Android Application to Assess Smartphone Accelerometers and Bluetooth for Real-Time Control

M.A. Nugent
Faculty of Engineering and Computing
Dublin City University, Dublin 9, Ireland

Dr. Harold Esmonde
Faculty of Engineering and Computing
Dublin City University, Dublin 9, Ireland

***Abstract*—Modern smart phones have evolved into sophisticated embedded systems, incorporating hardware and software features that make the devices potentially useful for real-time control operations. An object-oriented Android application was developed to quantify the performance of the smartphone's on-board linear accelerometers and bluetooth wireless module with a view to potentially transmitting accelerometer data wirelessly between bluetooth-enabled devices. A portable bluetooth library was developed which runs the bluetooth functionality of the application as an independent background service. The performance of bluetooth was tested by pinging data between 2 smartphones, measuring round-trip-time and round-trip-time variation (jitter) against variations in data size, transmission distance and sources of interference. The accelerometers were tested for sampling frequency and sampling frequency jitter.**

Keywords—Android; Bluetooth; control; real-time; sensors; smartphone

I. INTRODUCTION

Smartphones evolved from the PDAs of the late 1990s. PDAs were handheld computers essentially used for organising information. They were equipped with small keyboards that the user could utilise to input information. IBM Simon was the first PDA with mobile phone functionality and can be considered the first smartphone. In 2007 Apple Inc. introduced the iPhone which incorporated a large multi-touch screen for direct finger touch input as its main method of interaction.

Mass-produced ARM based microprocessor technology delivers high speed multi-processing on an inexpensive, battery powered platform, that only a decade ago, industrial computers would have been envious of. Smartphones have a myriad of onboard sensors, such as motion sensors including accelerometers and gyroscopes, environmental sensors that can measure pressure, light, temperature and humidity as well as position sensors such as orientation sensors, magnetometers and GPS locators. The original purpose of the smartphone was for communication and smartphones have expanded their capabilities here also including bluetooth and infrared.

Smartphones comprise 2 operating systems, a low-level operating system that handles the drivers for the hardware and a higher level user-interfacing operating system. The most common operating system installed worldwide is Google Inc.'s Android operating system which is built on top of a Linux kernel.

Android was first established in 2003 with the aim of developing a more user oriented operating system than Symbian and Microsoft. The main advantages of Android over its rivals are its flexibility and upgradability. Android has grown in popularity among consumers and developers alike to the point where an industry survey [1] in 2013 shows that 71% of all mobile development is for the Android operating system.

Since its inception there have been many evolutions of the Android operating system dashboard, from the original 'Froyo' through to 'KitKat' shipped with new smartphones. Table 1 displays the various distributions of Android dashboards.

TABLE I. ANDROID DASHBOARDS AND DISTRIBUTION LEVELS

Version	Codename	API	Distribution
Froyo	2.2	8	0.7%
GingerBread	2.3.3 – 2.3.7	10	13.6%
Ice Cream Sandwich	4.0.3 – 4.0.4	15	10.6%
JellyBean	4.1.x	16	26.5%
JellyBean	4.2.x	17	19.8%
JellyBean	4.3.	18	7.9%
KitKat	4.4	19	20.9%

Developing an application that is compatible with API 10 and higher will guarantee coverage of 99.3% of the Android market, but older APIs are not compatible with newer android features. Some features are only available with more recent APIs due to continuous developments by Google. Developers need to be aware of the features available with each version and the size of the market associated with that version. The bluetooth application on which this paper is based is compatible with all dashboards from Froyo to KitKat.

The Android accelerometers are primarily intended for screen orientation and game play. Android [2] describes the accelerometer sampling periods in terms of data delays in sending sensor readings to the application, ranging from 200,000 microseconds for the 'Normal' sampling rate to 0 microsecond delay for the 'Fastest' sampling rate. The delay is only a suggested delay and the Android system and other applications can change it.

Bluetooth is a wireless communication protocol invented by Ericsson in 1994 as a wireless replacement for serial port communications between mobile phones and headsets [3, 4]. Management of the specification passed to the Bluetooth Special Interest Group (BSIG) in 1998 and Bluetooth 1.0 was released in 1999 with a data rate of 721 kbit/s. Bluetooth 2.0 Enhanced Data Rate (EDR) was adopted in 2004, providing a data rate of 1 Mbit/s without EDR and 3 Mbit/s with EDR, and coinciding with the landmark of 3 million product shipments per week mark. In 2009 Bluetooth 3.0 High Speed (HS) was adopted with a data rate of up to 24 Mbit/s and Bluetooth 4.0 Low Energy was adopted in 2012 when annual product shipments exceeded 2 billion. Current specification development is in the area of IP connectivity preparing bluetooth for the Internet of Things revolution.

Bluetooth is a low energy, short-range, short wavelength radio transmission protocol operating within the unlicensed ISM radio frequency band from 2.4 – 2.485 GHz. A bluetooth radio can have a range from 1 meter up to 100 metres, depending on the class of device with smartphones typically ranging up to 10 metres. Once connected, a small network called a piconet is dynamically created which allows a master device to connect with up to 7 slave devices [5]. Each device can be connected to multiple piconets simultaneously allowing for complex, wide-ranging connectivity. One of the main advantages of bluetooth networks is their ease of set up. Two devices can connect with the push of a button with little configuration required from the user.

Research [6, 7] has been carried out into the performance of bluetooth over varying distances, data sizes and sources of interference, but the data sizes tested were large (11 kB - 5000 kB) in comparison to the 4 bytes typical for sensor readings. There is an exponential correlation between data size and transmission times with data size having negligible effect for smaller data sizes and a much greater effect at large data sizes. There is also a direct correlation between distance and transmission times for large data sizes with negligible effect of distance for smaller data. This paper specifically assesses the effect of distance and data size when sending small data packets consistent with the transmission of sensor data in real time control applications.

