TMD (Terasic Mini Digital) expansion header. The sensor daughter card includes seven photo interrupters used to track dark line(s) on a white background.

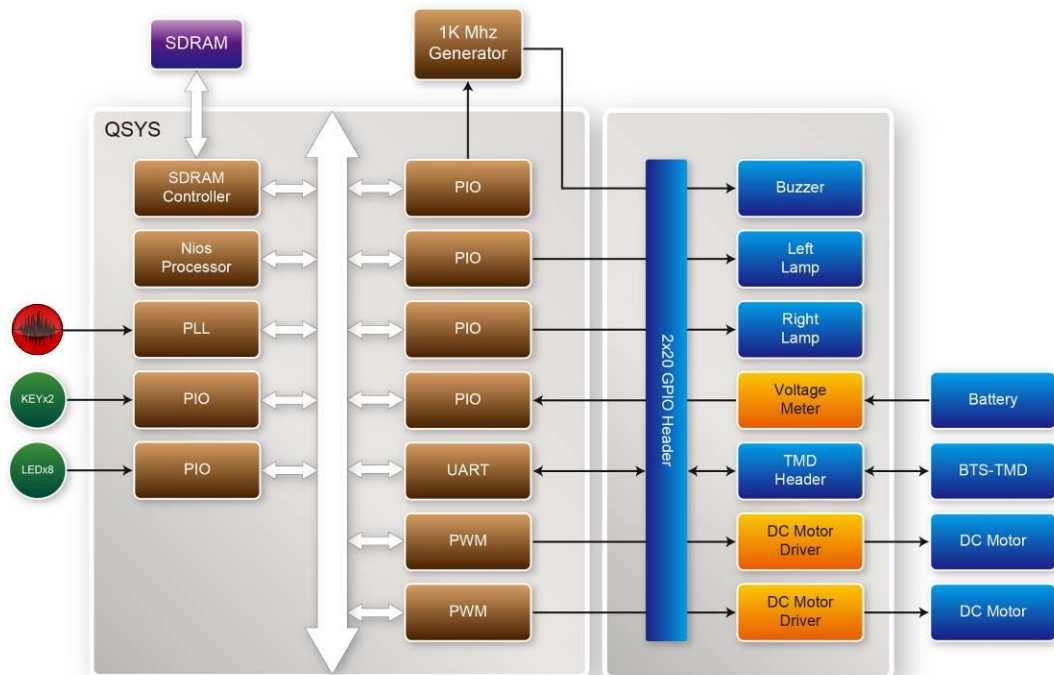


**Figure 1** Terasic A-Cute Car



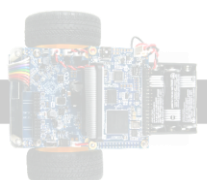


**Figure 2** shows the hardware block diagram of Terasic A-Cute car remotely controlled via Bluetooth. The Android device remote program is written in C++ and the operation is executed on SDRAM by Nios II processor. The pair of Lamps is controlled via PIO. The 1K waveform IP generates 1M frequency to drive the buzzer and the associated GPIO controls the beep sound. This demo also monitors the battery power left on A-Cute car in real time from Android device. Users can read the battery voltage via the LTC2308 ADC chip onboard and the data can be transferred to display on the Android device.

