Android

For

TM1

User Manual

Document Reference: TM1 Android Guide

Document Issue: 1.1

Associated SDK release: 1.0

Author: D Robinson

Contents

1. Introduction

Android was originally designed as a mobile operating system for phones and tablets, however many of Androids features are desirable in the context of embedded systems. These include:

- Standard SDK API's for most hardware interfaces
- Free feature rich development tools
- Royalty free software distribution without requirement for disclosing source code
- Large developer base
- Rich native multimedia capabilities
- Standard user interface
- Quick time to market
- Native Java and C++ language support. Other languages supported through third party tools.

Where Android falls short in the embedded space, is lack of support for embedded hardware interfaces like serial ports, i2c buses, and GPIO's. As a rule of thumb, unless hardware is defined in the Compatibility Definition Document (CDD) there is unlikely to be a native Android API available.

Blue Chip Technology have overcome this limitation by making a custom API available to customers, which allows access to nonstandard hardware interfaces.

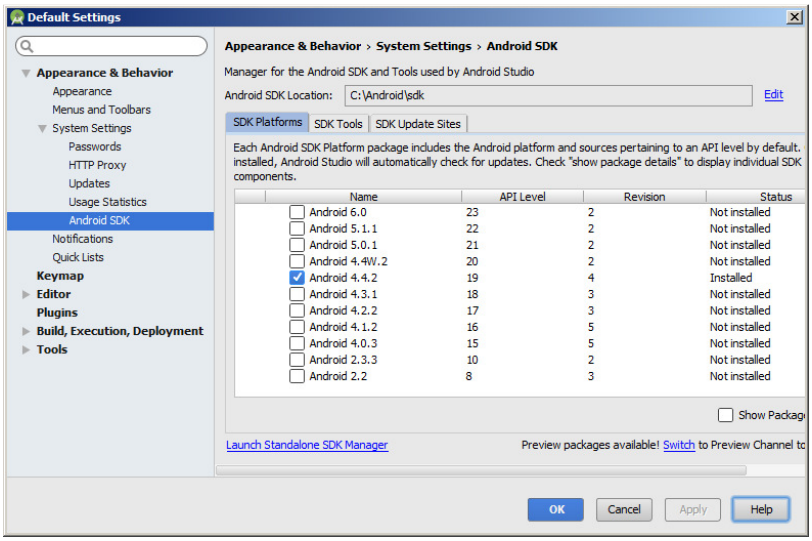
The content of this document provides information required to start building Android applications for the BCT TM1 / HB5 platform. It covers:

- Development environment requirements
- How to Enable debugging on the TM1/HB5 platform
- How to setup a simple hello world application in Android Studio
- How to import the BCTAPI hardware library into Android Studio
- Definitions of the BCTAPI hardware library class structure

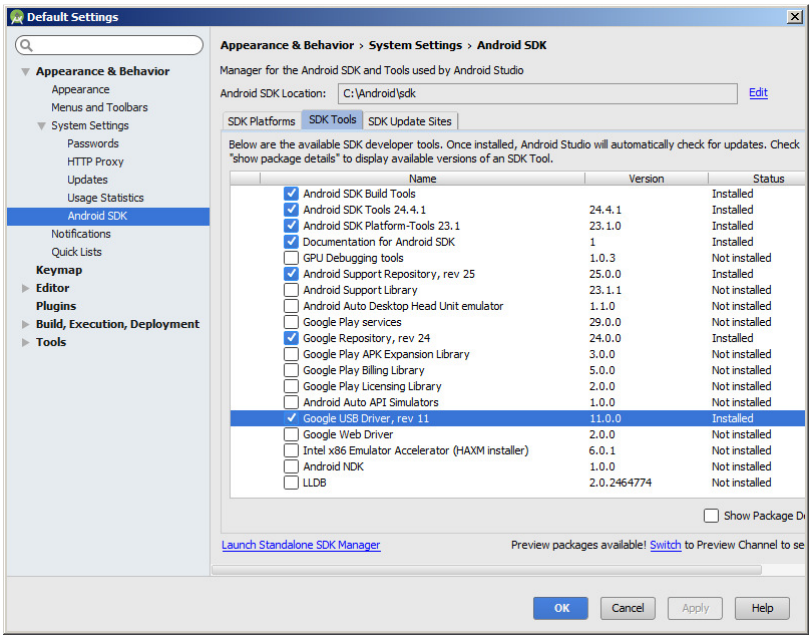
2. Environment

Android applications for TM1/ HB5 can be implemented in either Android Studio or with the older Eclipse ADT plugin. The examples in this document focus on the Android Studio environment.