Other conclusions from [6] are that concrete walls and metal barriers reduce the effective range of bluetooth to 3 metres, and that transmitting large data, and direct sunlight reduces the effective range to 4-7 metres. Interestingly, wi-fi had no effect on transmission times for data sizes less than 100 kB. Delay variation was measured in [7] and found to be greater than 16% which is less than the industry recommended maximum of 20%. No difference was found in transmitting between mobile phones, pcs or computers.

Some testing [8] was carried out into streaming of MIDI music files via bluetooth with message lengths of between 1 and 6 bytes, with particular emphasis on comparing delays when master and slave device transmit. The result is that the master transmits with mean delay of 30 mS and standard deviation of 10 mS compared to the slave transmitting with mean delay of 20 mS and standard deviation of 20 mS. The discrepancy is occurring because of the different permissions of the master and slave devices in when they can transmit.

II. Android Software Stack

The Android architecture is a software stack comprising applications, a Linux operating system, a runtime environment and various services and libraries. Each layer in the stack and each component in each layer are tightly knitted together to provide a very effective application development and execution platform, as depicted in Fig. 1.

At the bottom of the stack is the Linux 2.6 kernel. Its role is to abstract the hardware into low level software that the higher layers can interact with. It achieves this through hardware device drivers and low level power, process and memory management. Linux is an open source operating system that has been around for decades with widespread application in servers, embedded systems and robotics due to its reliability, efficiency and modularisation of hardware drivers that can be loaded and unloaded while the system is operating.

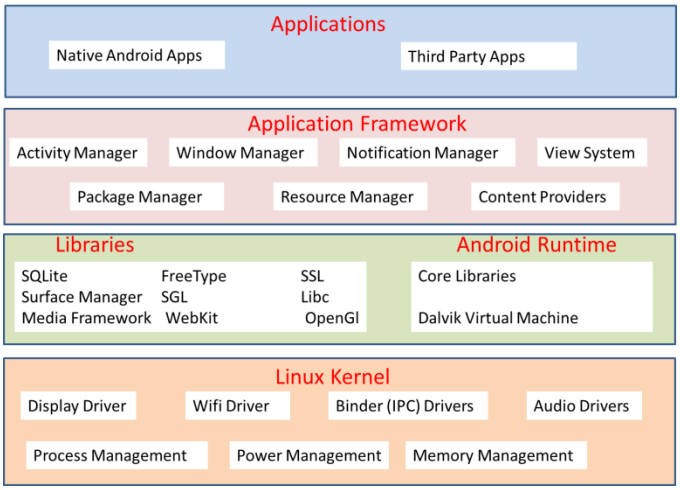

Fig. 1. Android operating system architecture

Each application running on Android is executed on an instance of the Dalvik Virtual Machine. Each application is effectively isolated from every other application, from the operating system and from the hardware device drivers. So each application is developed to run on the Dalvik VM rather than a particular hardware platform. The operation of Dalvik VM is similar to Java VM but more efficient in terms of memory usage and processing power requirements and thus more suitable for smart phones.

The application framework provides already developed support tools for the application while it is running and the basic resources required for an application to run, enabling the developer to program at a higher level. Low level tasks are automated such as the construction, management and end-of-

life clean-up of an activity, the package file structure of an application, and access to common resources. It contains the graphical views that the user would use for the GUI and content providers to share data between processes.

III. BLUETOOTH PROTOCOL

The industrial, scientific and medical (ISM) bands are ranges of radio frequencies reserved internationally for devices that generate electromagnetic emissions that have the potential to cause interference with telecommunication equipment. Devices that can generate electromagnetic emissions, such as microwave ovens, RF heaters and medical diathermy machines, are required to limit their power emissions in these frequency bands. Telecommunication equipment sensitive to electromagnetic interference should avoid these frequencies. However ISM bands have become popular for short-range radio frequency communications like Bluetooth and Wi-fi LAN networks where the potential for interference is limited by their short broadcasting ranges.

The bluetooth channel is a pseudo-random frequency hopping pattern of 79 channels, each with a bandwidth of 1 MHz, within the 2.4 GHz. ISM band, operating between 2.402 – 2.78GHz. The hopping pattern is determined by an algorithm using the address and clock of the master device, to which all devices in the piconet are connected and synchronized. A packet of data will be transmitted on a channel and then each device will switch to the next channel in the frequency hopping pattern before another packet is sent.

Bluetooth incorporates some features to make it more resilient to interference and data loss. Adaptive Frequency Hopping Spectrum is employed to dynamically alter transmission frequencies to avoid frequencies where there is interference. Operating within the ISM band, interference can be expected from other bluetooth devices, IEEE 802.11 WLAN and microwave ovens. The frequency hopping pattern determined by the master device's address and clock can be changed dynamically to avoid frequencies where poor performance due to interference has been detected.

Communication over the channel is serial in nature but parallel communication is achieved by creating time slots to share transmission time, called time-division-duplex (TDD). Each time slot is 625μs in length giving a nominal hopping rate of 1600 hops/sec. Master and slave devices take turns to transmit; the master device transmits in even-numbered time slots and the slave device transmits in odd-numbered time slots. The hop frequency remains constant for the duration of the transmission. When the transmission has completed the channel changes frequency to the next hop frequency in the pattern and the other device transmits.