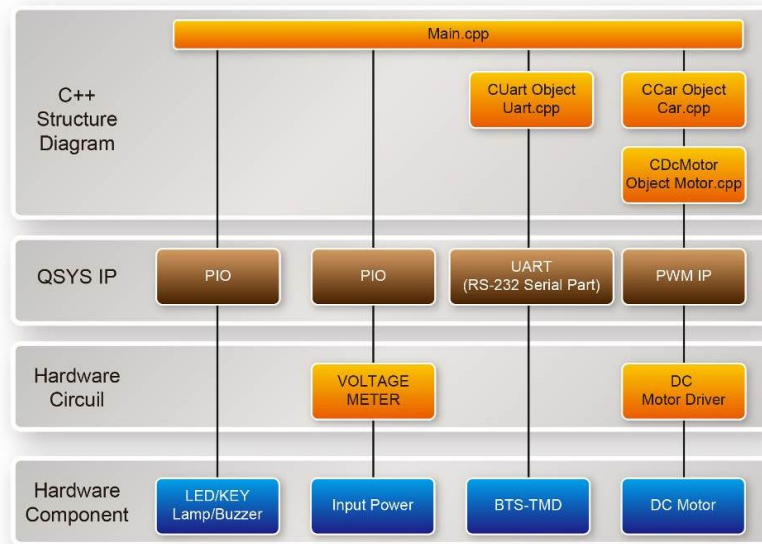


**Figure 2 System block diagram of remote control program on Terasic A-Cute car via Bluetooth**

The optional BTS-TMD is connected to the system via TMD0. The data transmission interface is UART. The movement of A-Cute car is controlled by the motor through PWM mechanism. Commands can be sent constantly from Android device to change the direction and speed of A-Cute car instantly. For more information about the remote control from Android device, please refer to the following chapters.

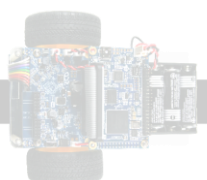


**Figure 3** shows the software block diagram of the demonstration. The section on the top is C++ Structure Diagram, which interfaces with the QSYS IP via Altera Avalon Memory-Mapped (AVMM) bus. Both IORD and IOWR are used to communicate with the QSYS IP.



**Figure 3 Software block diagram of remote control program on Terasic A-Cute car via Bluetooth**

The Main. cpp includes the PIO control for LED, KEY, Lamp, Buzzer, and battery power detection. The battery power detection uses hardware circuit to convert voltage measured from analog to 12-bit data. Users can read the data via PIO and send the UART package the BTS-TMD through UART object to Android device. The CCar object includes two **CDcMotor** sub-objects which control the two DC motors on the A-Cute car. The **CDcMotor** objects control the PWM IP to control the speed and rotation direction of DC motors.

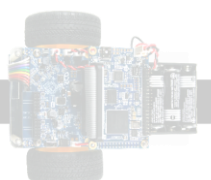


# UART

When the Bluetooth pairing is complete, the A-Cute car will stay in standby mode and wait for commands. **Table 1** shows the command set of all commands available. The Nios II processor will analyze the command after receive it through UART object.

Command	Description
ATSP	Speed setting
ATOF	Set the offset value of two motors for turning the direction
ATOFA	Adjust the sensibility of turning the direction
ATSP	Stop the car
ATLON	Turn on the lamps
ATLOFF	Turn off the lamps
ATSPKER	Honk
ATAB	Show battery level

**Table 1 Command set**



# LTC2308 Voltage Meter

LTC2308 is a low noise, 500Ksps, 8-channel, 12-bit ADC chip. Its last channel is used to monitor the input power voltage of A-Cute car. The ADC is configured as single-ended and the output value 0~4095 represents voltage 0~4.095V i.e. 1LSB represents 1mV. **Table 2** show the ADC index.

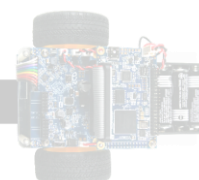
The last channel of the ADC chip can monitor larger range of input voltage by dividing input voltage connected to the ADC chip. The digitized value must be multiplied equally back to get the actual input voltage in mV.

In this demonstration, SPI bus is used to communicate between the FPGA and the LTC2308. The SPI clock of LTC2308 can be 40MHz at maximal. Considering the reliability of GPIO cable, 20MHz is used in this demonstration. The source code of LTC2308 IP is located under the “ M\_METER” folder in the system CD. The register file of the IP is defined below.

Register Index	Register Name	Description	Read/Write
0	CH0	Not use	R
1	CH1	Not use	R
2	CH2	Not use	R
3	CH3	Not use	R
4	CH4	Not use	R
5	CH5	Not use	R
6	CH6	Not use	R
7	CH7	12-bit battery voltage level detection	R

**Table 2 Read ADC index**

The member functions `"IORD(CAR_VOLTAGE_DATA_BASE,0) * 4"` can report the input power voltage individually.





