Test report for EV3 Robot Kit

Issued for: University of Adelaide

Authors:
Xiang Guo (a1077337)
Thomas McAtee (a1608768)

December 19th 2014

Contents

1 Introduction

This report was created to test the functionality of the new Lego EV3 Robot kit acquired by the University of Adelaide for potential use as part of future Software Engineering Project course. The EV3 will be tested against the previously used Lego NXT Robot kits and judged on their feasibility to replace the older NXT Robot.

The following sections will provide a brief overview on some of the differences between the NXT and the EV3. For more details please refer to the rest of this document.

1.1 Brick Comparison

(a) EV3 Brick

(b) NXT Brick

1.2 New Features

TODO: Add images of new modules (e.g. WiFi, USB host port, medium motor, improved ultrasonic sensor, etc.)

A: USB Host Port - Can be used currently to connect WiFi dongle. Other hardware are possible such as keyboard, mouse, USB Memory stick, web cam etc but subject to availability of additional software.

B: Micro SD card slot - Can support up to 32 gb SD cards as long as it is Fat-32.

C: Motor Ports - 4 ports in total, 1 more than the NXT Brick.

D: LCD Screen - More than double the pixel resolution of the NXT brick.

E: Shifted button - Exit/Cancel button location shifted.

F: Additional buttons - 2 additional buttons compared to the NXT Brick, there are Up and Down respectively.

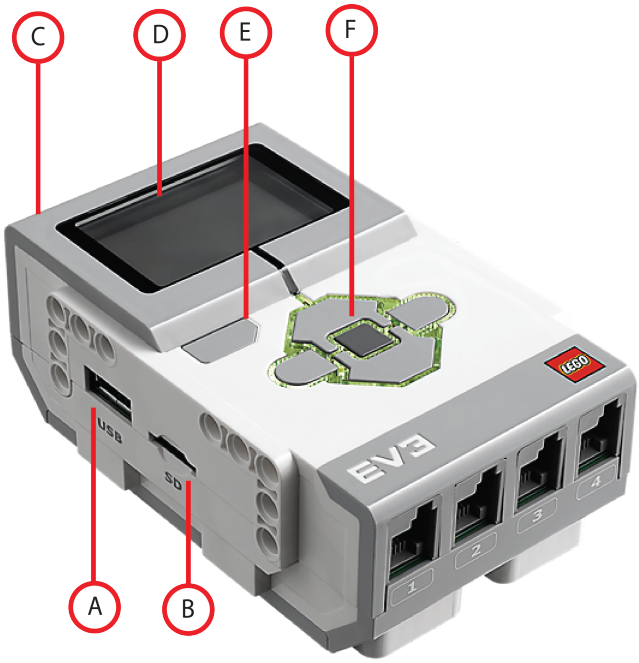

Figure 2: EV3 Brick with new features

1.3 Improvements

- EV3 Bump sensor is slightly shorter than the NXT Bump Sensor.
- EV3 Ultrasonic sensor is slightly longer than the NXT Ultrasonic Sensor, but is vastly improved.
- EV3 Colour sensor is much improved compared both the NXT colour and light sensor.
- EV3 Pivot wheel design is much improved.

2 Hardware

2.1 EV3 Brick

2.1.1 Display

- Resolution more than tripled. EV3:178x128 (Total pixels: 22784) vs NXT: 100x64 (Total pixels:6400).
- Improved exception feedback. NXT exception feedback came in the form of a code which needs to be checked against a table. EV3 screen displays a full non-wrapping stack trace which is navigable using the hardware buttons.

- Coordinates on screen has changed. 0,0 on NXT was the bottom left of screen. 0,0 is top left on EV3 display.

2.1.2 Ports

- Modified RJ-12 connector: The EV3 still uses the same type of proprietary modified RJ-12 connector that NXT used. These differ from normal RJ-12 connectors and the position clip is located off-centre. These connectors can be purchased online if needed.
- 5 sensors supplied with the kit but only 4 sensor ports available. There are after market hardware which will allow multiple sensors to be attached to one port. But this requires heavy amount of permanent modifications to the sensor cables which is not advised.
- USB port can be used to connect up to 4 EV3 bricks which allows for control of up to 16 motors and 16 sensors on one robot.
- USB port can be used with a WiFI dongle to provide WiFi connectivity(see section 4 for more details.)
- USB port can be used to connect a wide range of accessories such as keyboard, mouse, web cam etc. This is subject to availability of software drivers for the device.
- Micro SD port supports up to a 32gb SD card. Can be used to increase brick storage, but mainly as the boot drive for LeJOS. SD card needs to be formatted using FAT32.