Large data files will be broken down into packets small enough to be transmitted in one time slot. Each packet consists of a header and payload. The header contains information for channel maintenance and error detection codes and the payload contains user data being transmitted. However packet construction is dynamic where size and composition can be adapted to the conditions. Table II gives a breakdown of the various data packets that can be used. DM1 refers to a small packet designed to be transmitted in one time slot with error detection (FEC) overhead in the header. DH5 refers to a large packet designed to be transmitted in 5 consecutive time slots with no FEC overhead in the header.

TABLE II. PACKET TYPES

Type	Header (Bytes)	Payload (Bytes)	FEC	CRC
DM1	1	0-17	2/3	YES
DH1	1	0-27	NO	YES
DM3	2	0-121	2/3	YES
DH3	2	0-183	NO	YES
DM5	2	0-224	2/3	YES
DH5	2	0-339	NO	YES
AUX1	1	0-29	2/3	NO

Bluetooth employs 3 error detection techniques. FEC1/3 (forward error correction) simply repeats each bit in the header of the packet 3 times. Errors in the header can be easily detected if the bits are not in triplicate and corrected by majority vote. FEC2/3 is a shortened type of Hamming code implemented by appending 5 parity bits to the end of each 10 bit word, making it a 15 bit word. It can correct all single errors and detect all double errors. CRC code (cyclic redundancy check) is used on the data payload in the packet to check it's integrity by referencing the remainder of a polynomial division calculation on the bits in the payload. ARQ (automatic retransmission request) ensures packets will be re-transmitted until an acknowledgement is received from the intended recipient device of a successful, error-free transmission. Error checking overhead can add to transmission delays therefore there is a trade-off between the dual objectives of transmission speed and transmission reliability when using bluetooth.

IV. APPLICATION DESIGN

The application assesses the Android accelerometers and bluetooth module independent of each other. By assessing them independently, their individual contribution to the collective performance when transmitting real-time sensor data wirelessly could be quantified.

Although Android provides the BluetoothAdapter class as an abstraction of the bluetooth hardware, the process of working with bluetooth programmatically is still quite complicated. Because this application was built to assess bluetooth for real world applications, it was decided to build the bluetooth component as a reusable library that could be imported into any future project requiring bluetooth connectivity. Bluetooth operations are effectively simplified to creating an object of BluetoothLibrary and making the correct method calls and interface implementations.

Taking these points into consideration, the application has 3 components;

- An activity to measure the performance of the accelerometers
- An importable BluetoothLibrary to manage the bluetooth connection
- A set of activities to conduct the bluetooth performance assessment

A. *Sensor sampling period testing*

Within the main activity the user can select the sensor testing activity. There are 4 programmable sampling periods for the linear accelerometers. These correspond to the delay in Android sending the sensor data to the application. 'Fastest' corresponds to no imposed delay in sending the reading to the application, 'Game' corresponds to an imposed delay of 20 mS, 'UI' corresponds to an imposed delay of 70 mS and 'Normal' corresponds to an imposed delay of 200 mS. The user can select the sampling period via the radio buttons, Fig.2. By pressing on the 'Begin Test' button the application begins polling the accelerometers. When the test is complete the mean sampling period and standard deviation of the sampling period are posted to the screen. The user can choose to save the results to a csv file in the smartphone's primary memory location.

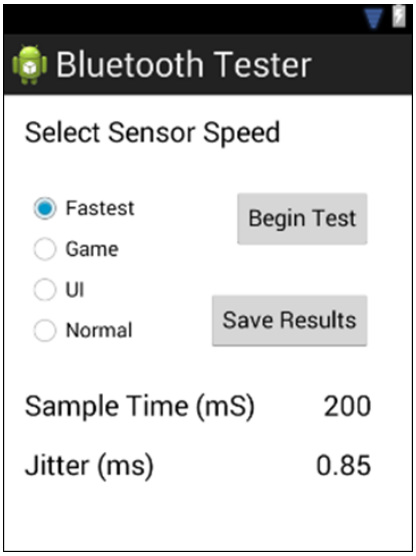

Fig. 2. Sensor sampling period testing activity

Fig.3 is a graphical representation of the operation of the test. Sensor data is received from the onSensorChanged call-back method on the main UI thread and the current time of that event is recorded. The sensor value itself is unimportant for testing, just its timestamp. The timestamp is sent to a parallel thread where all calculations and screen updates are processed in parallel to avoid blocking the callback method in the main thread. This is particularly important when the sampling period is set to 'Fastest' or 'Game' where the GUI can hang due to blocking of the sensor callback. The timestamps from the sensor readings are buffered to avoid overwhelming the run method of the thread. Within the thread calculations are performed to determine the sampling period and a running average of the sampling period is displayed on the screen. Also within the thread the sensor timestamp and period are inserted into a SQLite database for temporary storage and can be saved to the phones memory card for further analysis if the user so wishes. Upon completion of the test the standard deviation of the sampling period (jitter) is calculated by iterating through the SQLite database and displayed on the screen.