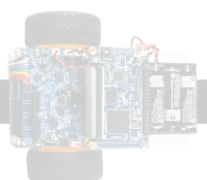
# PWM Controller

The PWM controller generates required duty cycle to control the speed of motor rotation. The source code of this IP is located under the folder “ip\TERASIC\_DC\_MOTOR\_PWM” in the system CD. The IP is encrypted as a QSYS compliant IP. The **Table 3** show register file of the IP is defined below.

Register Index	Register Name	Description	Read/Write
0	TOTAL_DUR	32-bit integer, which represents the tick number of one PWM cycle.	RW
1	HIGH_DUR	32-bit integer, which represents the tick number of high level in one PWM cycle	RW
2	CS	Control Register Bit0: Start bit 1 : Start 0: Stop Bit1: direction bit 1: forward 0:backward	RW

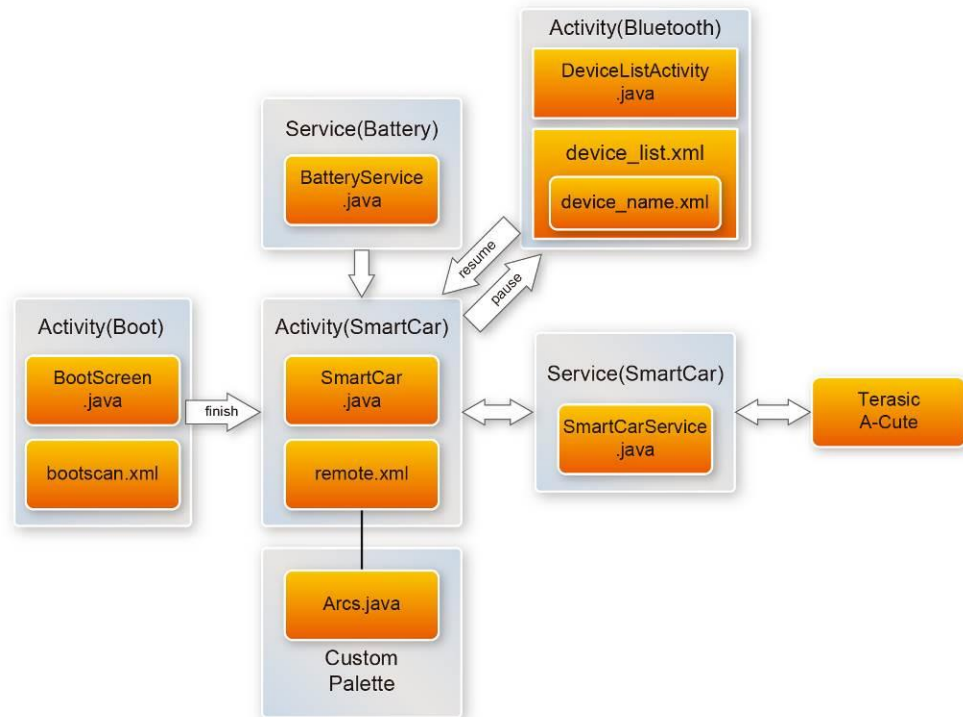
**Table 3 PWM control register table**

The CDCMotorC++ class defined in Motor.cpp/h communicates with the PWM hardware controller. The member function **SetSpeed** with an input parameter **fSpeed** controls the motor speed and direction. The value range of **fSpeed** value range -100.0~100.0. The positive and negative values presents forward rotary and backward rotary, respectively. The values 100 and -100 represents maximal speed for forward rotary and backward rotary, respectively. The **SetSpeed** function translates the input parameter **fSpeed** to required PMW parameters for the PWM controller. The translation formula depends on the input voltage level, which drives the DC motor. The member function **SetInputPower** is designed for users to input the current input voltage level. After setting the motor speed and calling the member function **Start**, the motor rotation can get started. Users can stop the motor rotation by calling the member function **Stop**. The CCar C++ class defined in Car.cpp/h control the movement of A-Cute car by controlling the two DC motors on the A-Cute car. The member function **SetSpeed** setups car movement speed and direction. The member function **Start** is designed to start the car moving and the member function **Stop** is designed to stop the car moving.



