Reference

[1] Cheju halla University, S. H. Moon, C. sick Lee, and Cheju halla University, "Dynamic Management Software Design in Embedded System using Middle," 2014, pp. 186–191.

[2] E. Zeeb, A. Bobek, H. Bohn, and F. Golatowski, "Service-oriented architectures for embedded systems using devices profile for web services," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, 2007, vol. 1, pp. 956–963.

[3] A. Scholz*et al.*, "□ SOA-Service Oriented Architectures adapted for embedded networks," in *2009 7th IEEE International Conference on Industrial Informatics*, 2009, pp. 599–605.

[4] D. E. Culler, J. Hill, P. Buonadonna, R. Szewczyk, and A. Woo, "A network-centric approach to embedded software for tiny devices," in *International Workshop on Embedded Software*, 2001, pp. 114–130.

[5] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-Oriented Cloud Computing Architecture," 2010, pp. 684–689.

[6] G. L. Gopu, K. V. Kavitha, and J. Joy, "Service Oriented Architecture based connectivity of automotive ECUs," in *Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on*, 2016, pp. 1–4.

[7] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Internet of Things (IOT), 2010*, 2010, pp. 1–8.

[8] X. Cai, G. Ouyang, and X. Zhang, "Design of Fan Performance Detection System Based on ARM Embedded System," *Int. J. Smart Home*, vol. 8, no. 1, pp. 311–316, Jan. 2014.

[9] S. H. Moon, "Designing and Embodiment of Software that Creates Middle Ware for Resource Management in Embedded System," *Int. J. Softw. Eng. Its Appl.*, p. 12, 2014.

[10] S. Arulselvi, "Advanced Internet Access System Using Embedded Linux," p. 4, 2014.

[11] P. Pannuto *et al.*, "A networked embedded system platform for the post-mote era," 2014, pp. 354–355.

[12] S. Thilagavathi, J. Sathyapriya, T. M. Minipriya, T. Mohanapriya, and C. Subashini, "Modern Coding Theory Based On Fountain Code Implementation In Embedded System," vol. 3, no. 2, p. 4, 2014.

[13] R. K. Tiwari and S. K. Agrahari, "Arduino Compatible World Wide Web Controlled Embedded System," vol. 3, no. 9, p. 4, 2014.

[14] Cheju halla University, S. H. Moon, C. sick Lee, and Cheju halla University, "Dynamic Management Software Design in Embedded System using Middle," 2014, pp. 186–191.

[15] PG. Student of, Sinhagad college of engineering,Pune, P. Kumbhar, and D. S. . Lokhande, "Embedded Based Compact Fuzzy System to Speed Control of Single Phase Induction Motor," *IOSR J. Electr. Electron. Eng.*, vol. 9, no. 5, pp. 56–59, 2014.

[16] S. Salgaonkar, M. J. D. Bhosale, and M. P. Bangde, "Embedded Web Server based on ARM Cortex for DAC System," vol. 5, no. 7, p. 6, 2014.

[17] J. K. Arthur, T. Robinson, and R. Latha, “Implementation aspects of Bio-Metric system in Electronic Voting Machine by using embedded security and big data approach,” vol. 1, no. 3, p. 6.
[18] P. G. Pachpande, “Internet Based Embedded Data Acquisition System,” vol. 5, p. 4, 2014.
[19] S. V. Shinde and M. P. Zaware, “Wi-Fi based Monitoring and Controlling of Embedded System,” vol. 5, p. 7, 2014.
[20] “International Journal of Combined Research & Development (IJCRD) eISSN:2321-225X; pISSN:2321-2241 Volume: 2; Issue: 5; May -2014,” vol. 2, no. 5, p. 4.

Appendix A -Acronyms

Endpoint - This is an URL where a service can be accessed by a client application

Peripheral device -This is generally defined as any auxiliary device such as a computer mouse or keyboard that connects to the computer

EmSOS - Enhanced **S**ervice **O**riented **S**oftware Framework for **Em**bedded Android

