TESTING AND DEPLOYMENT OF INTELLIGENT OBJECT FRAMEWORK

Sasa Savicand Hao Shi

College of Engineering and Science, Victoria University, Melbourne, Australia

ABSTRACT

Intelligent Object Framework (IOF) is a platform independent framework allowing different devices to be controlled, monitored and communicated. The Core Framework is developed using Microsoft Visual Studio, Microsoft's .NET Framework and Microsoft's Windows Mobile SDK. Secondary Intelligent Object is developed using Tibbo Integrated Development Environment (TIDE) and T-BASIC programming language that is loaded on an EM1026 Embedded Device Platform running Tibbo Operating System (TiOS). The backend is using Microsoft's SQL Server.The successful implementation of the IOF reliesheavily on the testing and deployment which allows seamless monitoring and control of two intelligent objects without any communication glitch between HTC Windows Mobile enabled device and Tibbo EM1026 embedded module. In this paper, embedded systems and Bluetooth applications are first reviewed. The integration testing including First Time Access, Function Exchange, and Function Execution is presented and then followed by deployment of the IOF and the associated components. Finally conclusions are drawn.

KEYWORDS

Intelligent Object Framework, Embedded Systems, Bluetooth and Wireless Network.

1. INTRODUCTION

"A mass in movement resists change of direction. So does the world oppose a new idea? It takes time to make up the minds to its value and importance. Ignorance, prejudice and inertia of the old retard its early progress. It is discredited by insincere exponents and selfish exploiters. It is attacked and condemned by its enemies. Eventually, though, all barriers are thrown down, and it spreads like fire. This will also prove true of the wireless art."[1]. Those were the words spoken over one hundred years ago by Nikola Tesla, the father of modern day Alternating Current (AC) and the first person to patent radio technology [2].

Human society has come a long way since then and is continuing to spiral towards the never ending technological advancements. With the introduction of the Internet in 1990's [3] to the general public, the world of communication has spread its wings far and wide to the pinnacles of the world creating a fast spreading web of information and data. To accompany the growing data exchange and communication frenzy, there has been exponential growth in communication devices, data transfer standards and software technologies and methodologies. Each one contributing significantly to the other in order to work in perfect harmony in creating some of the world's most marvellous communication mediums such as the very internet we use every day. Email communications, data storage and retrieval, device control and monitoring, remote assistance and many more. The sophisticated hardware devices that managed data transfers would not be what they are without the underlying programs which ran them deep within their core.

Taking things even further into lower levels of their infrastructure, Embedded Systems play an equally important role.

In this paper, the core embedded technologies and their underlying management processes are reviewed. Current development environments of the IOF that helped in shaping managed applications facilitate developers and systems designers come up with new innovative, fast and reliable methodologies and applications that are being used [4, 5]. The integration testing including First Time Access, Function Exchange, and Function Execution are covered. The deployment of the IOF and the associated components are presented with a hopeful endeavour on how to utilize the functionality to accomplish some basic control structures via a protocol standard through a wireless medium [6].

2. EMBEDDED SYSTEMS

The earliest embedded systems were making use of microprocessor development specifically for those kinds of applications and they were typically known as microcontrollers. Each new product required a costly and labour-intensive development phase [7]. One of the most surprising developments of the last few decades has been the ascendance of computers in our homes, yet many of these computers and their counterpart smaller components and systems are not recognized as such by their users. The first such system was introduced by Intel, the world's first single chip microprocessor in 1971 [8]. Figure 1 shows the chip's physical layout and the internal structure.

An embedded system is a combination of hardware and software components which are designed to carry out dedicated functions in order to achieve an output of some sort. A good example of a device housing an embedded system would be a Wi-Fi router or a mobile phone. Effectively, a device such as a mobile phone or a Wi-Fi router has a number of embedded modules, each one programmed specifically to handle the type of input/output designated. One embedded module might be specific to a keypad, whilst another is responsible for the LCD display of the mobile phone. Wi-Fi router wouldn't have many embedded units within the structure for the reason that it is very specific in its set of operations. Modulated signals are sent and received over a wireless medium with a couple of helper processors carryingthe encoding and the decoding process of the data.

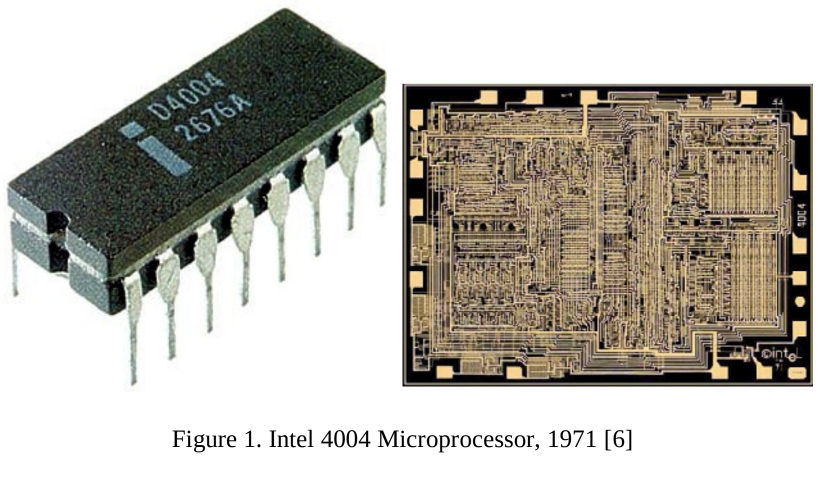


Figure 1. Intel 4004 Microprocessor, 1971 [6]

As the years progressed, the designs were becoming more and more complex but utilizing a lot of features within their processors which were getting smaller with each new design implementation. This paved a way of unimaginable possibilities of implementation in the real world providing people with the ability to become their own masters of design. The details of the Embedded Systems Development are summarized in Table 1.