At the time of writing, TM1 / HB5 supports Android 4.4.3 (Kit Kat) which corresponds to Android API version 19. It is important that API 19 is installed in the Android SDK manager.



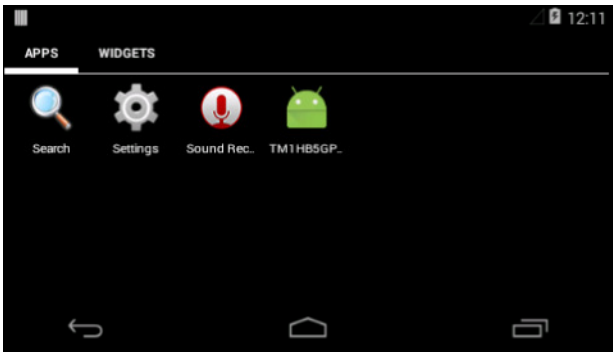
Debugging of Android applications is typically performed over the Android ADB USB interface. To enable this feature within Android Studio the USB debug feature must be installed in the Android SDK manager.



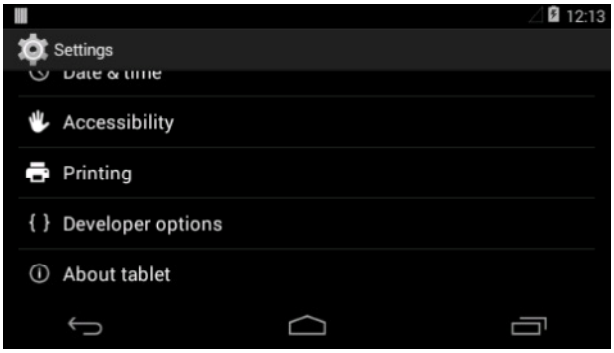
3. Enable USB ADB Debugging on TM1/HB5

By default the USB ADB Debugging interface is turned off. To enable the debug interface follow the below steps.

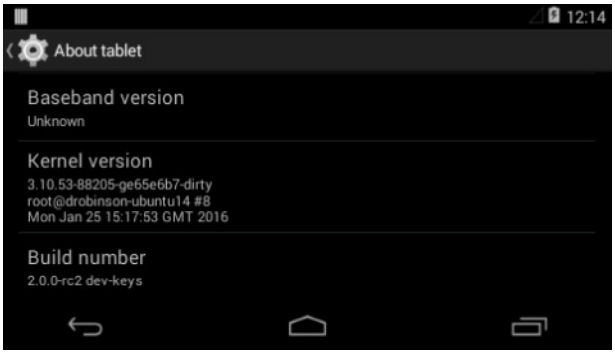
1. Navigate the Android settings control panel.



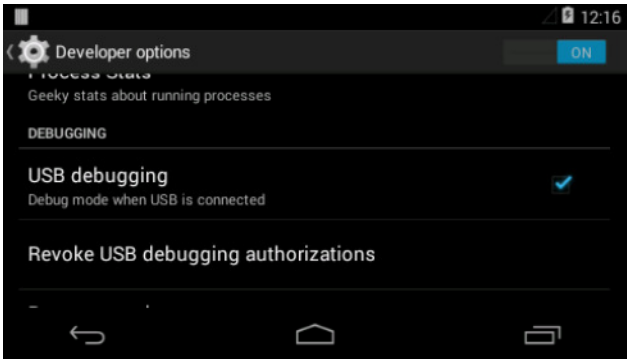
2. Scroll to the bottom of the list and click, “About tablet”



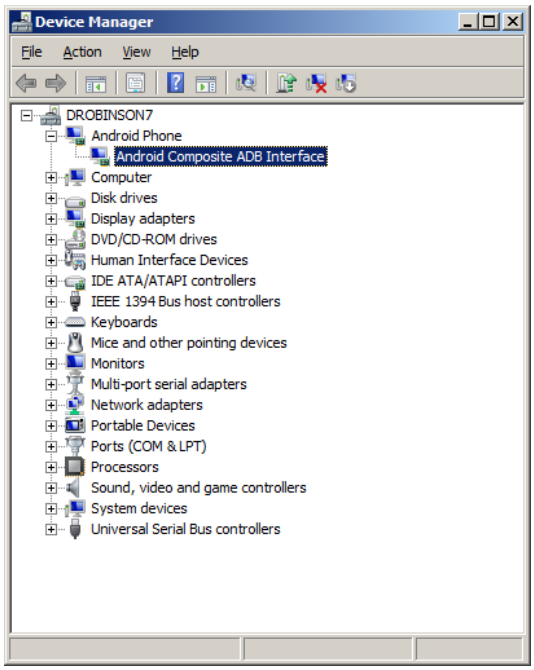
3. Scroll to the bottom of the list and click, “Build Number” repeatedly until a message is displayed saying, “you are now a developer”.