# Android Application

The development environment for the A-Cute car is Eclipse and the corresponding software block diagram of Android application is shown in **Figure 4**.

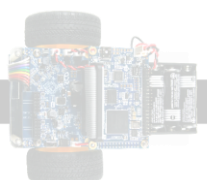


**Figure 4 Software block diagram of Android application**

An object created to interact with users is known as an Activity. It can be defined as stop, finish, kill, pause, resume, and restart etc. According to the illustration, there is a .java file and a corresponding UI .xml file in each Activity.

When the application is executed, it starts from Activity(Boot) and enters the boot screen for about 4 seconds before the object is switched from there to Activity(SmartCar).

If the A-Cute car is connected to a device such as Android device via Bluetooth, the Activity(SmartCar) will be paused and the Activity(Bluetooth) will be executed. After the connection is established, it will switch back to Activity(SmartCar). The Service(SmartCar) is running at the background of Android device and cannot be found on the UI. It is responsible for the transmission of all commands between the A-Cute car and Android device. The



Service(Battery) sends command to the A-Cute car for battery power detection in every 3 seconds via Bluetooth.

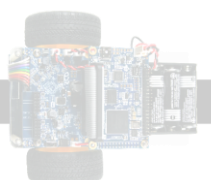
### Demo Setup

The steps to setup the demonstration are:

1. Insert a BTS-TMD module to the TMD header of SDC card, as shown in **Figure 5**.
2. Set the power switch of SDC card to OFF position Install four AA batteries to the battery holder
3. Set the power switch of SDC card to ON position
4. The demonstration will be executed upon power up. If the default code is erased, please execute test.bat from the folder below to configure the FPGA on DE0-Nano:

A-Cute Car CD/DE0\_NANO\_Remote\_Car/demo\_batch

- Prepare for remote function:
  - ✧ Scan the QR code shown in **Figure 6** from Android device to install the SmartCar application.
  - ✧ Launch the SmartCar application installed
- Remote control from Android device via Bluetooth, as shown in **Figure 7**.
  - ✧ Click “connect a Device” on the right-top corner. Search for Bluetooth device named RNBT-xxx and connect to it. The status of Bluetooth connection is shown on the top-left corner.
  - ✧ Click the yellow power button located on the right-top corner to ignite the car. Click the button again will power off the smart car.
  - ✧ The car movement can be controlled in four directions with the G-sensor on Android device.
  - ✧ Click the LAMP button on the left to turn on/off the lamps.
  - ✧ Click the center area of the wheel to make the beep sound.
  - ✧ There is a scrollbar under the wheel. It is used to adjust the sensitivity of car turning.
  - ✧ The orange curve bar on the left-bottom corner shows the estimated car speed.