Table 1: Timeline of Embedded Systems Design[11]

Year	Event
1936 – 1937	Alan Turin publishes article "On Computable Numbers, with an Application to Entscheindungsproblem", which paves the way not only for computers but for stored-program architecture.
1939	John Atanasoff and Clifford Berry create a prototype of the Atanasoff-Berry Computer (ABC), the first digital computer.
1944	Colossus, the first fully functioning electric digital computer, used by the Bletchley Park cryptanalysis team.
1946	Eniac, the world's first "programmable" stored program electronic computer comes to life. It housed 18,000 vacuum tubes and it roughly had the complexity of today's cheap digital watch.
1962	Texas Instruments introduces the 7400 series of logic IC's
1965	DEC introduces the 12bit PDP-8 minicomputer, which by 1973 was the best-selling computer in the world at a staggering cost of $18k in 1965 dollars, or $120k today
1970	Intel releases the 1103 chip, the first generally available dynamic random-access memory (DRAM) chip.
1971	Intel releases first microprocessor, the 4004, a 4bit central processing unit.
1971	Intel releases the 1101 chip, a 256bit programmable memory, and the 1701 chip, a 256byte erasable read-only memory (EROM)
1972	Intel introduces the 8008 microprocessor
1973	Intel releases the Intellec 8 development system for the 8008 processor
1973	Texas Instruments introduces the first dynamic random-access memory (DRAM) that is a 4k-bit chip
1975	MITS releases the ATLAIR 8800 at an unheard-of price of $400. The home computer was born.
1978	Intel introduces the 8086 microprocessor chip. The beginning of x86 architecture
1989	Intel releases the i486, 32-bit x86 microprocessor.
1993	Intel releases Pentium processor
2002	Intel Itenium 2 is released
2005	IBM, Intel and AMD release their first multicore processors

Embedded systems have become an essential part of daily lives enabling us to undertake our daily duties in a much simpler and quicker way. Whilst a lot of people accept these technological advancements as a positive drive for human inspiration and exploration, nevertheless it did impact our natural way of thinking. Whilst more and more work is done by automated processes, human involvement in manual labour has drastically decreased. Islam Hassan El-Adaway says "one can trace that as a man reaches more and more progress, he loses more and more of his good instinctive human nature" [11], however according to Jacob Bronowski, "Man is unique not because he does science, and he is unique not because he does art, but because science and art are expressions of his marvellous plasticity of mind". So, in respect to the great minds of the 20th and 21st century, we continue to evolve our systems based on their legacy, making them more robust and efficient in order to prove ourselves that we are more than capable of taking them to the next level.

Embedded systems have come at such a low cost, that just about anyone with basic electronic and programming knowledge can design an efficient system. Progression of the technology of Embedded Systems has also enabled people to easily scope out, design, build and deploy their prototypes or full scale products ready for the commercial world. The advantage of Embedded Systems is the integration capability with other embedded Devices. This suggests that design may integrate many numbers of other embedded components allowing thedevice

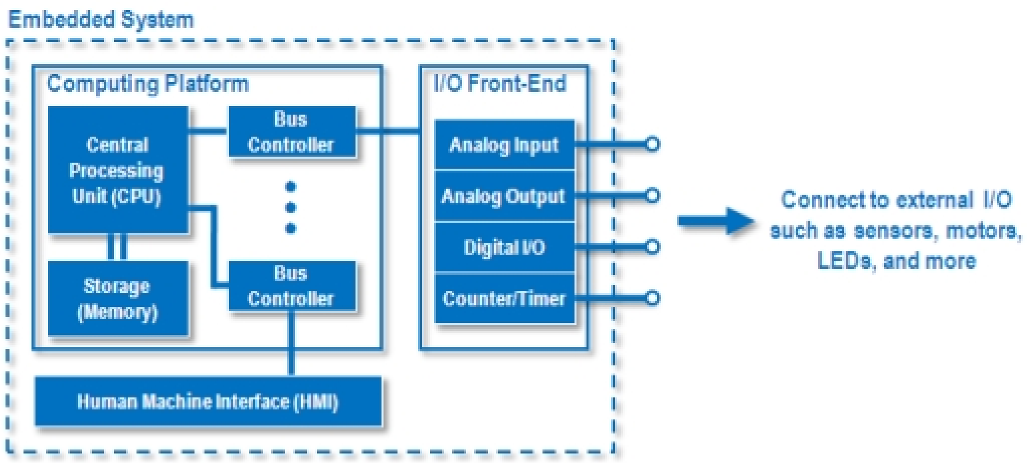


Figure 2. Simple Embedded System [9]