API - Application Programming Interface

jUnit – A unit test framework for java programming language

OWASP- Open Web Application Security Project

IDE – Integrated Development Environment

REST – This is an architectural style for designing distributed systems

Appendix B -Test results and source codes

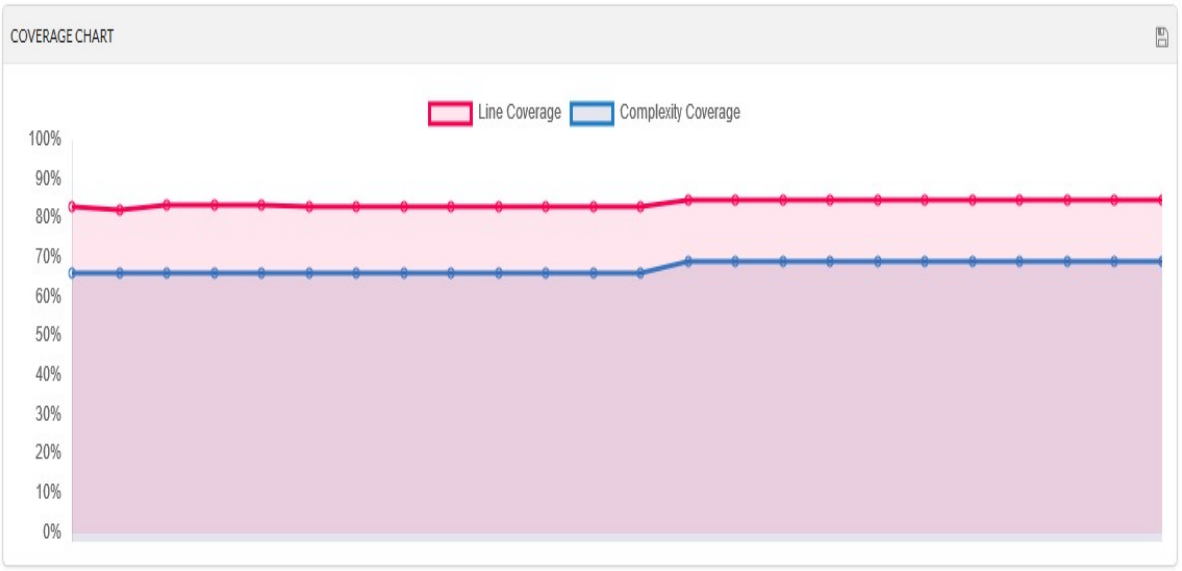

Figure 4 Coverage chart with the development commits which has done with version controlling tool. (Git)

```
publicclassSysInfTest {
     */
publicstaticvoid main(String[] args) {

System.setProperty(org.slf4j.impl.SimpleLogger.DEFAULT_LOG_LEVEL_KEY,
"INFO");
System.setProperty(org.slf4j.impl.SimpleLogger.LOG_FILE_KEY,
"System.err");
        Logger LOG = LoggerFactory.getLogger(SystemInfoTest.class);

LOG.info("Initializing System...");
SystemInfosi = newSystemInfo();

HardwareAbstractionLayerhal = si.getHardware();
OperatingSystemos = si.getOperatingSystem();

System.out.println(os);

LOG.info("Checking computer system...");
printComputerSystem(hal.getComputerSystem());

LOG.info("Checking Processor...");
printProcessor(hal.getProcessor());

LOG.info("Checking Memory...");
printMemory(hal.getMemory());

LOG.info("Checking CPU...");
printCpu(hal.getProcessor());

LOG.info("Checking Processes...");
printProcesses(os, hal.getMemory());

LOG.info("Checking Sensors...");
printSensors(hal.getSensors());

LOG.info("Checking Power sources...");
printPowerSources(hal.getPowerSources());

LOG.info("Checking Disks...");
printDisks(hal.getDiskStores());

LOG.info("Checking File System...");
printFileSystem(os.getFileSystem());

LOG.info("Checking Network interfaces...");
printNetworkInterfaces(hal.getNetworkIFs());
```