4. Press the back button, and click on, “Developer Options”.
5. Scroll down to the option, “USB Debugging”, and click to enable the feature. You may be prompted to confirm that debugging is allowed.



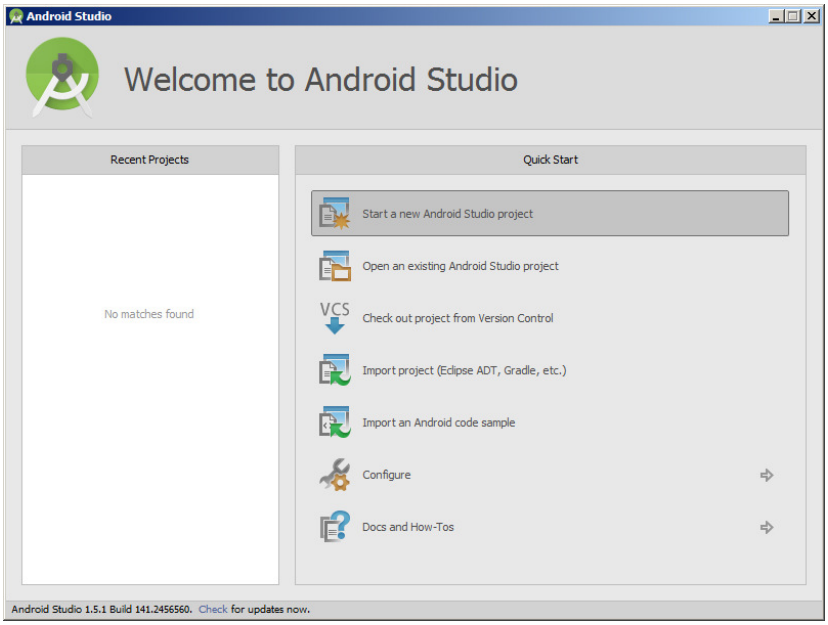
6. Connect a USB cable between the development PC and TM1/HB5.
7. Windows should detect a new ADB USB device and search for drivers. If a driver cannot be found automatically it may be necessary to point Windows device manager at the following location. <Android SDK root>\extras\google\usb_driver.
8. Upon successfully loading the ADB driver, Windows device manager should display an Android ADB device.



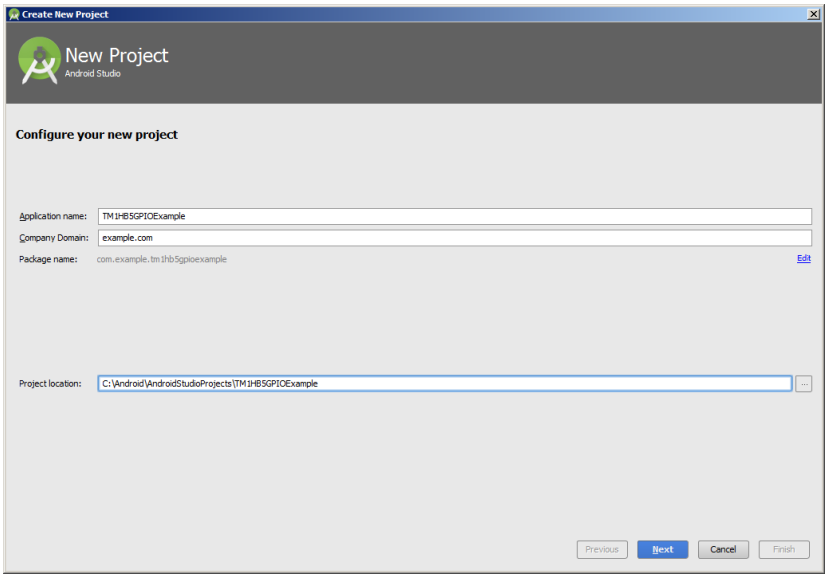
4. Simple GPIO light switch example

The following steps describe how to setup and deploy a basic Android App to TM1/HB5. The app has a simple toggle button that controls a GPIO output. The walkthrough presumes that Android Studio is installed, and that the SDK manager is setup as per the previous section.