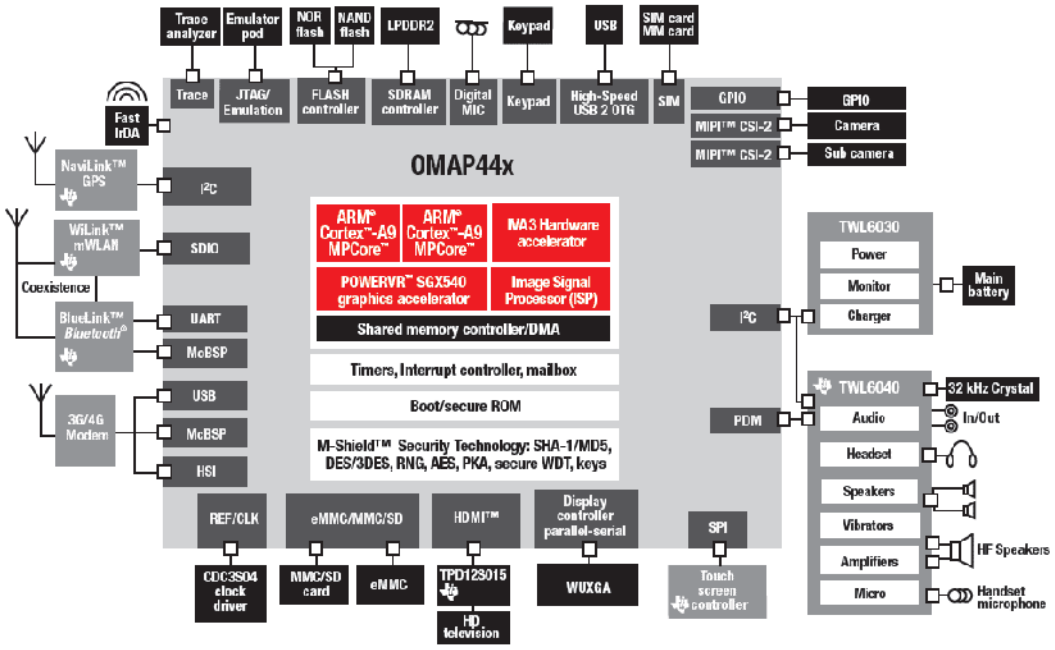


Figure 3. Texas Instruments OMAP Embedded Controller [13]

a wide range of functionalities. Robotics is a perfect blend of integration factors when designing such complex pieces of equipment (this includes the complex industrial robots in addition to the popular Honda's ASIMO robot). They encapsulate a wide number of different components such as:

- Motion detection sensors
- Ultra-sonic collision detection sensors
- Camera controllers
- Servo motor controllers – movement of the limbs or other parts
- Pressure sensors

Each of these components, depending on the nature of their complexity, has an embedded module talking to one another and communicating directly or through a series of other embedded components that act as a central intelligence of communication. The basic layout of an embedded system is shown in Figure 2.A more complex diagram is outlined in Figure 3 where Texas Instruments have introduced a new Open Multimedia Application Platform (OMAP) 4 used for mobile applications. It is to target smart-phones and Mobile Internet Devices (MID) and it introduces a number of benefits over other processors [12]. It features three hardware accelerators to support 1080p, Dual-core ARM Symmetric Multiprocessors and graphics accelerators. With a 1 GHz+ processor, this means that a mini super-computer is in a pocket. The sheer complexity of the design can clearly be seen. Essentially the basic components of any embedded systems are the Central Processing Unit (CPU), BUS Controllers, some form of interfaces and an I/O front end which is the bridge to external I/O devices such as the ones mentioned previously.

3. BLUETOOTH APPLICATIONS

Bluetooth RF (physical layer) operates in unlicensed ISM band at 2.4 GHz. The symbol rate is 1 mega-symbol per second, supporting bit rates of 1 Megabit per second or with Enhanced Data Rate, a gross air bit rate of 2 or 3Mb/s. There are two modes, Basic Rate and Enhanced Data Rate (EDR). Bluetooth technology has the ability to link digital devices within a range of 10m, which can be expandable to 100m by increasing the transmission power. The most basic overview of Bluetooth can be divided in two areas: Radio and Link & Channel Management Protocols.

The Radio Channel, during a typical operation, is shared by group of devices that are synchronized to a common clock and frequency hopping pattern. A secondary component of the operation of Bluetooth is the Piconet, and is described as an ad-hoc computer network linking a user group of devices. Bluetooth technology utilizes this network allowing master to slave communication of 7 devices that can be extended to 255. One device provides the synchronization reference and is referred to as the master. The other devices are classified as slaves. Devices in Piconet use a distinctive frequency hopping pattern (FHP) which is algorithmically determined [14].

Link and Channel Management Protocols consist of control layers that sit above the physical channel. The hierarchy consists of channels and links from the physical channel upwards, physical link, logical transport, and logical link and L2CAP channel. Within a physical channel, physical links are formed between two devices that transmit packets in either direction. In a Piconet physical channel, there are restrictions to which devices are allowed to form a physical link. The physical links are only allowed between slave and master, whereas a physical link is not directed between the slaves.

The logical link is used as a transport for one or more logical links that support uni-cast synchronous, asynchronous and isochronous traffic. The link is multiplexed onto the physical layer by occupying slots assigned by a scheduling function and for resource management of communication. Link Manager Protocol (LMP) is used to control the operation of devices in the Piconet, thus providing services to manage the lower levels of architectural layers (radio and baseband). Finally above all layers is the L2CAP layer which provides a channel-based abstraction to applications and services.

Bluetooth home based automation systems have been used wildly within homes and cars over the last decade. The proposed system [15] is an adoption of HAP (House Automation Protocol) which works on the Bluetooth communication by establishing a PAN (Personal Area Network) of devices through utilization of Piconet. The system proposes a network which contains a remote, mobile host controller and several client modules, acting as home appliances. The client modules communicate with the host controller through Bluetooth devices. The HAP protocol is constructed above the Bluetooth software stack and it follows the layer model proposed by Bluetooth Special interest Group (SIG). The HAP protocol is described by the home automation descriptor table, which explains the HAP protocol stack. In order to show the proposed system in action, significant software development need to be fulfilled in order to provide the custom firmware for the microcontroller which was driving a room temperature sensor circuitry. The attempt at home automation systems is a successful one but fails to address the issue of complexity in circuitry design and development. The Intelligent Object Framework attempts to hide the complexity from the user by utilization of an adaptive software based protocol stack that can be implemented on a platform that does not need to conform to a specific architectural limitations or design. This can be an ad-hoc network or a specified Wi-Fi, 3G or Ethernet. Limitations with running Piconet PAN's are clearly shown in their lack of direct communication with slave to slave. The multiplexing of data and communication is done via a master controller. The limitations on communication can highly depend on the initial designs of host controllers. Intelligent Object Framework mitigates the limitation by providing a protocol that is scalable across multiple networks and devices. This can further be extended to include device to device communication, as protocol runs on IP.