Figure 9 Test class of System test in test driven development

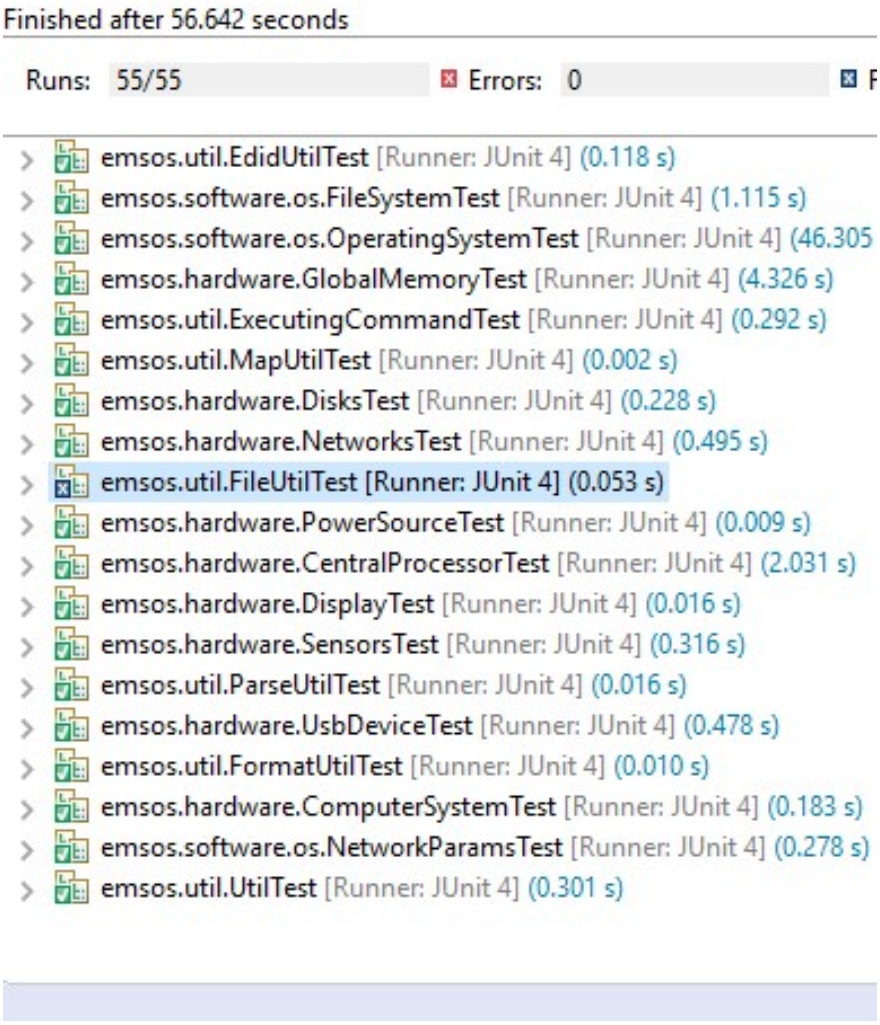

Figure 6 jUnit test results

```
importemsos.json.SystemInfo;
importemsos.json.hardware.CentralProcessor;
importemsos.json.hardware.Display;
importemsos.json.hardware.HardwareAbstractionLayer;
importemsos.json.software.os.NetworkParams;
importemsos.json.software.os.OperatingSystem;
importoutcomes.auth.AuthTokenGenerator;
importemsos.json.hardware.impl.DisplayImpl;
importjava.util.Arrays;

importjavax.json.JsonString;

importorg.springframework.web.bind.annotation.RequestHeader;
importorg.springframework.web.bind.annotation.RequestMapping;
importorg.springframework.web.bind.annotation.RequestMethod;
importorg.springframework.web.bind.annotation.ResponseBody;
importorg.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("v1")
publicclassHardwareInfoController  {
	publicstaticvoid main(String args[]) {
		SystemInfosi = newSystemInfo();
		HardwareAbstractionLayerhal = si.getHardware();
		OperatingSystemos = si.getOperatingSystem();

		System.out.println(os.toJSON());
		System.out.println(hal.toJSON());
	}
```