2.1.3 Connections

TODO: is this an overview of the wifi/usb/bluetooth capability??

2.1.4 Battery

- Lithium ion (AA battery pack also supplied)
- 2050mAh compared to 2100mAh of the NXT.
- The EV3 seems to have better battery performance than the NXT battery that was tested. This was most likely due to the age of the NXT kits.

2.1.5 Buttons

- Backlit
- 6 hardware buttons

2.2 Sensors

TODO: include images for each sensor?

2.2.1 Light

Same device as the NXT Robot. New API on EV3 which returns a float instead of an int. Accurate to 2 decimal places. New API does not allow calibration (which isn't a new feature; the NXT API did not include methods to truly calibrate the sensor anyway, it simply created upper or lower bounds and normalised the returned data). If users wish to do this kind of soft calibration, they can use the source code for the NXT LeJOS API as a guide.

Has two modes, reflection and ambient. Reflection is the default mode and should be used for line following code. Black lines on normal print paper show up at around 40 percent reflection whereas white paper shows up as roughly 75%.
Ambient mode can be used to detect light sources. This might be an interesting feature if the robot can orientate itself based on a light source.

2.2.2 Color

The Color sensor has three modes: Reflection, Ambient and Color. Ambient and Reflection modes are the same as the Light Sensor equivalent, however the sensitivity of the sensor while measuring both is different.

Ambient mode is around half the sensitivity of the Light Sensor which means it should be able to differentiate between relatively bright light sources but might have trouble differentiating low level light sources.

Reflection mode is much more sensitive than the Light Sensor. Black lines on standard paper at 10mm range returned a reflection result of around 10 percent whereas white paper at the same distance returned 70 to 80 percent. This increased sensitivity should help in the project for Line Following and should be able to better handle smudges and imperfections on the test maps if the user implements a suitable tolerance in the code.

The color mode is meant to differentiate between up to 8 different colors, black, gray, red, green, yellow, blue, magenta, white. Testing was done using a laser printed standard 80gsm A3 paper, which contained a 256 color RGB chart. All tests were conducted at a range of approximately 5mm and at varying angles between 60 to 120 degrees. The sensor was able to pick up solid primary colors without issue at the angles tested. By primary we mean any colour where it's RGB hex value was represented by only pairs of F or

0. Eg. FFFFFF for white, FF0000 for red, FFFF00 for yellow etc.

Intermediate colors produced mixed results depending on the angle. Eg. FF8700 can be recognised as red at certain angles and as yellow at other angles. This is due to the reflection of the floodlight used by the sensor and the reflectivity of the ink on the paper. Therefore it is recommended in future projects to use only primary colors and non reflective paper to reduce the likelihood of erroneous readings. Other precautions that can be taken is to use colors that a true primary RGB e.g. Black 000000, white, FFFFFF, Red FF0000, Green 00FF00 or Blue 0000FF only and calibrate the sensor to only return one of those options.

2.2.3 Gyroscope

This new sensor is amazing when actually working. It will report either changes to the angle, or rate of change to the angle of the sensor with a decent amount of accuracy (plus or minus a degree or two).

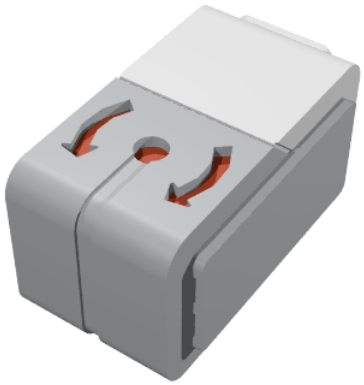

Figure 3: EV3 Gyroscope Sensor

During testing it was noted that an error that causes a sensor drift occasionally appears where the sensor reading will slowly increment or decrement. Using the Lego test suite, this issue could be solved by unplugging and replugging the sensor from the port. However, using the standard LeJOS test suite, unplugging the sensor resulted in an exception which it was not able to recover from even when sensor is replugged in. A restart of the brick may be required if this bug appears. This will be something of concern if future students wish to incorporate the sensor in a project. This appears to be a firmware issue, and not related to LeJOS, future firmware releases may

address this issue. Please note that this bug will appear straight away when the sensor is activated, it will not occur mid operation. So if the sensor is properly started and checked when the brick is turned on, it should not present a problem later during other operations. It is then advised to have a gyroscope test in LeJOS before starting any main programs.