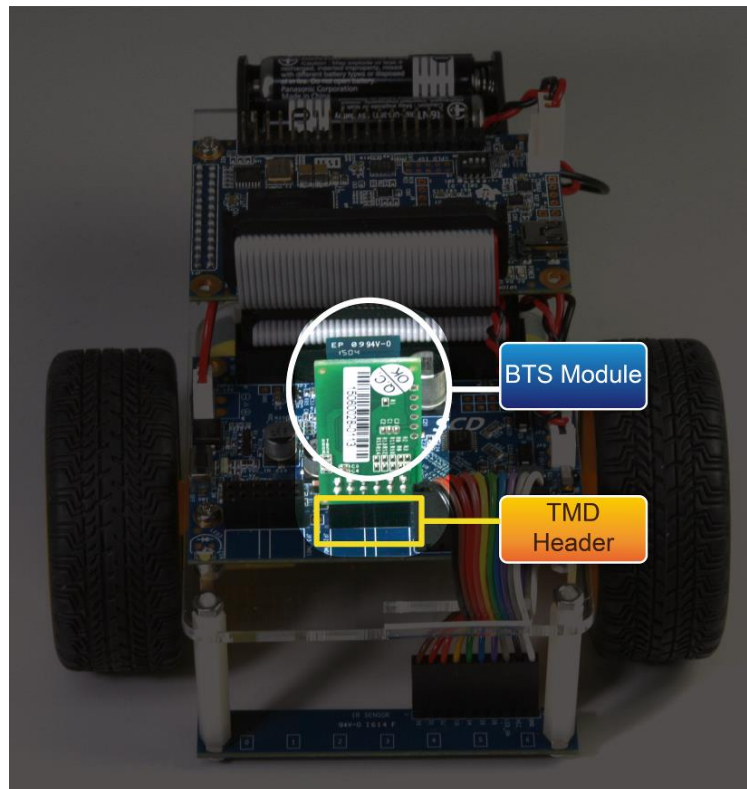


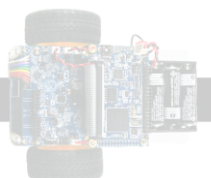
Figure 5 Insert BTS module



Figure 6 QR-code for A-Cute car Android application



Figure 7 GUI of Smart Car application on Android device



# Rebuild Project

The project is built in Quartus II v15.1. The source code is located under the folder:

A-Cut Car System CD/DE0\_NANO\_Remote\_Car

Users can open the Quartus project file DE0\_NANO\_Remote\_CAR.qpf and click the menu item “Processing→Start Compilation” to initiate the rebuild process. When the compilation is complete, an output file named DE0\_NANO\_Remote\_CAR.sof will be generated under the output\_files folder.

The Nios II project is created by Nios II v15.1 Software Build Tools for Eclipse. The source code is located under the folder:

A-Cut Car System CD/DE0\_NANO\_Remote\_Car/software

Launch Nios II v15.1 Software Build Tools for Eclipse and set the above folder as workspace. In the Project Explore Window, right click “RemoteCar\_bsp[nios\_system]” and select “NIOS II Generate BSP” to build the BSP. Right click “RemoteCar” and select “Build Project” to generate binary file. When the rebuild is complete, an output file named RemoteCar.elf will be generated under the folder:

A-Cut Car CD/DE0\_NANO\_Remote\_Car/software/RemoteCar

# Import Android Project

Both Android SDK and Eclipse ADT Plugin must be installed to complete the installation for this project prior to the development of Android application. For more settings about the API 17 used in this demo, please refer to

Figure 8

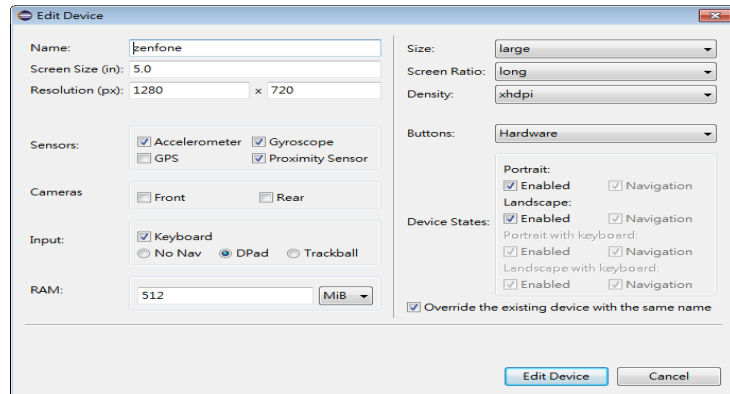
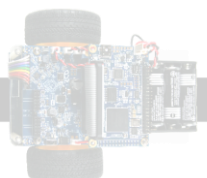