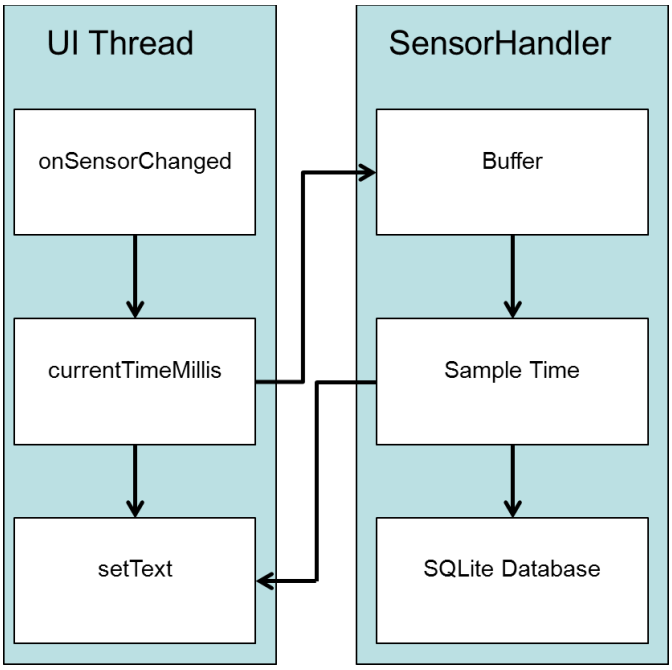

Fig. 3. Operation of the sensor testing activity

B. *Bluetooth Library*

All of the bluetooth related operations that are valid for Android API 8 and that were required for this project are contained within the bluetooth library. This library enables turning bluetooth on and off, making the device discoverable, enabling discovery of other devices, connecting with up to 7 other devices and establishing the input and output streams of a bluetooth connection. Within the bluetooth library is a class, BluetoothLibrary, which contains all of the public methods required for the bluetooth operations. The bluetooth library can be imported into any Android application requiring bluetooth functionality. The structure of the library is shown in Fig. 4.

The connectivity part of the library is contained within its own service. A service in Android is independent of the lifecycle of any activity of the application or the application itself with the advantage that it will allow the established connections to be maintained between activities. Otherwise if the user switches between activities, the activity that established the connection would be paused or destroyed and the connection would be lost.

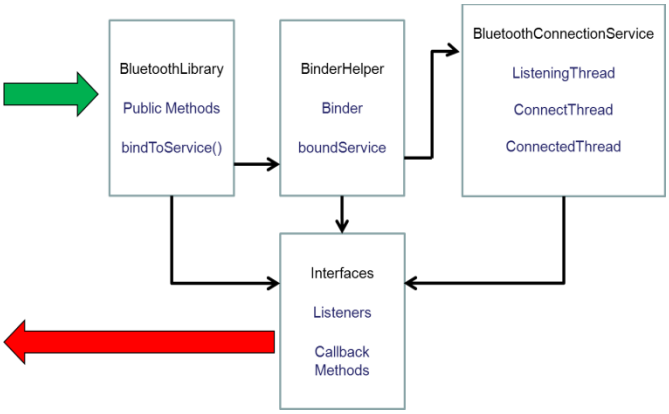

Fig. 4. Structure of the portable library – 'BluetoothLibrary'

The service contains 3 threads, a listening thread which listens for a connection attempt and blocks on the accept method, a connect thread which tries to connect and blocks on the connect method and the connected thread which sets up the input and output streams and blocks on the read method. A typical server operation will listen for a device on the listening thread before switching to the connected thread when it accepts a connection whereas the typical client operation will try to connect on the connect thread before switching to the connected thread to manage the connection.

If a programmer using the library wishes to do anything bluetooth related, they should create an object of BluetoothLibrary within the activity and then call its public methods. If they wish to perform a connection related operation such as checking if a thread is running they will need to bind to the service by calling the bindToService method in the onResume method and unBindFromService method in the onPause method. The call to bindToService is asynchronous which means that the next line of code will be executed before the activity is bound to the service, potentially crashing the application. The programmer can avoid this problem by implementing the onBindListener interface that provides a callback when the activity is bound to the service. Other interfaces are available for turning bluetooth on/off, discovering new devices and receiving data on the input stream.

C. Bluetooth performance testing

The main activity of the application allows all of the normal bluetooth operations to be performed – turning the bluetooth radio on/off, making the device discoverable by other devices, scanning for other devices and initiating a connection. These can be achieved by using the imported bluetooth library. Once two devices are connected, the bluetooth testing activity can begin.

Fig.5 shows the operation of the bluetooth testing activity. One smartphone takes on the role of client and the other smartphone takes on the role of server. The client device sends data to the server device who then returns the data to the client. The client device then calculates the round-trip-time (rtt) for the transmission. This procedure is repeated for 30 seconds until the testing automatically stops.

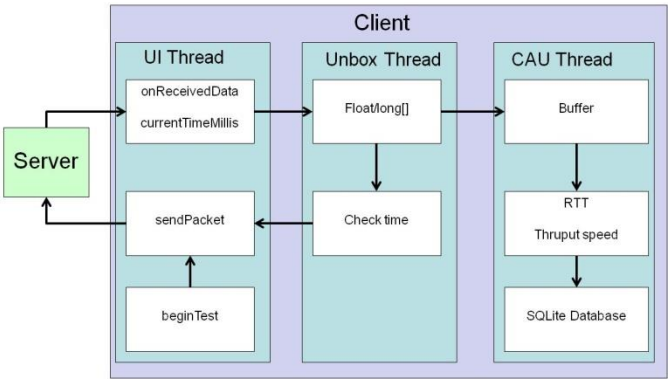

Fig. 5. Operation of the bluetooth performance test

When the transmission of data begins the current timestamp is retrieved from the client system clock. A timestamp is of type long (8 bytes) and it is the timestamp that will be transmitted in the experiment. Depending on the size of data under test, a long array is constructed consisting of the timestamp and filler material, with the exception of the 4 byte data payload size which is tested differently. The data payload is constructed at the beginning of the experiment and sent from client to server and back again in a 30 second loop. Each time the payload is transmitted by the client the current timestamp is retrieved from the client's system clock and inserted into the start of the long array.