The gyroscope instantly decreases the complexity of some of the robots algorithm as it no longer relies on tachometer readings to calculate it's current angle. The old way of angle calculation was very much prone to error as any wheel spin will cause errors in the result.

Using the rate of change mode, you can accurately detect how many degrees per second the gyroscope is moving. This can be used a number of different ways such as limiting the speed of rotation on an action etc.

2.2.4 Ultrasonic

Figure 4: EV3 Ultrasonic Sensor

Much more accurate than the NXT sensor. Testing against different objects under 40cm distance against a tape measurer revealed a much improved sensor in the new kit. The EV3 US sensor had much higher fidelity and returned measurements in 1mm increments although the accuracy can have a margin of error between +- 4mm when detected surface was front on to the sensor. The older NXT US sensor was only able to report in 1cm increments and much higher margin of error of +- 3cm.

The minimum range of the sensor was also much improved. The NXT sensor loses accuracy within roughly 12cm. EV3 sensor was able to maintain accuracy to within 4cm.

The new sensor's cone of detection remains relatively narrow, no different to the NXT version. And both sensors suffer decreases in accuracy when the detected surface is at a sharp angle. The EV3 sensor was able to maintain accuracy when detected surface is at no more than 30 degree angle to the sensor(where 0 degree means completely front on to the sensor).

2.2.5 Bump

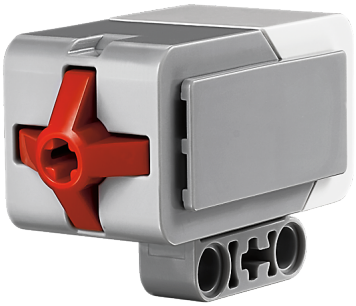

Figure 5: EV3 Bump Sensor

The EV3 bump sensor is physically shorter than the NXT equivalent. The EV3 sensor is 44mm while the NXT sensor is 48mm. They were identical in performance.

2.2.6 Backwards Compatibility

All NXT sensors can be used on the EV3 Brick with out issue. However, EV3 sensors will not be recognised by the NXT brick.

2.3 Motors

2.3.1 Large

The EV3 large motor is comparable with the NXT regulated motor. In testing the NXT motor is slightly more efficient and achieved higher RPM accross multiple voltages compared to the EV3 motor. The differences are negligible when considering the scope of this project.

2.3.2 Medium

The rotation sensor is a motor with less torque than a Large motor (typically used to move the robot) but with higher speed due to it's smaller diameter (it is capable of 260 RPM rather than the Large's 175).

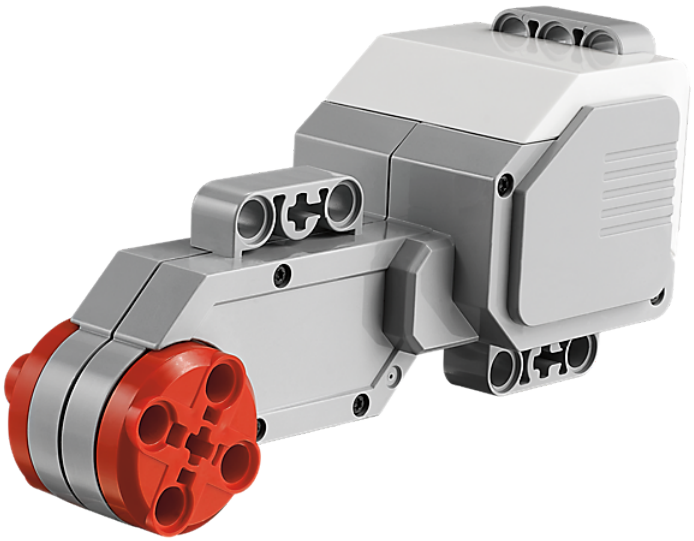

Figure 6: EV3 Large Motor

Possible problems: depending on the application it may be very important to keep track of the 'arm' being moved by the motors position. In tests it proved very easy to cause the shovel arm (design included in the Lego build manual) to collide with the motor itself which could potentially damage the internal mechanics of the motor.