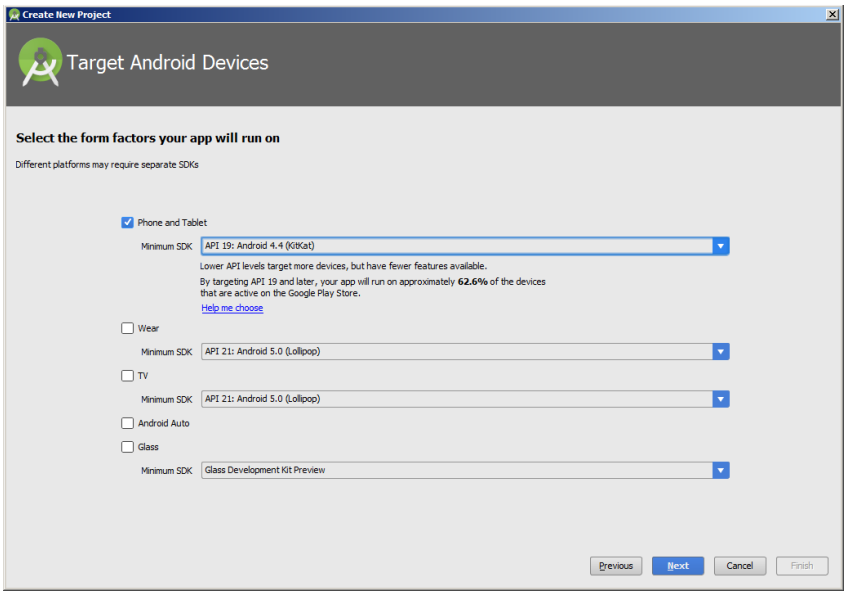
1. Start Android Studio
2. Click “Start a new Android Studio project“, in the “Quick Start” menu.



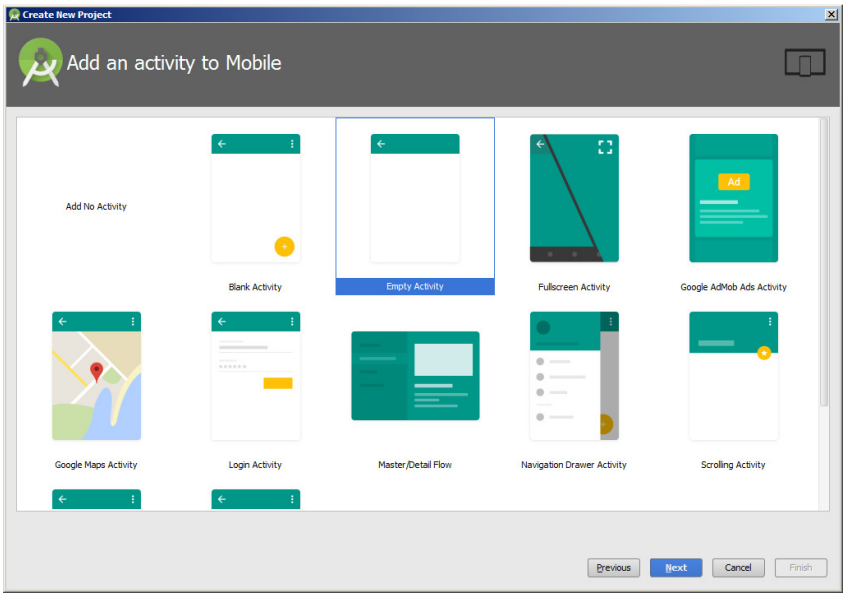
3. Give the project a name and namespace.