4. SYSTEM OVERVIEW

Intelligent Object Framework is a system of integrated set of services and components which leverage across a scalable network. The complete system provides the user with ability to control devices. A typical scenario of the system is an Intelligent Object enabled device such as an air conditioner. Once the air conditioner is set up, the user discovers IOF Services by initializing a discovery process. Once the protocol is set up and connection established, the IOF Services request device functions. The functions received are stored away in the database, and can be retrieved at any time. This enables quick and easy device access from anywhere in the world provided that the services are visible to the internet. The Intelligent Object Framework specification defines the protocol standard for function exchange. By following the specification, the functionality can be abstracted from the user, and integrated on any platform. The different spectrum of devices that can utilize the framework can range from house hold appliances such as TV's, fridges, air conditioning and heating system to portable device range such as mobile phones and PDA's. The implementation can also be adapted by new breed of Intelligent Cars that are Wi-Fi enabled, with programmable units.

5. INTEGRATION TESTING

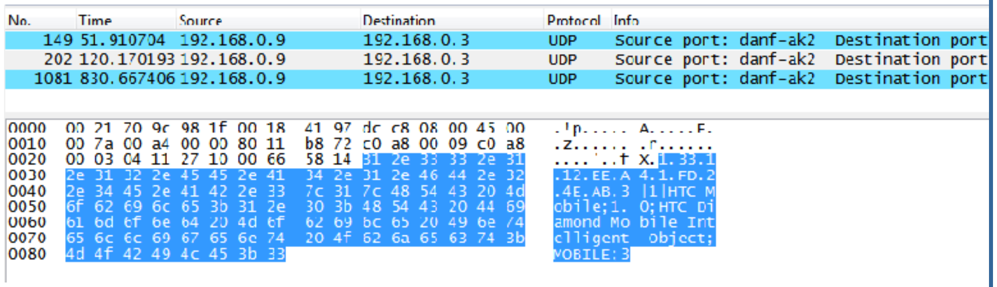
The framework provides a client application (IOF Management Studio) that uses its services (IOF Service and Protocol Layer) allowing device control and monitoring over Wi-Fi medium [4, 5].

The Data Layer allowed fast device function storage and on demand retrieval [6]. The testing is focused on First Time Access, Function Exchange, and Function Execution.

Table 1.First Time Access

Test Case #:	TC000001-1
Test Condition:	Device and services must exist and be part of a network. Device must be accepted by the services in order to precede with test
Procedures:	- Enter services IP address - Connect to services
Expected Results/Objectives:	Device connection attempt popup
Actual Results:	Command executed
Test Status:	**Pass:** ✓ **Fail:**
Comments	

Intelligent Object sends the first device descriptor packet to the services. Once the services capture the packet and present it to the user, the user’s initiative is to accept or reject.



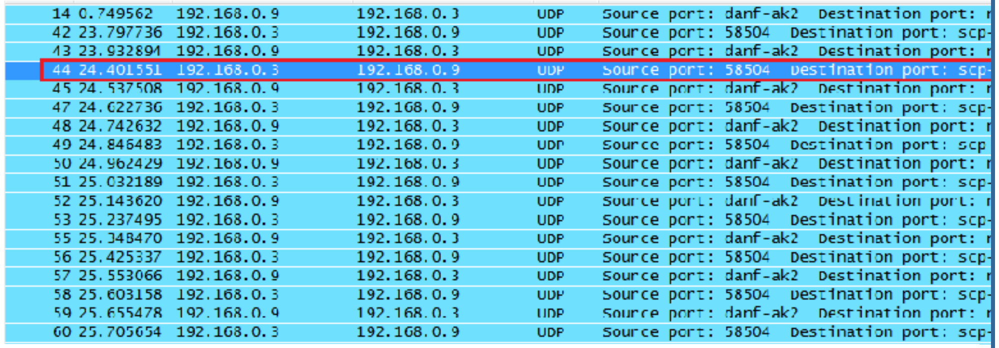
```
No.    Time       Source        Destination   Protocol Info
   149 51.910704  192.168.0.9   192.168.0.3   UDP      Source port: danf-ak2  Destination port
   202 120.170193 192.168.0.9   192.168.0.3   UDP      Source port: danf-ak2  Destination port
  1081 830.667406 192.168.0.9   192.168.0.3   UDP      Source port: danf-ak2  Destination port

0000  00 21 70 9c 98 1f 00 18  41 97 dc c8 08 00 45 00   .!p..... A.....E.
0010  00 7a C0 a4 00 00 80 11  b8 72 c0 a8 00 09 c0 a8   .z...... .r......
0020  00 03 C4 11 27 10 00 66  58 14 31 2e 33 33 2e 31   ....'..f X.1.33.1
0030  2e 31 32 2e 45 45 2e 41  34 2e 31 2e 46 44 2e 32   .12.EE.A 4.1.FD.2
0040  2e 34 45 2e 41 42 2e 33  7c 31 7c 48 54 43 20 4d   .4E.AB.3 |1|HTC M
0050  6f 62 69 6c 65 3b 31 2e  30 3b 48 54 43 20 44 69   obile;1. 0;HTC Di
0060  61 6d 6f 6e 64 20 4d 6f  62 69 6c 65 20 49 6e 74   amond Mo bile Int
0070  65 6c 6c 69 67 65 6e 74  20 4f 62 6a 65 63 74 3b   elligent  Object;
0080  4d 4f 42 49 4c 45 3b 33                            MOBILE;3
```