Figure 7: EV3 Medium Motor

(insert image of collision here)

To counteract this, it may be necessary to 'reset' the position of the arm during device shutdown (this method description implies that the maintained

'position' of the motor will be reset upon a device reset)

The Rotation Sensor utilises a motor port, which as part of the project shouldn't be a problem as students will only be provided with three modules that can utilise the four ports.

2.3.3 Backwards Compatibility

All NXT motors are compatible with the EV3 brick. EV3 motors can function with the NXT brick. In both cases use NXTRegulatedMotor class.

2.3.4 Enhancement

As noted on Wikipedia (`https://en.wikipedia.org/wiki/Lego_Mindstorms_EV3#Enhancements`), the EV3 motors can be hacked to increase the resolution of the encoder. This required updating the firmware with a modified version. While there are significant benefits to this modification, it is most likely unnecessary for projects considered for this course. But having this option is a positive in favour of the EV3.

2.4 Building blocks

2.4.1 Pivot wheel

New ball bearing style pivot wheel design is excellent with omni directional movement. NXT pivot wheel design often resulted in uncontrolled shift in robot position when reversing or sharp turns. The new EV3 ball bearing wheel design completely solves this issue by allowing 360 degree movement without any mounting blocks moving. Still has issues when moving over edge of paper. Recommend sticky-taping test maps down to surface of table for any testing.

2.4.2 Frames

The addition of several pre-made rectangular frames greatly assists robot design and build. Many of the other pieces (such as the EV3 sensor modules) easily and logically slot in to varying positions on these frames to allow quick, intuitive and strong robot designs.

2.4.3 Sensor mounts

Every sensor now has an axle connector between two standard round holes. These axle connectors allow for quick mounting of any new sensor in essentially any desired position on the robot.

(a) Piece 4540797 (b) Piece 4539880

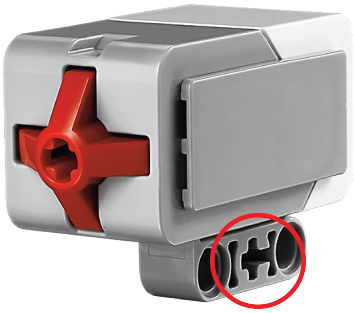

Figure 9: EV3 Sensor mount indicated by red circle.

3 Software

3.1 Eclipse

Each version of leJOS has been released alongside plugins for Eclipse which enabled a high level of integration between the leJOS library and the Eclipse IDE. The NXT block allowed for relatively simplistic functionality however the EV3 plugin is quite improved and allows for advanced functionality such as:

- The ability to push code straight to the EV3 brick with minimal configuration (almost none is required if a USB connection is utilised). **TODO:** add image describing the simplicity of pushing code to the EV3 block
- Fully leJOS API documentation; meaning to see what a method does it's only necessary to hover over the method signature with the mouse (rather than investigating it in an internet browser window).

3.2 Java RMI

The EV3 version of the leJOS library allows EV3 devices to be accessed through a RMI (**R**emote **M**ethod **I**nvocation) interface; essentially this means that a PC (or other device) can call methods on the EV3 as if it were actually the EV3 itself! An example of this is included in the figure

below:
TODO: insert demo of RMI code

```
package rmiTestEv3;

import java.net.MalformedURLException;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

import lejos.hardware.Sound;
import lejos.remote.ev3.RemoteEV3;

public class Driver {
    public static void main(String args[])
            throws RemoteException, MalformedURLException, NotBoundException
    {
        RemoteEV3 ev3 = new RemoteEV3("192.168.1.107");
        ev3.setDefault();
        Sound.beep();
    }
}
```

Figure 10: A complete EV3 program which connects and sets the connected EV3 as the default device and triggers a Beep sound from the EV3s embedded speaker

If the programmer on the PC issues a command which conflicts with the current EV3 directive the new command will be queued to be enacted after the current command. For example if the EV3 has autonomously decided to move forward 10 centimetres and the programmer told the robot to move forward 3 centimetres the robot will move 3 centimeters **after** moving the previous 10 centimeters.