4. Tell the wizard that the app is targeting API 19 for a Phone / Tablet device.

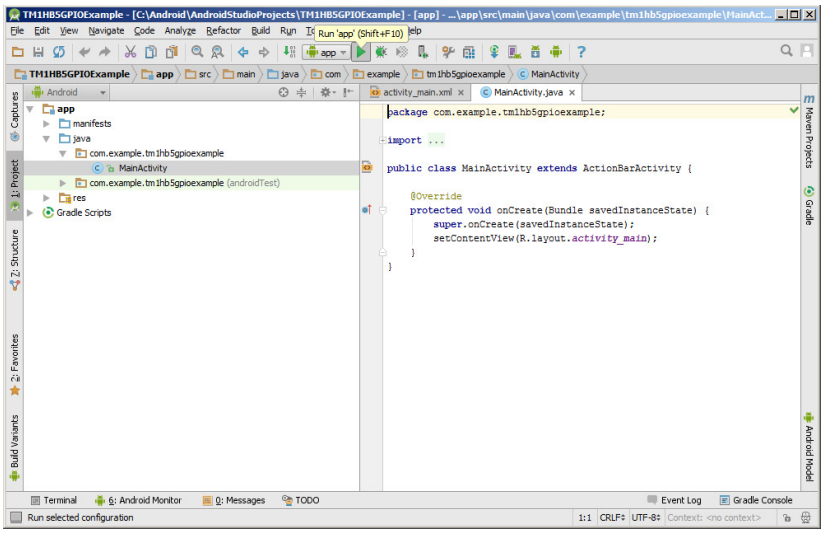


5. Select Empty Activity

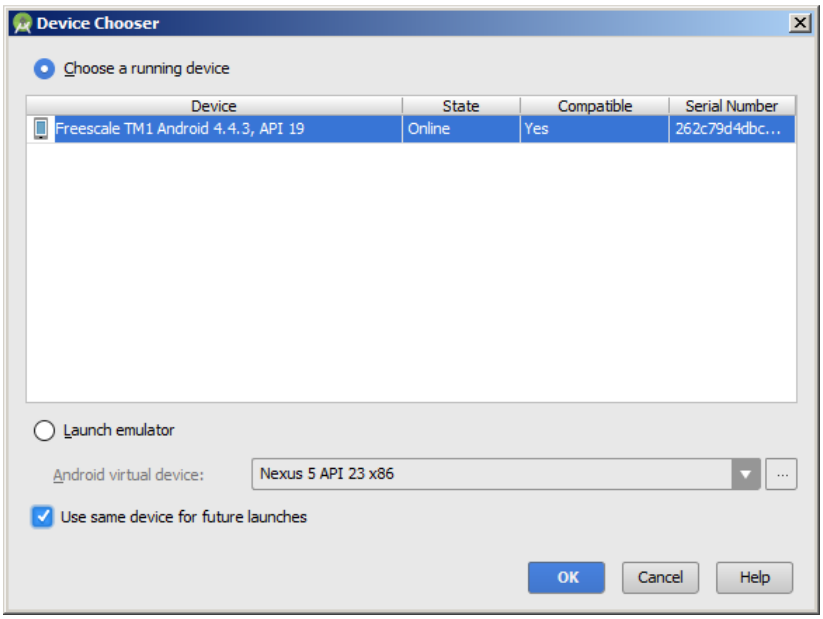


6. Use the default activity name and click finish. Android studio will now initialise the development environment.
7. At this stage it is advisable to build and deploy the app in its default state. Ensure that the TM1/HB5 device has been setup for debug over USB and that the appropriate driver has been installed on the development PC. See previous section for details.