Table 2. Function Exchange

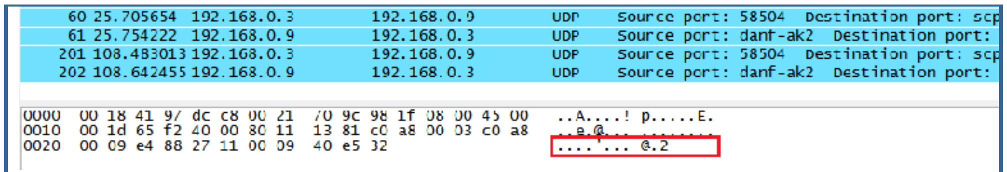
Test Case #:	TC000001-2			
Test Condition:	Authenticated() = true, DeviceConnected() = true			
Procedures:	Accept device connection request			
Expected Results/Objectives:	Device function successfully loaded to Intelligent Object Framework Management Studio			
Actual Results:	Commands added and initiated			
Test Status:	**Pass:**	✓	**Fail:**	

Comments

Packet exchange illustrated in the bellow UDP stream trace indicates the function exchange between services and intelligent object. The exchange is initialized by FUNC_DATA_REQ flag set. In the below byte stream trace, hex 32 (decimal 2) is streamed to the device.

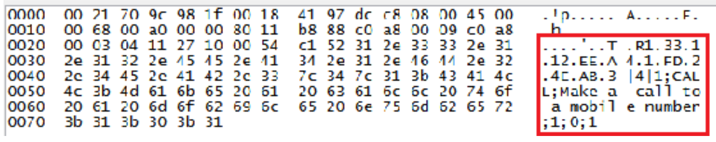


```
14 0.749562   192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
42 23.797736  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp-
43 23.932894  192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
44 24.401551  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp-
45 24.537508  192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
47 24.622736  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp-
48 24.742632  192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
49 24.846483  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp
50 24.962429  192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
51 25.032189  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp-
52 25.143620  192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
53 25.237495  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp-
55 25.348470  192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
56 25.425337  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp-
57 25.553066  192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
58 25.603158  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp-
59 25.655478  192.168.0.9        192.168.0.3        UDP     Source port: danf-ak2  Destination port: r
60 25.705654  192.168.0.3        192.168.0.9        UDP     Source port: 58504  Destination port: scp-
```



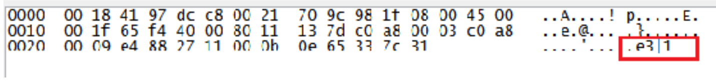
```
     60 25.705654  192.168.0.3        192.168.0.9      UDP    Source port: 58504   Destination port: scp
     61 25.754222  192.168.0.9        192.168.0.3      UDP    Source port: danf-ak2   Destination port:
    201 108.483013 192.168.0.3        192.168.0.9      UDP    Source port: 58504   Destination port: scp
    202 108.642455 192.168.0.9        192.168.0.3      UDP    Source port: danf-ak2   Destination port:

0000   00 18 41 97 dc c8 00 21  70 9c 98 1f 08 00 45 00   ..A....! p.....E.
0010   00 1d 65 f2 40 00 80 11  13 81 c0 a8 00 03 c0 a8   ..e.@... ........
0020   00 09 e4 88 27 11 00 09  40 e5 32                  ....'... @.2
```

The intelligent object device responds by sending the first function to the services

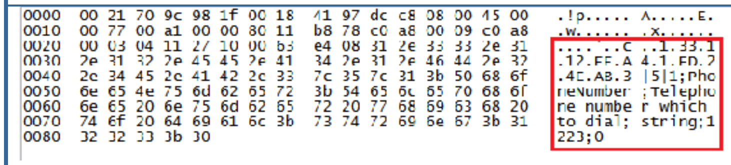


```
0000   00 21 70 9c 98 1f 00 18  41 97 dc c8 08 00 45 00   .!p..... A.....E.
0010   00 68 00 a0 00 00 80 11  b8 88 c0 a8 00 09 c0 a8   .h......
0020   00 03 04 11 27 10 00 54  c1 52 31 2e 33 33 2e 31   ....'..T .R1.33.1
0030   2e 31 32 2e 45 45 2e 41  34 2e 31 2e 46 44 2e 32   .12.EE.A 4.1.FD.2
0040   2c 34 45 2c 41 42 2c 33  7c 34 7c 31 3b 43 41 4c   .4E.AB.3 |4|1;CAL
0050   4c 3b 4d 61 6b 65 20 61  20 63 61 6c 6c 20 74 6f   L;Make a  call to
0060   20 61 20 6d 6f 62 69 6c  65 20 6e 75 6d 62 65 72    a mobil e number
0070   3b 31 3b 30 3b 31                                  ;1;0;1
```

The services respond with an acknowledgement and request for parameters for the corresponding function

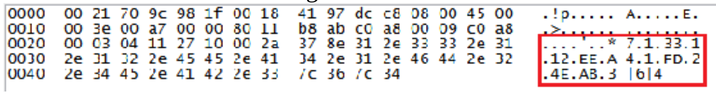


```
0000   00 18 41 97 dc c8 00 21  70 9c 98 1f 08 00 45 00   ..A....! p.....E.
0010   00 1f 65 f4 40 00 80 11  13 7d c0 a8 00 03 c0 a8   ..e.@... .}......
0020   00 09 e4 88 27 11 00 0b  0e 65 33 7c 31            ....'... .e3|1
```