Figure 11 HardwareController class

```
                System.out.println(os.toJSON());
                System.out.println(hal.toJSON());
        }

public String authToken() {
        returnAuthTokenGenerator.nextToken();
    }
        // Operating System + Hardware related information
        @RequestMapping(headers = "User-Agent", method = RequestMethod.GET
, produces = "application/json")
        public@ResponseBody String getOsInfoInJSON(@RequestHeader("User-
Agent") String userAgent) {
System.out.println(this.authToken());
                SystemInfosi = newSystemInfo();
                OperatingSystemos = si.getOperatingSystem();
                HardwareAbstractionLayerhal = si.getHardware();
                if (userAgent.equals("")) {
                        returnos.toPrettyJSON();
                } elseif  (userAgent.equals("hw")){
                        returnhal.toPrettyJSON();
                }elseif (userAgent.equals("cpu")) {
                        returnhal.getProcessor().toPrettyJSON();

                }elseif(userAgent.equals("cpucount")) {
                        CentralProcessorprocessor=hal.getProcessor();
                        return ("{ \"No of
cpu\":"+processor.getPhysicalProcessorCount()+"}");
                }elseif (userAgent.equals("d")) {
                        returnhal.getComputerSystem().toPrettyJSON();
                }elseif (userAgent.equals("disk store")){
                        returnhal.getDiskStores()+"";
                }elseif (userAgent.equals("display")){
                        returnhal.getDisplays()+"";
                }elseif (userAgent.equals("memory")) {
                        returnhal.getMemory().toPrettyJSON();
                }elseif (userAgent.equals("nic")) {
                        returnprintNetworkParameters(os.getNetworkParams());
                }elseif (userAgent.equals("power")) {
                        returnhal.getPowerSources()+"";
                }elseif (userAgent.equals("sensors")) {
                        returnhal.getSensors().toPrettyJSON();
                }elseif (userAgent.equals("usb")) {
                        returnhal.getUsbDevices(true)+"";
                }elseif (userAgent.equals("l")) {
                        returnhal.getUsbDevices(true)+"";
                }elseif (userAgent.equals("m")) {
                        returnhal.getUsbDevices(true)+"";
```

Figure 12 Configuration and linking settings

```
<?xmlversion="1.0"encoding="UTF-8"?>
<assembly
	xmlns="http://maven.apache.org/plugins/maven-assembly-
plugin/assembly/1.1.3"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/plugins/maven-assembly-
plugin/assembly/1.1.3 http://maven.apache.org/xsd/assembly-1.1.3.xsd">
	<id>distribution</id>
	<formats>
		<format>tar.gz</format>
		<format>tar.bz2</format>
		<format>zip</format>
	</formats>
	<!-- Root of archive has files -->
	<includeBaseDirectory>false</includeBaseDirectory>
	<!-- Put all dependency jars (except modules) in /lib -->
	<dependencySets>
		<dependencySet>
			<useProjectArtifact>false</useProjectArtifact>

	<useTransitiveDependencies>true</useTransitiveDependencies>
			<outputDirectory>lib</outputDirectory>
			<unpack>false</unpack>
			<excludes>
				<exclude>${project.groupId}:*:*</exclude>
			</excludes>
		</dependencySet>
	</dependencySets>
		<files>


	</files>
	<!-- Include module jars in root level -->
	<moduleSets>
		<moduleSet>
			<!-- Enable access to all projects in the current
multimodule build! -->

			</binaries>
		</moduleSet>
	</moduleSets>
</assembly>
```

Figure 9 Configuration and linking settings

```
<useAllReactorProjects>true</useAllReactorProjects>
                    <!-- Now, select which projects to include in this
module-set. -->
                    <includes>
                          <include>com.github.emsos:emsos-core</include>
                          <include>com.github.emsos:emsos-json</include>
                    </includes>
                    <binaries>

      <includeDependencies>false</includeDependencies>
                          <outputDirectory>/</outputDirectory>
                          <unpack>false</unpack>
                    </binaries>
             </moduleSet>
      </moduleSets>
</assembly>
```

Security Scan results of code base

Matrix	**Reached Level**
Unit Test Coverage	92.30%
Security Analysis	
Coverage	87.50%
Line Coverage	87.20%
Condition Coverage	90.00%
Bugs	0
Complexity Function	1.2