- Explain what the RMI model involves
- Explain what this enables an EV3 programmer to do
- Give an example of a code block for this

Show examples, demonstrate different paradigms (e.g. setting one device as the 'default' block vs keeping a hold of several blocks)
TODO: example of obtaining a device connection in new leJOS compared to old

3.3 leJOS

Previously the leJOS Java Virtual Machine has been utilised within the SEP course in order to provide an interface for programming the NXT using the

```
void connect()
{
    /*
     *  Initialises a connection between the PC and the robot
     *  Returns nothing; sets internal variable
     */
    try{
        NXTComm nxtComm = NXTCommFactory.createNXTComm(NXTCommFactory.USB);

        NXTInfo[] nxtInfo = nxtComm.search(null);

        if(nxtInfo.length>0)
        {
            NXTInfo current = nxtInfo[0];
            boolean truth = nxtComm.open(current);
            if(truth){
                this.addLogMessage("Connection obtained");
                System.out.println("Connection obtained");
                this.out = nxtComm.getOutputStream();
            }
            else
            {
                this.addLogMessage("Connection could not be obtained");
                System.out.println("Nah");
            }
            this.connected = this.isConnected();
        }
        else
        {
            this.addLogMessage("No NXT blocks detected...");
            System.out.println("No NXT blocks connected...");
        }
    }
    catch (NXTCommException e) {
        this.addLogMessage("NXT Comm Exception - is a device connected?");
        e.printStackTrace();
    }
}
```

Figure 11: A method inside a NXT program for obtaining a connection to the NXT device. Note the use of four custom classes *just* for obtaining a connection.

Java programming language. However as well as solid functions the leJOS NXT library had incomplete, untested and unimplemented methods, all of which handicapped SEP project development.

Given the incomplete nature of the previous library we were at first apprehensive, but given exposure to the leJOS EV3 library we are very satisfied with the progress that has occurred since the previous library.

- New features supported
- QWERTY on-screen-keyboard is a little unwieldy

- Longer boot time can be frustrating, but this is offset by the decreased need to reboot due to ease of uploading software from Eclipse and increased stability in LeJOS.
- LeJOS now comes with full sensor test suite. This is great for the early stages of the project as groups can test the sensors straight after receiving the kit. This should assist in better robot designs. This will also allow each group to test their sensors and discover any faulty hardware early in the project.

3.3.1 Installation

See the following URL for installation details.
`http://sourceforge.net/p/lejos/wiki/Installing%20leJOS/`

A micro SD card is required for LeJOS to operate on the EV3 Brick. The older NXT Brick which lacked any expansion ports, had to replace the existing Lego software with LeJOS. The EV3 however can boot off the SD card directly, there by retaining the previous OS. However, this does mean that one micro SD card is required for each EV3 brick to operate LeJOS.

Creating the bootable SD card was simple on both Windows and OSX computers. There are a number of written and video tutorials available.

3.3.2 Operation

Operation is simplistic; the hardest part is to insert a properly formatted SD card into the EV3 device and after which it will automatically launch after the initial firmware boots.

4 Connectivity

A core feature of the NXT was its connectible nature; it could be programmed and controlled by a wide range of heterogeneous hardware and operating systems (including Windows, OSX and other Linux-based systems).
The EV3 has built upon this fundamental property by introducing a new connection method (WiFi based connections) and by improving upon existing connections (for example by increasing the speed of USB connections enormously and by streamlining Bluetooth connectivity). The change from the use of a proprietary operating system to a Unix-based one has also enabled greater connectivity; EV3 blocks can obtain and hold IP addresses which make them incredibly easy to address and connect to. Additionally the devices can also be SSHed directly to enabling very deep configurability of the OS and robot.

4.1 WiFi

- Requires the use of a WiFi dongle to activate WiFi capabilities.
- Plug and play functionality (supports several mainstream chipsets)
- leJOS has a 'connection' menu item, IP address is clearly displayed
- Connections can be configured on the Micro SD image to avoid repeated configuration

4.1.1 OSX

Tested, functional

4.1.2 Windows

4.2 Bluetooth

4.2.1 OSX

- Still unable to simply connect via Bluetooth
- Forum posts show that other users' efforts are similarly problem affected

4.2.2 Windows

- Easy Bluetooth connection using Windows 7
- Windows 8 connection slightly more tricky. Need to install the device as a Bluetooth device then go to "Devices and Printers", right click on the EV3 icon and select Connect As -¿ Access Point.
- Once connected, the EV3 brick has its own IP address. You can ping this to check connectivity.
- Can use SSH to copy files to and from the brick.
- Highly recommend Eclipse EV3 plugin as it simplifies the whole uploading process.