8. Click the “Run App” button to deploy the app

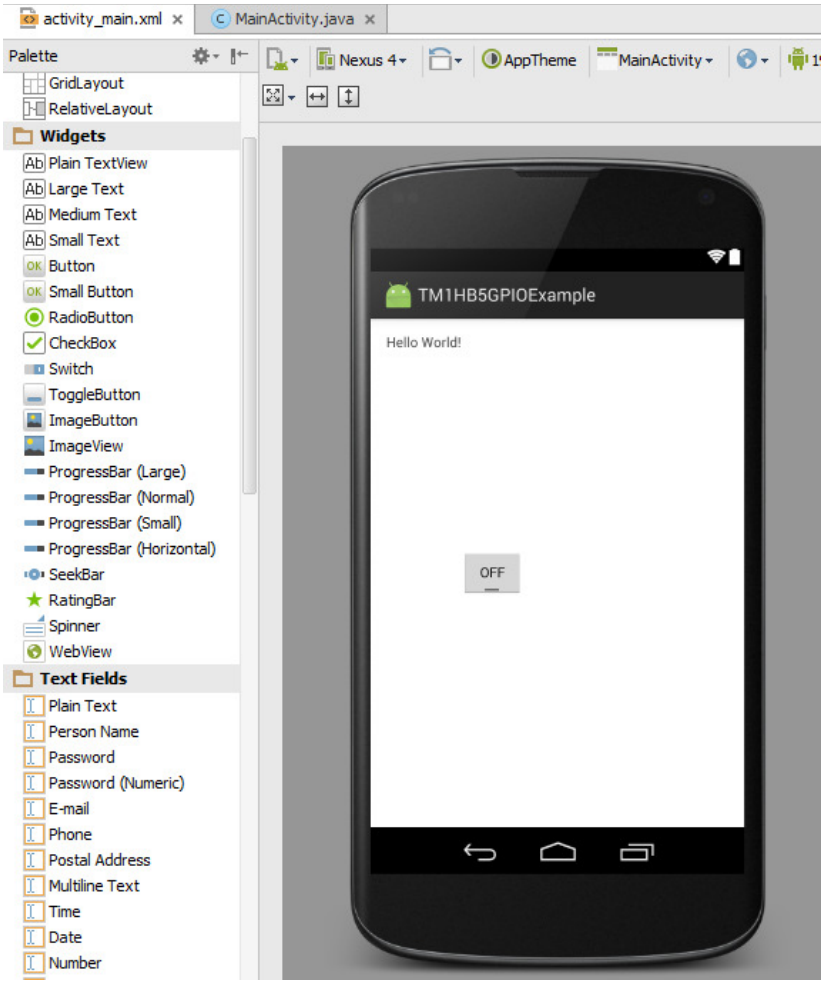


9. Select the TM1/HB5 device in the “Device Chooser” dialogue box and press ok. If the device is displayed as unauthorised, check the TM1 /HB5 display for an authorisation request.



10. The Hello world application should automatically deploy to the device and execute.

11. Add a toggle button control to the activity_main.xml gui designer.



12. Import the BCTAPI.jar library into the Android project.
    a. Obtain the BCTAPI.jar library from Blue Chip Technology
    b. Copy BCTAPI.jar to <AndroidStudioProjects>\<projectname>\app\libs

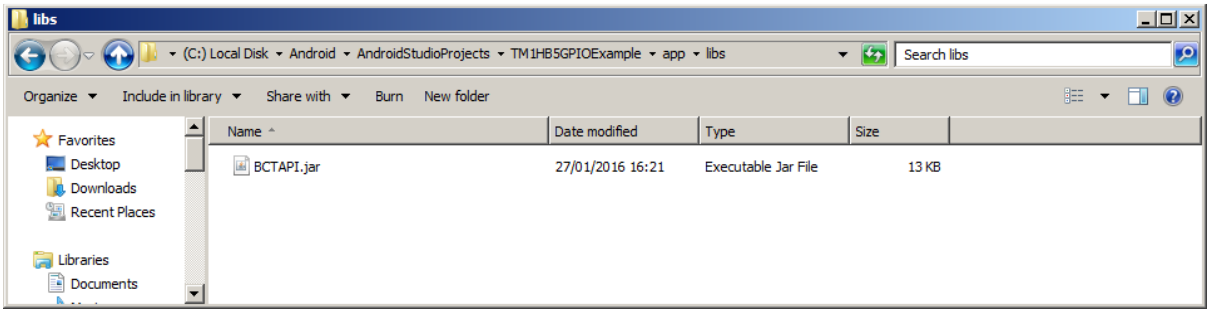
13. Modify the MainActivity.java source code so that it contains the following.

```
package com.example.tm1hb5gpioexample;

import bct.hwapi.*;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.CompoundButton;
import android.widget.ToggleButton;

public class MainActivity extends ActionBarActivity {

    private static final String TAG = "tm1hb5gpioexample";
    GPIO gpio = null;
    ToggleButton toggle = null;

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

        try {
            gpio = new GPIO(GPIODefinitions.GPIO_USER_LED); //Create GPIO object for user gpio on P12
            gpio.InitialiseGPIO(GPIO.GPIODirection.OUTPUT); //Set GPIO as an output
        }
        catch(Exception ex)
        {
            Log.e(TAG, "Failed to initialise GPIO: " + ex.getMessage());
        }

        toggle = (ToggleButton) findViewById(R.id.toggleButton);
        toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
            {
                try {
                    if (isChecked) {
                        gpio.SetOutput(1);
                    } else {
                        gpio.SetOutput(0);
                    }
                }
                catch(Exception ex)
                {
                    Log.e(TAG, "Failed to set GPIO: " + ex.getMessage());
                }
            }
        });

    }
}
```

14. Run the app in the same way as step 8. The LED on p12 (Ethernet Connector) will indicate the state of the toggle button.

5. BCTAPI

The BCT API for Android is distributed in the form of a java JAR file with filename BCTAPI.jar. The library can be imported into an Android Studio project by copying the BCTAPI.jar into the “app\libs” directory within the project structure. E.g.



The BCT API contains class definitions which allow Android apps to control serial ports, i2c ports, GPIO pins, and other hardware bespoke to the TM1/HB5 platform.

BCTAPI Namespace

All BCT API class definitions are implemented in the namespace bct.hwapi. By including the following import definition in an Android Studio source file, all defined namespaces will be available to the developer.

```
import bct.hwapi.*;
```

SerialPort Class

Class Namespace:

```
bct.hwapi.SerialPort
```

Constructor:

```
SerialPort(File serialportfile, int baudrate, int wordlendth, int stopbits) throws
SecurityException, IOException
```

Description:

Opens a serial port with the specified parameters.