Upon receipt of the payload the client retrieves the timestamp and sends it to a thread to perform some calculations, screen updates and data storage while it resends the packet with the new current timestamp. Within parallel threads the round-trip-time is calculated, a running average of the round-trip-time is updated on the screen and the new data is inserted into the SQLite database for storage. The data rate is calculated from the round-trip-time and packet size. Just like in the sensor sampling period test, the network jitter statistic is determined by iterating through the database and calculating the standard deviation of the round-trip-time. The results screen from this part of the application is illustrated in Fig. 6. Multi-threading is used to avoid blocking the main UI thread and slowing the performance of the transmission.

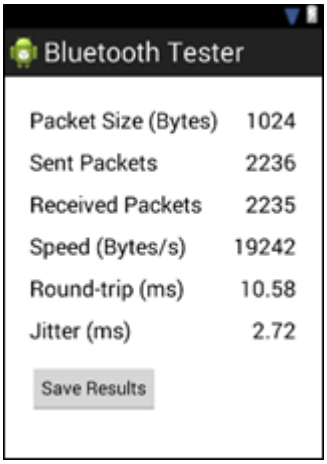

Fig. 6. Client results screen

In the case of the 4 byte test, the long array payload containing the timestamp when the payload was sent is not an option. If the user selects the 4 byte test the payload sent is an auto-incrementing integer which identifies the payload. The timestamp for when the 4 byte payload was sent is recorded in a long variable elsewhere. Apart from that difference, the remainder of the 4 byte test is the same as for the rest of the payload sizes.

V. Testing

The smartphones used in testing were a Samsung S3 running JellyBean and a TCL V860 running GingerBread.

A. Sensor Sampling Period Testing

The procedure for testing the sensor sampling period is straightforward. The user can select one of 4 options for the Android based sampling period. When the test is started a running average of the sampling period is displayed on the screen and upon completion of the test, the standard deviation (jitter) of the sampling period is determined. During tests the running average of the sampling period converges on a value and further testing is not required as the running average will not change significantly from this value. The data stored in the SQLite database is the timestamp of the sensor reading, the period since the previous reading for each sample, a mean sample period and the mean jitter for the experiment as a whole.

B. Bluetooth Testing

The testing of the bluetooth medium was carried out indoors where it is envisaged bluetooth will be used most of the time. When testing bluetooth's performance the factors examined are distance, data payload size and sources of interference. The maximum distance that class 2 bluetooth devices are operable at is 10 meters. Distance between the 2 devices is varied at intervals of 1, 3, 5, 7, and 9 metres. Payload size is varied at intervals of 4, 8, 64, 256, 1024, and 2048 bytes. 4 bytes is the typical size of a sensor reading float value or an integer value and 8 bytes is the size of a long value such as the timestamp.

Testing is carried out in the presence of no interference, an 802.11 wi-fi wireless router and a microwave oven. The testing is carried out for each payload size, at each distance and each source of interference.

VI. Results and Analysis

A. Sensor sampling period testing

The sensor sampling period was tested as outlined in the previous section. The sampling period for all sensor events was stored in the database and graphed for the 4 programmable sampling periods in Android. The mean sampling period was calculated in real time and the jitter was calculated from the sampling periods in the database. The results of testing the 'Normal' (200 mS, 5 Hz) sampling rate are presented in Fig.7.

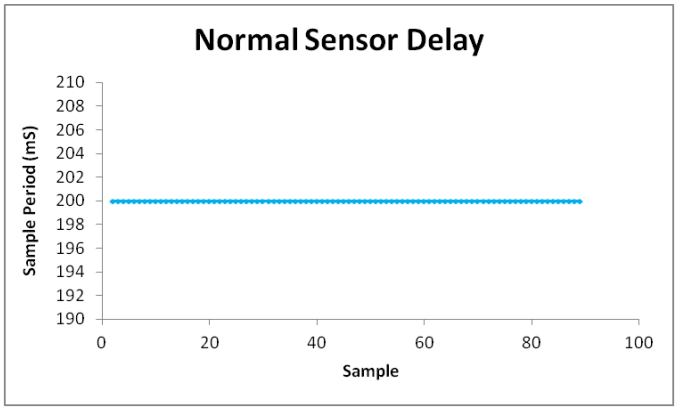

Fig. 7. Performance of 'Normal' sensor sampling rate

'Normal' represents the lowest sampling frequency and longest sampling period which is programmable in Android, 5 Hz and 200 mS respectively. It therefore puts the lowest strain on both the hardware and software. There were approximately 90 readings taken and the performance was consistent for all samples. Jitter was measured at 0 mS in this test. Although the jitter performance of the accelerometer was excellent in this test, there is limited use for such low frequency sampling. Possible uses are measuring the movement of large structures, recording seismic activity and tall building reactions to seismic activity and wind conditions.

Fig.8 presents the results of the 'UI' sensor sampling rate. UI aims to sample the accelerometers every 70 mS (14.3 Hz) which is almost 3 times faster than the 'Normal' sampling rate. From the graph the performance of Android at 'UI' is very good and consistent for the most part. There were 176 sensor samples taken and there were 2 inconsistencies at around sample no. 65 and no. 175.