Figure 8 Edit Device

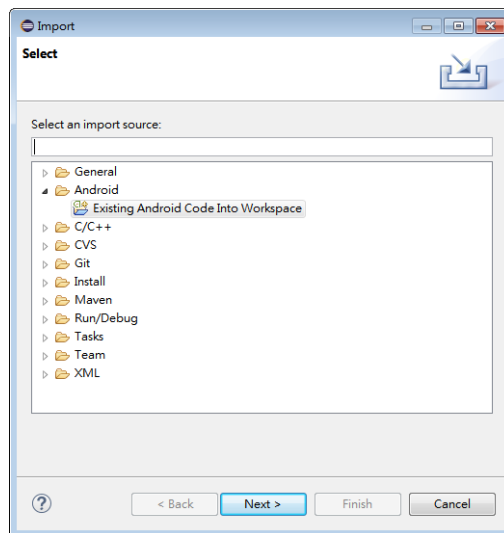




There are many Android SDK versions to be chosen from and the version 4.2.2 (API 17) must be installed. For more details about the installation, please refer to:

<http://developer.android.com/sdk/installing/installing-adt.html>

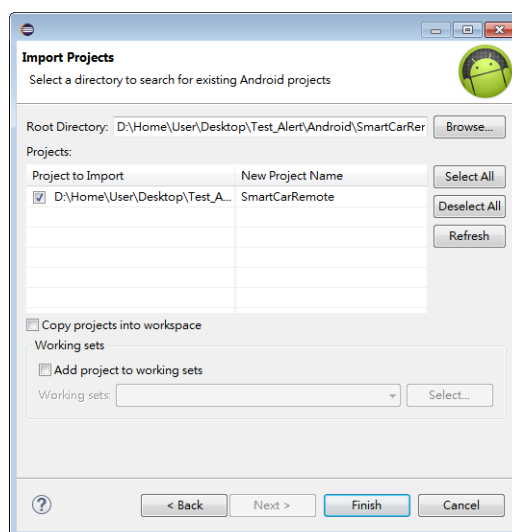
The project folder needs to be imported prior to the start of building the project. Choose File→Import→Android→Existing Android Code Into Workspace from the menu, as shown in **Figure 9**.



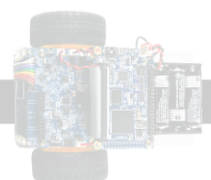
**Figure 9 Import existing Android code into workspace**

Press Next and browse to the project folder under the root directory. Click the Finish button, as shown in

**Figure 10**



**Figure 10 Import project to finish**



# Update Demo File

Before the demo file can be downloaded to Android device through Eclipse tool, the Debug mode must be enabled to allow the installation from an unknown source. The corresponding driver also needs to be installed on the host PC. Right-click on the project folder and select Run AS -> Android Application to download the demo file to Android device, as shown in

Figure 11

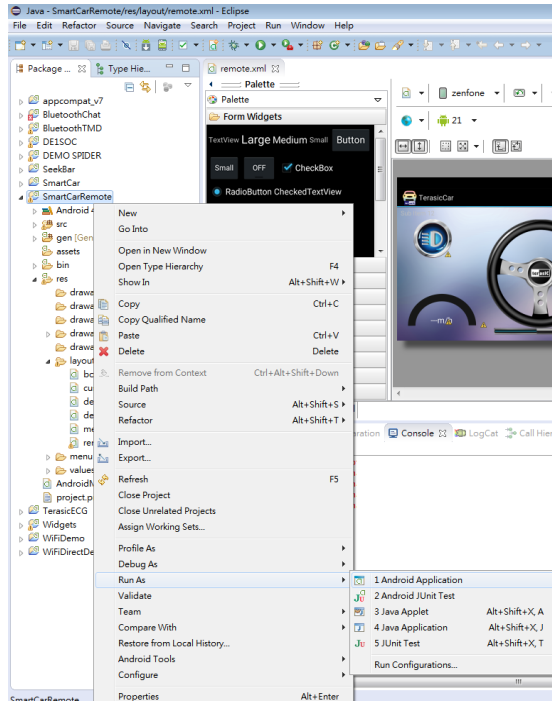


Figure 11 Update demo file