Parameters:

serialportfile – The filename of the serial port to open. TM1 / HB5 supports two serial ports by default these are:

/dev/ttymxc1 - RS232 levels on P4

/dev/ttymxc2 - RS232 or RS422/485 levels on P4.

baudrate – The baud rate that the serial port will operate at.

wordlength – The length of each word transmitted or received. Valid values are 5, 6, 7, and 8.

stopbits – The number of stop bits included with each word. Valid values are 1 and 2.

SerialPort.Close()

Definition:

```
void Close();
```

Description:

Closes an open serial port.

SerialPort.WriteString

Definition:

```
void WriteString(String text) throws IOException
```

Description:

Write a string of characters in UTF-8 format to the serial port synchronously.

Parameters:

text – String of characters to transmit.

SerialPort.ReadString

Definition:

```
String ReadString() throws IOException
```

Description:

Read a string of characters in UTF-8 format from the serial port synchronously.

SerialPort.getInputStream

Definition:

```
InputStream getInputStream()
```

Description:

Function to retrieve the InputStream of an open serial port. This is useful if byte level access to a serial port is required. The return value will be null is the serial port failed to open.

SerialPort.getOutputStream

Definition:

```
OutputStream getOutputStream()
```

Description:

Function to retrieve the OutputStream of an open serial port. This is useful if byte level access to a serial port is required. The return value will be null is the serial port failed to open.

I2C Class

Class Namespace:

```
bct.hwapi.I2C
```

Constructor:

```
I2C(File device) throws SecurityException, IOException
```

Description:

Opens an I2C port with the specified parameters.

Parameters:

device – The filename of the I2C port to open. TM1 / HB5 supports two I2C ports by default these are:

/dev/i2c-0 – 1.8V bus on TM1

/dev/i2c-1 - 3.06V bus on HB5.

I2C.Close()

Definition:

```
void Close();
```

Description:

Closes an open I2C port.

I2C.ReadByte

Definition:

```
byte ReadByte(byte slaveaddress, byte offset) throws IOException
```

Description:

Read a byte of data from an I2C slave device.

Parameters:

slaveaddress – Address of I2C slave

offset – address offset to read data from.

I2C.WriteByte

Definition:

```
void WriteByte(byte slaveaddress, byte offset, byte data) throws IOException
```

Description:

Write a byte of data from to an I2C slave device.

Parameters:

slaveaddress – Address of I2C slave

offset – address offset to write data to.

Data – data to be written to I2C slave

I2C.BufferedRead

Definition:

```
byte[] BufferedRead(byte slaveaddress, byte offset, byte count) throws IOException
```

Description:

Read (n) bytes of data from an I2C device into a byte array.

Parameters:

slaveaddress – Address of I2C slave

offset – address offset to read data from.

count – number of bytes to read

I2C.BufferedWrite

Definition:

```
void BufferedWrite(byte slaveaddress, byte offset, byte[] buffer) throws
IOException
```

Description:

Write (n) bytes of data to an I2C slave.

Parameters:

slaveaddress – Address of I2C slave

offset – address offset to read data from.

buffer – array of bytes to be written to an I2C slave.

GPIO Class

Class Namespace:

```
bct.hwapi.GPIO
```

Constructor:

```
GPIO(GPIODefinitions gpio) throws Exception
```

Description:

Opens a GPIO pin.

Parameters:

gpio – A GPIO pin defined in GPIODefinitions enum. Values include:

GPIO_0 - GPIO_11 – These correspond to GPIO pins on the HB5 P5 GPIO connector.

GPIO_USER_LED – This GPIO controls an amber LED on the HB5 P12 connector (Magjack).

GPIO_422_485_TXEN – Control of the transmit control signal for the RS422 / 485 transeiver.

GPIO.InitialiseGPIO

Definition:

```
void InitialiseGPIO(GPIODirection direction) throws SecurityException, IOException
```

Description:

Initialise a GPIO pin, and set the pins initial direction

Parameters:

direction – Either GPIODirection.INPUT or GPIODirection.OUTPUT. GPIO inputs on HB5 are configured without any pull-up or pull down resistors.

GPIO.ReadInput

Definition:

```
int ReadInput() throws IOException
```

Description:

Read the logical state of a GPIO input. This function will not return the state of a GPIO output pin.

GPIO.SetOutput

Definition:

```
public void SetOutput(int value) throws IOException
```

Description:

Set the output value of a GPIO pin.