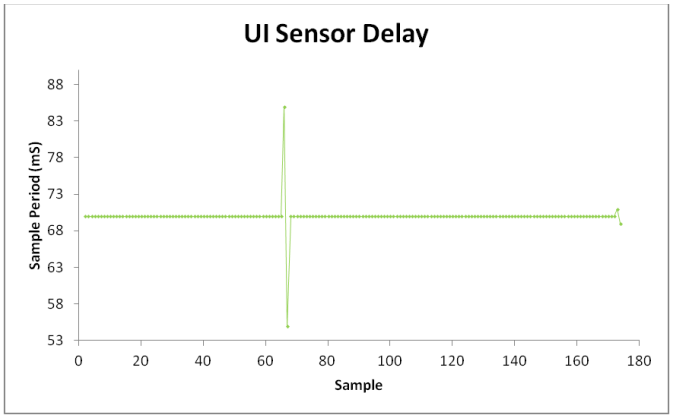

Fig. 8. Performance of 'UI' sensor sampling rate

Sample no. 65 was polled 85 mS after sample no. 64 which is 15 mS late. Also sample no.66 was polled 55 mS after sample no. 65 and 140 mS after sample no.64. This result demonstrates that Android schedules to sample the sensors at

regular time intervals from an initial setpoint, likely to be when the sensorManager is initialized, rather than the previous sensor sample. A similar result is observed at sample no. 175. Android cannot maintain regular sampling of the accelerometer at 14.3 Hz possibly caused by software running in the background using the hardware and operating system resources. It can be deduced that the accelerometers are not given priority by Android when under load. The mean sampling period was 70 mS with a jitter statistic was 0.85 mS in this test.

Understanding this result will aid in understanding the results from testing the 'Game' and 'Fastest' sampling rates in Fig.8. Android claims that 'Game' samples at 20 mS or 50 Hz and that 'Fastest' is limited only by the operating system with no imposed delay. The results in Fig.9 demonstrate the measured performance at these sampling rates.

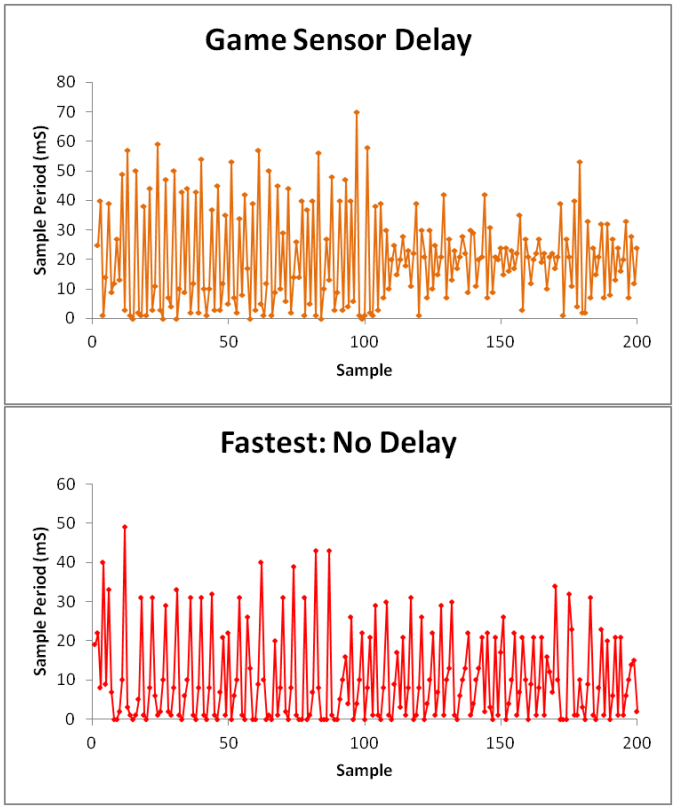

Fig. 9. Performance of 'Game' and 'Fastest' sensor sampling rates

In the case of 'Game', the mean sampling period is calculated to be 20.6 mS, very similar to the nominal value, but with a jitter of 7.8 mS. Sampling periods of 50 mS are not uncommon. Similarly, in the case of 'Fastest', the mean sampling period is calculated to be 5.06 mS with a jitter of 4 mS. Sampling periods of 20 mS are not uncommon followed by 2 or 3 samples taken within a couple of mS of each other. This sampling pattern is repeated throughout this particular test. Sampling period consistency is very poor at the higher sampling rates. The Android operating system and other applications have priority over system resources and interfere with sensor sampling.

B. *Bluetooth Performance*

The performance of bluetooth was tested as outlined in section V. The round-trip-times (rtt) from each experiment were saved in a csv file for analysis and graphing along with a calculation of the mean round-trip-time and standard deviation (jitter).

The effect of distance has on rtt is shown in Fig. 10. It can be seen that, for small data payload sizes, distance has no discernible effect on rtt when tested in an environment with no interference. However in the case of the two larger payload sizes, there is a step change of 10-14 mS in rtt performance improvement between 3 and 5 metres. At the application level it is not immediately obvious what low level bluetooth changes occurred to cause this step change. However bluetooth is a dynamic wireless transmission protocol, continuously changing frequencies, packet size and error correction overhead to improve performance. It is possible that in the case of data greater than 1 kb transmitted over distances greater than 3 metres that larger data packets were used, perhaps a 5 slot packet rather than a 3 slot packet used for smaller data sizes.

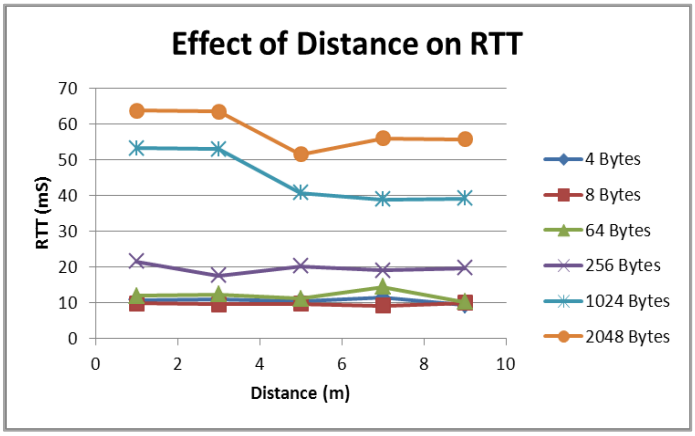

Fig. 10. Effect of distance on round-trip-time

Fig.11 shows the effect of payload size on rtt where it is apparent there is a strong correlation between payload size and rtt. Note the 7 metre curve is partially obscured by the 9 metre curve. Unexpectedly, the rtt performance of bluetooth is poorer at 1 and 3 metres for data payload sizes greater than 1 kb. The relationship appears to be non-linear but there is not a sufficient range of payload sizes to fully model the trend. The overhead induced by increasing payload appears more profound over the shorter distances. Larger data payloads require a greater number of packet transmissions. This effect is non-linear due to the dynamic nature of bluetooth packet assignment.