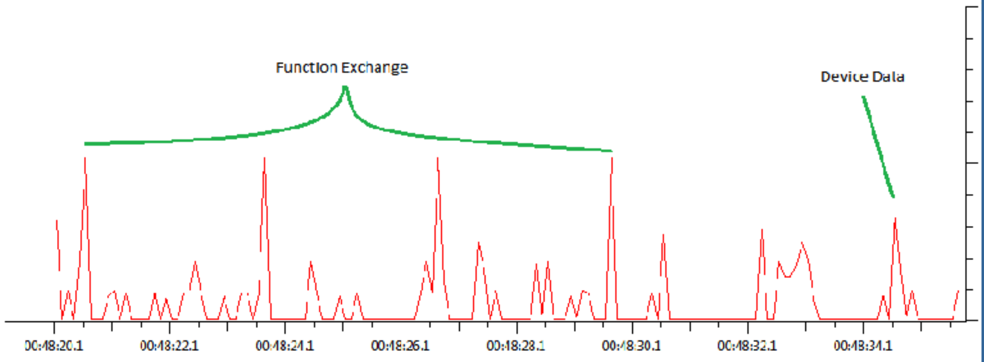
```
0000   00 21 70 9c 98 1f 00 18  41 97 dc c8 08 00 45 00   .!p..... A.....E.
0010   00 77 00 a1 00 00 80 11  b8 78 c0 a8 00 09 c0 a8   .w...... .x......
0020   00 03 04 11 27 10 00 b3  e4 08 31 2e 33 33 2e 31   ....'..C ..1.33.1
0030   2e 31 32 2e 45 45 2e 41  34 2e 31 2e 46 44 2e 32   .12.EE.A 4.1.FD.2
0040   2c 34 45 2c 41 42 2c 33  7c 35 7c 31 3b 50 68 6f   .4E.AB.3 |5|1;Pho
0050   6e 65 4e 75 6d 62 65 72  3b 54 65 6c 65 70 68 6f   neNumber ;Telepho
0060   6e 65 20 6e 75 6d 62 65  72 20 77 68 69 63 68 20   ne numbe r which
0070   74 6f 20 64 69 61 6c 3b  73 74 72 69 6e 67 3b 31   to dial; string;1
0080   32 32 33 3b 30                                     223;0
```

The process is finally completed once all function and parameter data has been exchanged with the device. The final status flag FIN is set and the device is added to the device collection.



```
0000   00 21 70 9c 98 1f 00 18  41 97 dc c8 08 00 45 00   .!p..... A.....E.
0010   00 3e 00 a7 00 00 80 11  b8 ab c0 a8 00 09 c0 a8   .>...... ........
0020   00 03 04 11 27 10 00 2a  37 8e 31 2e 33 33 2e 31   ....'..* 7.1.33.1
0030   2e 31 32 2e 45 45 2e 41  34 2e 31 2e 46 44 2e 32   .12.EE.A 4.1.FD.2
0040   2e 34 45 2e 41 42 2e 33  7c 36 7c 34               .4E.AB.3 |6|4
```

The stated byte trace correlates to the below graph by depicting transmission of each function and related parameters.



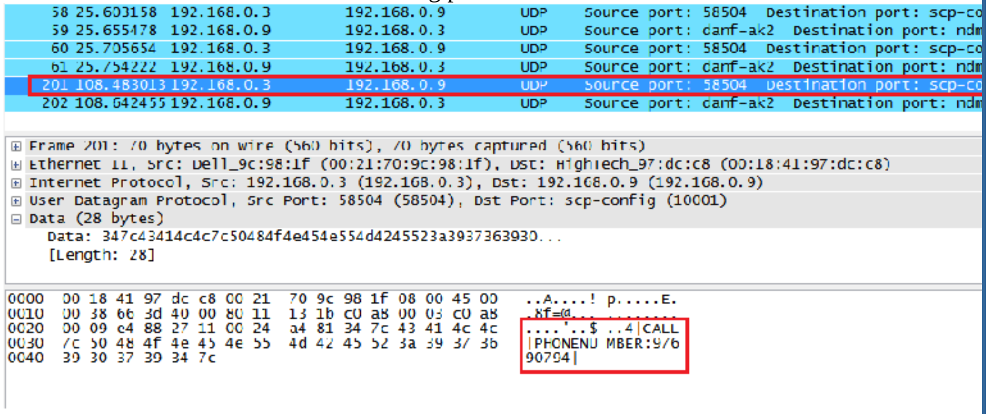
This proves the test case of function exchange

Table 3. Function Execution

Test Case #:	TC000001-3
Test Condition:	LoggedOn() = true, DeviceActive() = true, PhoneNumber = 9769xxxx
Procedures:	- Navigate to **Functions** - Select the appropriate command to execute - Define function parameters (if any) - Click on the **Execute** button
Expected Results/Objectives:	Device response with a status of successful command execution Status expected: IOF_STS_0001 – COMMAND OK Objective is a successful command execution

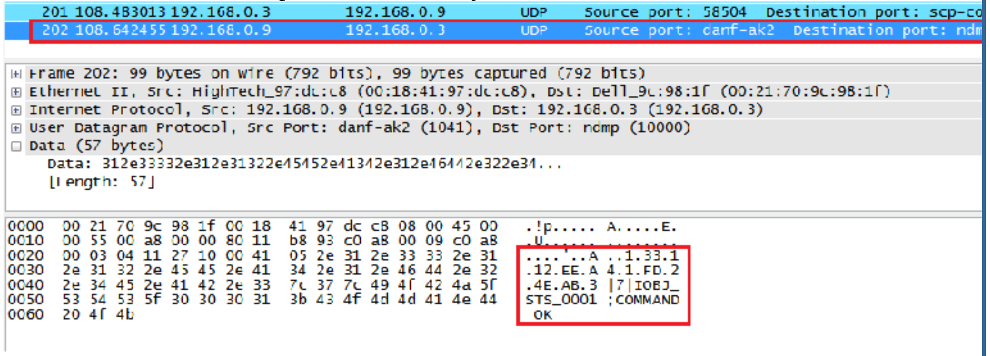
Actual Results:	Command executed		
Test Status:	**Pass:**	✓	**Fail:**
Comments			