Parameters:

value – 0 = low

1 = high

GPIO. SetDirection

Definition:

```
void SetDirection(GPIODirection direction) throws IOException
```

Description:

Controls whether a pin is setup as an input or output.

Parameters:

Direction – Either GPIODirection.OUTPUT or GPIODirection.INPUT

Audio Class

Class Namespace:

```
bct.hwapi.Audio
```

Constructor:

```
Audio() throws Exception
```

Description:

Creates in instance of the audio class

Audio.EnableClassD

Definition:

```
void EnableClassD() throws IOException
```

Description:

Enable the Class D speaker output on HB5

Audio.DisableClassD

Definition:

```
void DisableClassD() throws IOException
```

Description:

Disable the Class D speaker output on HB5

6. Sample Applications

TM1HB5GPIOExample

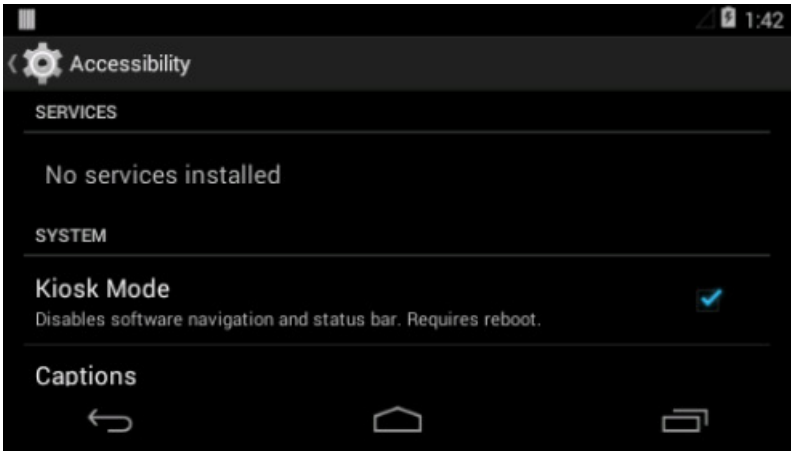
This sample demonstrates how to control the GPIO pins on HB5 using the GPIO API. The sample also demonstrates how an Android app can be made to operate in full screen mode, and even take the place of the default desktop to create a more embedded experience.

TM1HB5SerialportSample

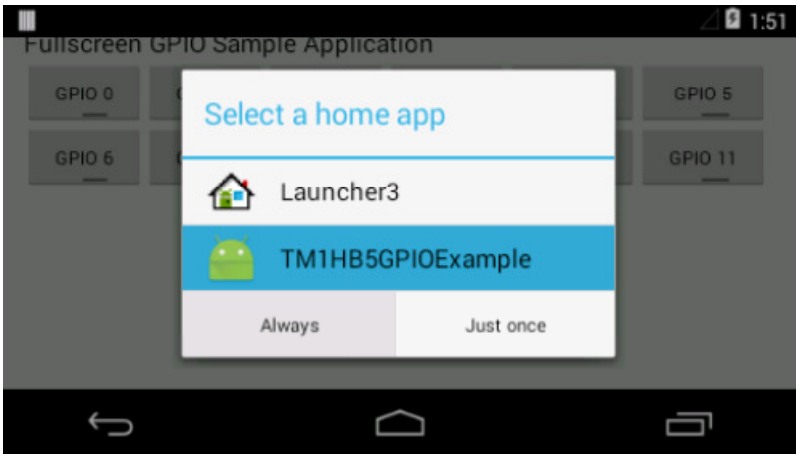
This sample demonstrates how to use /dev/ttymxc2 in RS232 and RS485 mode using the SerialPort API. The GPIO API is used for transmit enable control.

7. KIOSK Mode

Blue Chip Technology has made some OEM customisations to Android which allows a developer to lock down the operating system by hiding the system user interface. This can be achieved by navigating to the Settings -> Accessibility page, and checking the “Kiosk Mode” option. After selecting this option wait 5 seconds to allow time for the operating system to flush the setting to disk, and reboot the device. After selecting “Kiosk Mode” and rebooting the device, the software navigation buttons, and status bar will be disabled.



To complete the process of locking down the unit, a customer application must be installed which overrides the android “HOME” intent. See the TM1HB5GPIOExample for an example of how to do this.



8. Custom Boot Animation

The default Android boot animation can be overridden by copying a custom bootanimation.zip file into the bootanimation directory of the internal storage. There are various sources on the internet that describe the format of bootanimation.zip. E.g.

http://www.addictivetips.com/mobile/how-to-change-customize-create-android-boot-animation-guide/

A sample bootanimation.zip file downloaded from the internet is including in the TM1 Android SDK download.