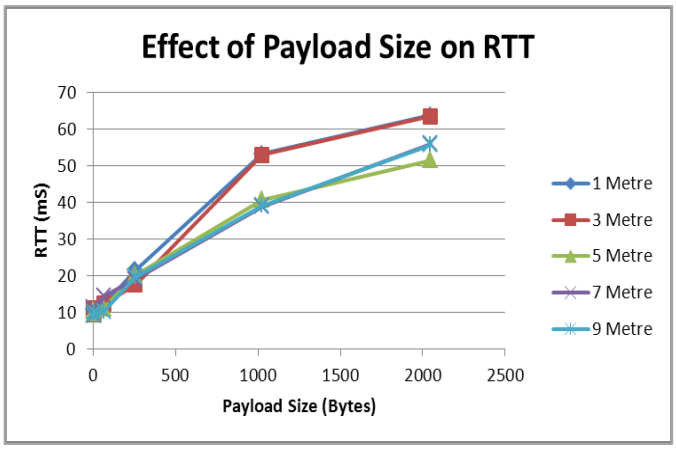

Fig. 11. Effect of data payload size on round-trip-time

One important result from Fig.11 is the offset on the rtt axis. Given that payloads of 4 and 8 bytes were tested, which are almost the smallest payload possible (only char is smaller at 2 bytes in size), there appears to be a minimum payload rtt overhead of approximately 10 mS per payload at the Android application level regardless of payload size. At the Android application level it will take the operating system a minimum of 10 mS to process the outgoing data and the incoming data for each payload in a round-trip scenario. The consequence of this result is that, in the case of round trips, the maximum payload frequency is limited to 100 Hz and an estimated 200 Hz in the case of a one-way transmission. This result suggests that Android is more suitable for single large data file transmission rather than multiple small packets.

From calculations of data rate, with payload size of 2048 bytes the data rate is in the region of 30-35 kB/s whereas for the 4 byte payload the data rate is approximately 800 bytes/s. Bluetooth is rated at 2.1 Mb/s and that may be possible when transmitting a single very large data payload. From Android's perspective most bluetooth users would be using bluetooth for transmitting large data files rather than bursty data of small size. This effect can also be seen in using bluetooth to stream audio where a noticeable latency can be detected.

The effects of wi-fi and microwave interference on rtt and jitter are shown in Fig.12. IEEE 802.11 Wi-Fi and microwave ovens operate within the same ISM band as bluetooth and have been identified as potential sources of interference. Microwave ovens operate at a fixed frequency of 2.45 GHz. IEEE 802.11 operates between 2.4 and 2.5 GHz and uses Direct Sequence Spread Spectrum (DSSS) to avoid interference.

From the graphs, only microwave interference appears to have an effect on rtt and an increasing effect for larger data payload sizes. There is a consistent jitter of 3-4 mS regardless of interference source or no interference which is to be expected from unprotected wireless transmission through the air.

However inconsistencies appear in the graph with wi-fi performing better than the interference free case in the rtt test which is unexpected. Further inconsistencies are apparent in the jitter graph where surprisingly wi-fi and microwave have more consistent round-trip-times than the interference free test.Interference by its nature is non-homogenous and inconsistent. Furthermore the interference free test is free of obvious sources of electromagnetic interference but it is not free of background radiation in the air. Therefore bluetooth's performance will be inconsistent and unpredictable dependent on the conditions of the environment in which it is operating.