Following the UDP stream, the information exhibits data exchange. The below packet bytes show the **CALL** command with the trailing parameter.



```
 58 25.603158  192.168.0.3      192.168.0.9      UDP    Source port: 58504   Destination port: scp-co
 59 25.655478  192.168.0.9      192.168.0.3      UDP    Source port: canf-ak2   Destination port: ndm
 60 25.705654  192.168.0.3      192.168.0.9      UDP    Source port: 58504   Destination port: scp-co
 61 25.754222  192.168.0.9      192.168.0.3      UDP    Source port: canf-ak2   Destination port: ndm
201 108.483013 192.168.0.3      192.168.0.9      UDP    Source port: 58504   Destination port: scp-co
202 108.642455 192.168.0.9      192.168.0.3      UDP    Source port: canf-ak2   Destination port: ndm

⊞ Frame 201: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
⊞ Ethernet II, Src: Dell_9c:98:1f (00:21:70:9c:98:1f), Dst: HighTech_97:dc:c8 (00:18:41:97:dc:c8)
⊞ Internet Protocol, Src: 192.168.0.3 (192.168.0.3), Dst: 192.168.0.9 (192.168.0.9)
⊞ User Datagram Protocol, Src Port: 58504 (58504), Dst Port: scp-config (10001)
⊟ Data (28 bytes)
    Data: 347c43414c4c7c50484f4e454e554d4245523a3937363930...
    [Length: 28]

0000  00 18 41 97 dc c8 00 21  70 9c 98 1f 08 00 45 00   ..A....! p.....E.
0010  00 38 66 3d 40 00 80 11  13 1b c0 a8 00 03 c0 a8   .8f=@... ........
0020  00 09 e4 88 27 11 00 24  a4 81 34 7c 43 41 4c 4c   ....'..$ ..4|CALL
0030  7c 50 48 4f 4e 45 4e 55  4d 42 45 52 3a 39 37 36   |PHONENU MBER:976
0040  39 30 37 39 34 7c                                  90794|
```

Furthermore, the device response can be clearly seen as the command execution is successful.



```
201 108.483013 192.168.0.3      192.168.0.9      UDP    Source port: 58504   Destination port: scp-co
202 108.642455 192.168.0.9      192.168.0.3      UDP    Source port: canf-ak2   Destination port: ndm

⊞ Frame 202: 99 bytes on wire (792 bits), 99 bytes captured (792 bits)
⊞ Ethernet II, Src: HighTech_97:dc:c8 (00:18:41:97:dc:c8), Dst: Dell_9c:98:1f (00:21:70:9c:98:1f)
⊞ Internet Protocol, Src: 192.168.0.9 (192.168.0.9), Dst: 192.168.0.3 (192.168.0.3)
⊞ User Datagram Protocol, Src Port: danf-ak2 (1041), Dst Port: ndmp (10000)
□ Data (57 bytes)
    Data: 312e33332e312e31322e45452e41342e312e46442e322e34...
    [Length: 57]

0000  00 21 70 9c 98 1f 00 18  41 97 dc c8 08 00 45 00   .!p..... A.....E.
0010  00 55 00 a8 00 00 80 11  b8 93 c0 a8 00 09 c0 a8   .U...... ........
0020  00 03 04 11 27 10 00 41  05 2e 31 2e 33 33 2e 31   ....'..A ..1.33.1
0030  2e 31 32 2e 45 45 2e 41  34 2e 31 2e 46 44 2e 32   .12.EE.A 4.1.FD.2
0040  2e 34 45 2e 41 42 2e 33  7c 37 7c 49 4f 42 4a 5f   .4E.AB.3 |7|IOBJ_
0050  53 54 53 5f 30 30 30 31  3b 43 4f 4d 4d 41 4e 44   STS_0001 ;COMMAND
0060  20 4f 4b                                            OK
```

6. DEPLOYMENT

The deployment of Intelligent Object Framework with the additional HTC Mobile (Intelligent Object) and Tibbo Embedded Device (Intelligent Object 2) proved to be a success.The integration of Intelligent Object Framework consists of two primary steps in allowing full communication between the intelligent object and the intelligent object framework services, and integration of internal IOF components. First step states that the devices and the framework must be part of a network. The network endpoints, being the devices and the framework, must be able to communicate with each other. The second step that is required to be completed is the installation of Microsoft's SQL Server. As discussed earlier, the database keeps intelligent object data, thus the data is being retrieved on demand from the user. Application components are integrated at the beginning of the design of the internal framework infrastructure. The deployment of Intelligent Object Framework and the associated components can be summarized in Figure 4.