Table 6 Security Scan results of code base

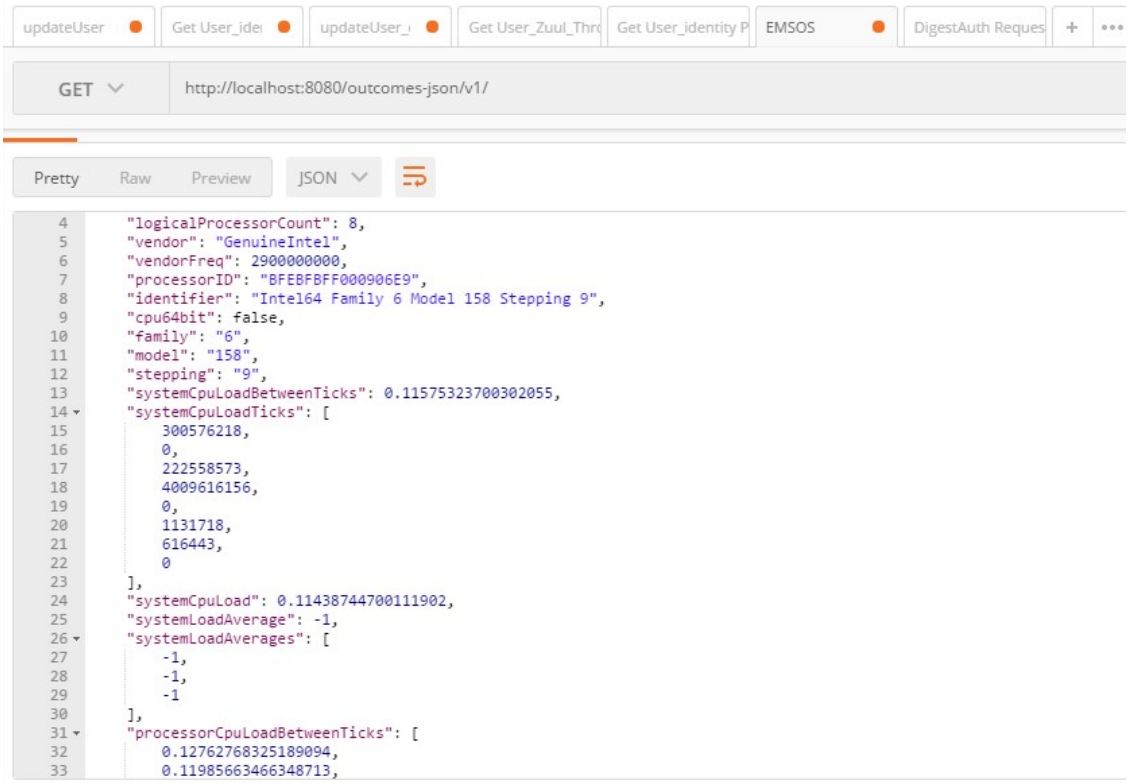

Figure 10 a sample JSON result of API

Appendix C –API documentation

Package	Description
emsos	Provides accessibility to OS information and Hardware profile **OS related :** OS type, Build version, **Hardware profile :** Manufacturer , model , description , version , serial number **Firmware Related :** manufacturer , description, version, name , release date
emsos.hd.hardware	Provide accessibility to CPU, Display, Memory Disks, NIC (interfaces), Sensors, Power and USB Devices
Emsos.com	Abstract layer to work with other classes
emsos.hd.plt.linux	Provide accessibility to power, memory and CPU which contains Linux systems. E.g. android, ubuntu core
emsos.hd.plt.unix.solaris	Provide accessibility to power, memory and CPU which contains solaris
emsos.hd.plt.unix.windows	Provide accessibility to power, memory and CPU which contains windows. E.g. windows mobile
emsos.hd.plt.unix.freebsd	Provide accessibility to power, memory and CPU which contains BSD Unix
emsos.hd.plt.unix.macos	Provide accessibility to power, memory and CPU which contains macos
emsos.jna.*	Provides Java Native Access libraries
emsos.json.*	Provide JSON formatted text for REST calls to get above OS access
emsos.sw.os	Provide cross platform accessibility for retrieve process, file systems, OS
emsos.util	Access utilities for parsing and formatting
emsos.json.util	Provide special utility methods for JSON
emsos.jna.plt.win.com	An special utility to access windows COM objects
emsos.util.plt.*	Provide utility access for different platforms.

Table 7 API Documentation

REST endpoints

GET <host hostip>/cpu
GET <host ip>/v1/

Header parameters : User-agent : (cpu/ram/file/,ded,ems,hw,fm)

GET <host ip>/gpu
GET <host ip>/mem
POST <host ip>/niclist
POST <host ip>/nicaccess/{id}
GET <host ip>/sensors
GET <host ip>/cpus
GET <host ip>/vcpus

GET <host ip>/staccess
GET <host ip>/staccess/{name}
GET <host ip>/systemaccess

GET <host ip>/mbd

GET <host ip>/pwrsources (get power resources)
GET <host ip>/PRO/ (retrieve processes that are currently running)
GET <host ip>/PID/{pid} (retrieve PID related details)
GET <host ip>/PROALLOCATE/ (allocating processors)
GET <host ip>/cpulgcore (get cpu logical cores)