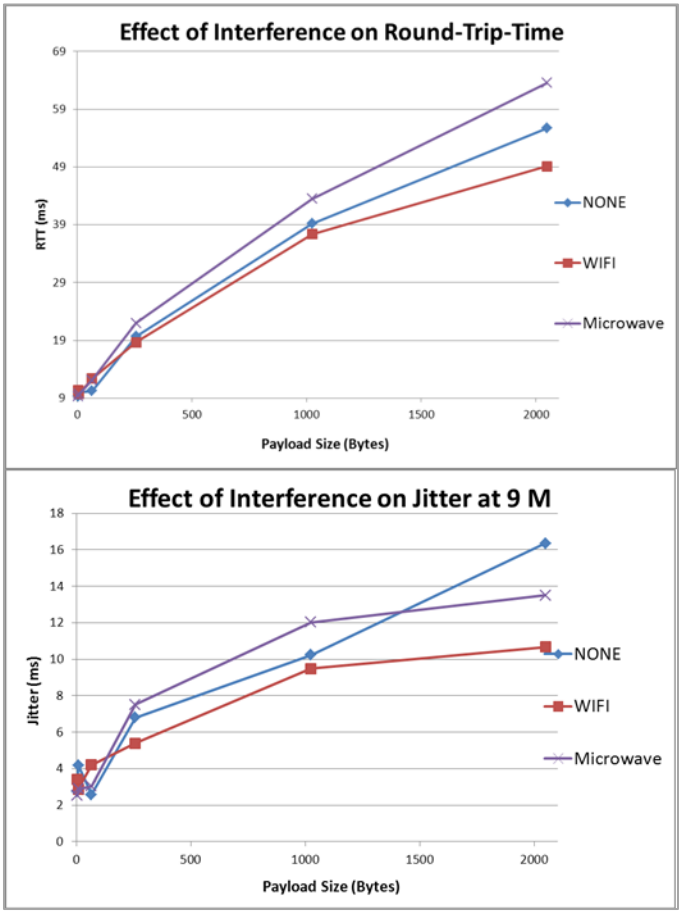

Fig. 12. Effect of interference on jitter at a range of 9 metres

The previous results in the interference tests were inconclusive due to the inconsistent nature of interference and bluetooth's adaptation to the transmitting environment and so further analysis of the data from the microwave oven test was carried out. Fig.13 shows the effects of microwave interference on rtt and jitter. Rtt is largely unaffected by distance from a microwave source. Previous results in Fig.10 had shown that distance alone had no effect on rtt. The jitter graph shows that microwave increases jitter when both smartphones are within 1-3 metres of the source.

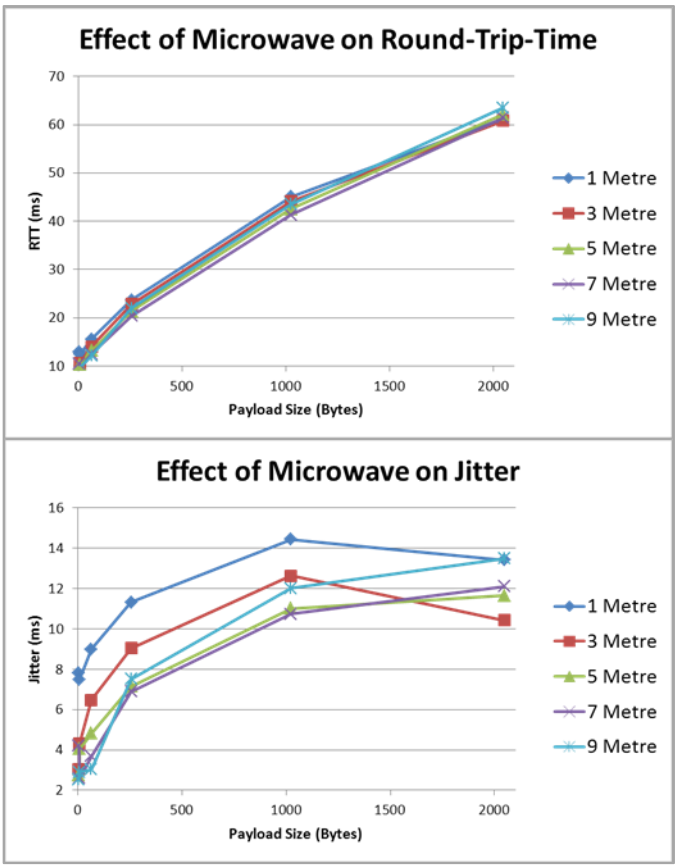

Fig. 13. Effect of microwave interference on round-trip-time and jitter at various distances

VII. Conclusion

This paper has determined that Android bluetooth is geared towards sending single large data files such as music or video or document sharing rather than high frequency small discrete pieces of data such as sensor readings. Bluetooth can transmit sensor readings of up to 64 bytes at 100 Hz round-trip, or 200 Hz one way. Distance has no effect on transmission time for a class 2 device within the 10 metre range. Transmission time increases non-linearly with increasing data size. Bluetooth is resilient to microwave and wi-fi interference.

The on-board accelerometers can only be consistently sampled at 14.3 Hz. The maximum mean sampling frequency is 200 Hz but with a standard deviation of 80%. The Android accelerometers are geared towards screen orientation and game play with low system priority given to the on-board sensors.

In terms of utilising Android smartphones' onboard sensor and bluetooth technology for real-time control applications, from the results of testing in this paper, it is estimated that the sensors reliable sampling limit is 14 Hz and that the sensors' output can be transmitted via bluetooth to another device within 5 mS over a range of 10 metres. The effects of distance and interference can be neglected.

The Android system sets up its bluetooth radio and accelerometers for the functionality it has deemed most useful for its users, file sharing and game play. Although Android's sensors and bluetooth radio are not suitable for most real-time control applications, quantification of the performance of Android in this paper may prove useful to readers in their own projects.

References

[1] [Online], "Developer Economics Q3 2013 analyst report", http://www.visionmobile.com/DevEcon3Q13, accessed December 2014

[2] [Online], "Android API", http://developer.android.com/guide/topics/sensors/sensors_overview.html, accessed December 2

[3] Andersson M., Bluetooth For Industry, The Industrial Ethernet Book, 11 (September 2002), pp. 5-11.

[4] [Online], "The Bluetooth Special Interest Group," http://www.bluetooth.com, accessed November 3013

[5] [Online], Garcia Pique, J., Lozano Almazan, I., Sanchez Garcia, D., web.udl.es/usuaris/carlesm/docencia/xc1/Treballs/Bluetooth.Treball.pdf, accessed March 2014

[6] Pudaruth, S., Ramdolin, H.K., Bissoonee, A., "An assessment of the performance of bluetooth as a broadcasting channel", Proceedings of the World Congress on Engineering 2010 Vol IWCE 2010, June 30 - July 2, 2010, London, U.K.

[7] Rashid, R.A., Yusoff, R., "Bluetooth performance analysis in personal area network", Proceedings of the 2006 International RF and Microwave Conference, September 12 - 14, 2006, Putrajaya, Malaysia

[8] Bartolomeu, P., Fonseca, J.A., Duarte, P., Rodrigues, P.M., Girao, L.M., "MIDI over Bluetooth", Proceedings of the Conference on Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE, Volume: 1