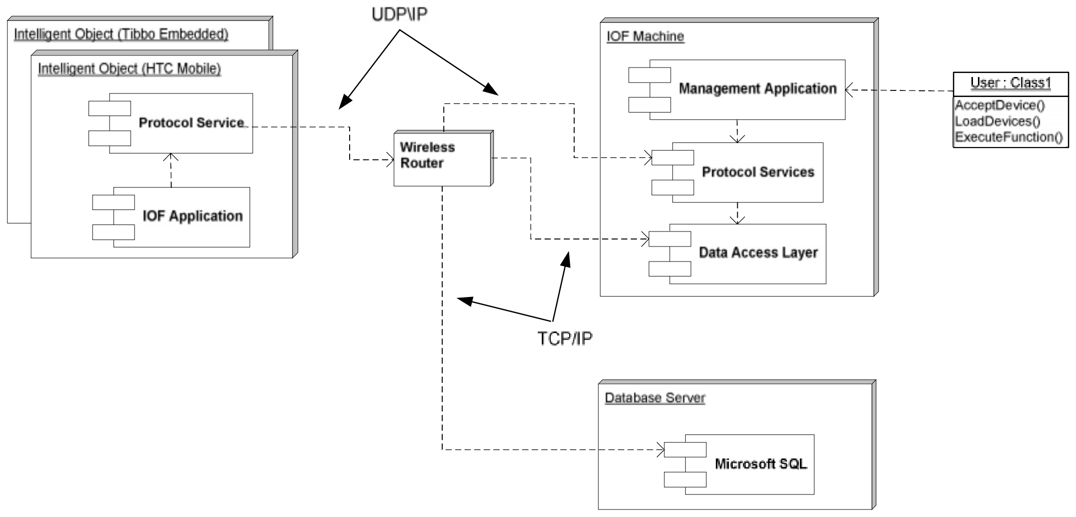


Figure4.IOF Deployment Diagram

7. CONCLUSIONS

It can be concluded that device control and management have become an essential functional addition to many devices ranging from remotely operated apparatuses, industrial equipment and network related devices. The ability to monitor, assess, fault rectify and manage devices wirelessly or over the Ethernet proves to be a fast, effective and cost reductive process, as companies and individuals no longer need to put up with resource constraints and budget for off-site support. The down side of having an infrastructure of different devices is managing and controlling individual components by using a range of different applications through different communication protocols. Each vendor has a set of protocols that applications need to conform to in order to satisfy production requirements. This also adds to the complexity of having to apply multiple firewall rules to allow application and device communication, increasing the threat to security of the internal network.

Whilst still in the primitive stage of development, the IOF shows another way of device control and management over the Ethernet or wireless medium, by implementation of the framework and its set of components which provide a bilateral communication protocol for multiple platform independent device monitoring and control. The primary requirement, for a successful completion of the proposed project, states that the IOF application must provide an interface for controlling two or more devices of different platforms.

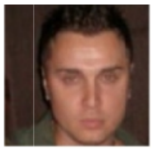
ACKNOWLEDGEMENTS

Sasa Savic would like to thank Victoria University for offering the Research Training Scheme (RTS) for his PhD study.

REFERENCES

[1] Massie W. W. and Underhill C.R. (1908). “The Future of the Wireless Art”, Wireless Telegraphy & Telephony: pp 67-71.
[2] Nikola Tesla (2007) “The Nikola Tesla Treasury”, Wilder Publications, 1st Edition, 8 August 2007.
[3] Bellis, M. "An outstanding scientist, Nikola Tesla paved the way for modern technology", http://inventors.about.com/od/tstartinventions/a/Nikola_Tesla.htm , viewed on 10 November 2013.
[4] Kahn R. E. and Cerf, V. G. (1999) "What Is The Internet (And What Makes It Work)", http://www.cnri.reston.va.us/what_is_internet.html, viewed on 10 November 2013.
[5] Savic S. and Shi, H. (2011), "An Intelligent Object Framework for Smart Living", Procedia Computer Science 5 (2011) 386–393.
[6] Savic, S. (2010) " Intelligent Object Framework", thesis for Master of Science in Computer Science, School of Engineering and Science, Victoria University, June 2010, 138 pages.
[7] Savic S. and Shi, H. (2013), “Detailed Design of Intelligent Object Framework", submitted toInternational Journal of Computer Networks & Communications (IJCNC).
[8] Holburn, D. (1997) "Embedding the PC", viewed on 10 November 2013, http://www2.eng.cam.ac.uk/~dmh/chiprack/chiprack.html#Embedded%20Computer ,
[9] Barr, M. and Massa, A. (2006). Programming Embedded Systems with C and GNU Development Tools, 2nd edition, O'Reilly Media Inc.
[10] Embedded (2008). "Intel Microprocessor", http://www.embedded.com, viewed on 10 November 2013.
[11] Embedded (2010) "Embedded Timeline", http://www.embedded.com/timeline, retrieved on 2 May 2010.
[12] Al-Adaway, I. H. (2004) "Moral Technology" 6(3).
[13] Texas Instruments (2010). "OMAP 4 Mobile Applications Platform.", http://focus.ti.com//general/docs/wtbu/wtbuproductcontent.tsp?templateId=6123&navigationId=12843&contentId=53243#chipDiagram,viewed on 10 November 2013.
[14] Embeded-System.Net (2009). "Open Multimedia Application Platform.", http://embedded-system.net/ti-omap-4-open-multimedia-application-platform-for-mobile-applications.html, viewed on 10 November 2013.
[15] Bluetooth (2010). "Overview of Bluetooth Operation",retrieved on 2 May 2010, http://www.bluetooth.com/English/Technology/Works/Pages/Overview_of_Operation.aspx .
[16] Sriskanthan N., Tan, F. and Karande,A. (2002). "Bluetooth based home automation system." Microprocessors and Microsystems 26(6): 281-289.

Authors



Sasa Savicis a part-time PhD student at Victoria University in Melbourne, Australia. He obtained Advanced Diploma of Computer Systems Engineering in 2002 from Royal Melbourne Institute of Technology (RMIT), Australia. He received Bachelor of Science and Master of Science in Computer Science from Victoria University in 2006and 2010, respectively. He is currently working at Telstra, the largest Australian telecommunications company, as a senior software engineer.

Dr. Hao Shi is an Associate Professor in College of Engineering and Science, Victoria University, Melbourne, Australia. She completed her Bachelor of Engineering degree at Shanghai Jiao Tong University, China and obtained her PhD in the area of Computer Engineering at University of Wollongong. She has been actively engaged in R&D and external consultancy activities. Her research interests include p2p Network, Location-Based Services, Web Services, Robotics Vision, Visual Communications, Internet and Wireless Technologies.