4.3 USB

- Device now has a USB host port (supports 3 levels of daisy chaining)
- Addition of standard USB port helps facilitate the use of a WiFi dongle and also allows for USB storage
- The new USB speed is a blistering 480 Mbit/s (compared to the NXTs 12 Mbit/s)

4.3.1 OSX

4.3.2 Windows

5 Conclusion

5.1 Feasibility

There is no doubt in our minds that the EV3 will be a great replacement for the NXT for the purpose of future projects. Every aspect of the NXT has been improved upon.
Increased sensitivity or functionality in all sensors greatly increases accuracy and stability of the robots functions. The changes in the API makes coding more intuitive.
The improved API should drastically decrease the difficulty of the project.

5.2 Recommendations

Line following sensor combinations

- Single colour sensor as line and underground object detector. Gyroscope sensor for robot positioning feedback. Ultrasonic sensor for above ground object detector. Single bump sensor with a wide bar attached could be used as a collision detector to further improve on aboveground object detection.
- Single colour sensor as line and underground object detector. Gyroscope sensor for robot positioning feedback. Two bump sensors mounted on different parts of the robot for collision detection.

5.3 Additional project ideas

These are ideas which may be suited for post-graduates.

- The EV3 brick comes with 4 motor ports, while 3 motors are supplied(2 large and one medium). One idea could be to cannibalise some of the older NXT kits which contains 3 motors each and give the post-grads an extra NXT motor to accomplish an additional task such as picking up an object and placing it elsewhere or planting a flag on a designated area.
- The additional motor can be used to change orientation of some sensors to accomplish different tasks. Such as rotating the colour sensor from a vertical position during line following to a horizontal position when an above ground object is detected and it has to then detect the color of the above ground object and change its behaviour based on the detected colour.

- Both the light and colour sensor also has a decent Ambient light sensing mode. This can be utilised to perhaps have the robot find a source of light on the map or alternatively if the light source is made to flickers on and off in a programmed sequence, it can be used to transmit binary or Morse code for the Robot to maybe detect and interpret.
- Providing a HTTP-served interface or data-view using the EV3 Brick as a web server.
- The gyroscope can be mounted on a motor arm which can change the orientation of the gyroscope to measure angle changes on multiple axis. This might be interesting if the robot is made to move over certain types of terrain. Or perhaps use the rate of change as a indicator of how quick the robot is turning and set either limits on the rate of turning to make sure the robot is slow and steady, or set goals for rate of turning to force robot in to quicker movements.

5.4 Possible future project

A "Google car" project where the Robot has to stay within two lines on a map which simulates a lane on a road. This would be a good use of the light and colour sensors on each side of the robot or if one of these sensors is mounted on a good length moving arms which can alternated between each side of the robot. The ultrasonic sensor can be used to detect objects in front of robot to avoid collision or "park" within a certain distance to an object or even keep a minimum distance to another robot.

The robot needs to stop at intersections which can be denoted by either a series of lines perpendicular to the lane lines or possibly a particular coloured line at the intersection(yellow and red). The detected colour will trigger either a stop behaviour or a go behaviour. Once stopped perhaps a green colour card can be manually placed under the colour sensor to indicate it can now move forward to continue operations.

Additional requirements could be, the robot needs to perform tasks in a certain time constraint or speed. It may have to travel at a minimum speed to simulate speed limits. It might have to demonstrate different speeds depending on what speed zone it is currently in. Speed zones might be able to be represented by different coloured lanes.

Automatic parking behaviour can also be part of the project. If the robot is given a command it might have to look for a parking space, drive past it and then reverse parallel park in to it or something like that. This might involve scanning the parking area with the ultrasonic sensor to make sure

the robot can fit in the space.

Multiple robots from different teams can even be placed on the map at the same time(assuming the map is of sufficient size) to make sure they behave correctly in the presence of moving objects.

Some of the requirements could be given to post-grads only. The test map would probably need to be A0 in size to allow a decent test area.

6 Revision History

Version	Date	Author	Description
1.0	15 Dec 14	Xiang Guo (a1077337)	Created Draft 1

TODO: Synchronise this with GitHub revision history