International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Bit-By-Bit Communication Based Sensor Middleware APIs

Prevesh Kumar Bishnoi, Dharmender Kumar

Abstract: ***Mainly the frameworks are developed for the developers for rapid application development and less knowledge or without going in depth of a specific area of science and engineering. In this research an API framework for sensor communication using the concept of bit by bit communication has been proposed and tested. That can be used by the programmer and researchers for rapid application development and research. Design and development of APIs is in Java. Sensors used in the research and development are LM35, MQ7, MQ6 etc. All sensors are connected through microcontroller NodeMCU via Wi-Fi 802.11 protocol, Internet and HTTP server for web enablement of the process of data acquisition. All APIs or frameworks of this research will be integrated with the basic firmata4J or it will be available separately so that the user can use it in both the ways. The developed framework can be used for the rapid application development. Framework APIs will be solving the problem of controlling the devices by computer or cloud. In bit by bit communication with NodeMCU or microcontroller research, firmata4J methods make communication with NodeMCU firmata firmware as per the architecture of NodeMCU and pinout diagram according to standard I/O. But it has few limitations. Now there are two frameworks one is basic that is already available firmata4J and second is FirmSens4J. It is being developed in my research. It will be updating to firmata4J.***

Keywords: Microcontroller, Middleware, API, NodeMCU, Sensors, IoT, Physical Computing, Framework

I. INTRODUCTION

All APIs or frameworks of this research will be integrated with the basic firmata4J or it will be available separately so that the user can use it in both the ways.

In bit by bit communication with node MCU or microcontroller, firmata4J methods make communication with NodeMCU firmata firmware as per the architecture of NodeMCU and pinout diagram according to standard I/O. But it has few limitations. Now there are two frameworks one is basic that is already available firmata4J and second is FirmSens4J. It is being developed in this research project. It will be updating to firmata4J.

Manuscript received on March 15, 2020.
Revised Manuscript received on March 24, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Prevesh Kumar Bishnoi*, Assistant Professor, Computer Science and Engineering, SET, MUST, Lakshamangarh
Dr. Dharmender Kumar, Professor, Department of Computer Science and Engineering, GJUS&T Hisar

Standard pinout of NodeMCU given below according to firmata protocol access:

Necessary steps are to be performed without FirmSens4J.

1. Firmata4J Configuration on client machine where the development is to be done.
2. Find the specific firmata for NodeMCU or microcontroller.
3. Install the Arduino IDE and compile the specific selected firmata. Now, ready firmware is to be install on Microcontroller.
4. Collecting the basic knowledge of sensors with pinout diagram.
5. Connect it to NodeMCU physically.
6. Make the I2C connection of Node MCU with PC/Server.
7. Do the high-level programming for the sensor
8. Process and calibrate as per standard rules.

Necessary steps are to be performed with FirmSens4J. It will reduce the number of steps from the above.

1. Setup firmata4J and FirmSens4J framework with Java IDE(NetBeans)
2. Just switch on Node MCU.
3. Call Firmata Upload method.
4. Call specific sensor class method from library.

A relationship of fundamental API framework Firmata4J layer and derived framework FirmSens4J are shown in Fig. 1.

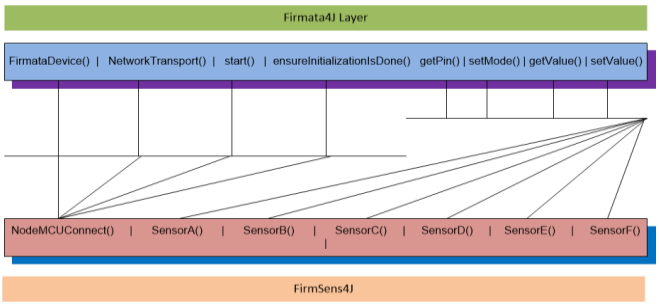

Fig. 1. API framework Firmata4J layer and derived framework FirmSens4J

II. REVIEW

Martin S. with others researchers did some work on the development of middleware for the mobile and ubiquitous learning ecosystems based on a reconfigurable plug-and-play architecture, Amarandei D., El-Said M. did the related work on the application of wideband VoIP middleware using embedded systems , Martin, Sergio; Diaz, Gabriel; Plaza, Inmaculada; Ruiz,

Elena; Castro, Manuel & Peire, Juan worked on the State of the art of frameworks and middleware for facilitating mobile and ubiquitous learning development and similar work carried out by Pasricha, Sudeep; Donohoo, Brad K. & Ohlsen, Chris,"A middleware framework for application-aware and user-specific energy optimization in smart mobile devices related to networking and mobile computing . But the most related work on the middleware, framework and APIs development with some specific algorithms was made by the Jonas Wåhslén, Thomas Lindh in his paper for the rapid application development in Java Script.

So the framework developed in Java script for the eHealth system. Work done particularly on WebSocket and Web Bluetooth APIs mainly on the Bluetooth for low energy. It is presently the applied for wireless network solution in area of eHealth and sports. However, most Bluetooth Low Energy sensors necessitate devoted submissions with very bounded development proficiencies. In his research he presented a process for fast development of software applications in distributed Bluetooth Low Energy and Internet of Thing systems in area of eHealth and sporting. The new approach suggested in the research used a JavaScript web framework based on HTML5 canvas, WebSocket and Web Bluetooth APIs. Research includes that the framework APIs will be applicable to develop the applications for observing physical activity and heart rate etc. The framework enables software and service operators to iteratively create, tune and deploy filter algorithms in distributed Bluetooth Low Energy and Internet of Thing systems, devoid of rebooting or restarting of programs using the concept of dynamic software updating.[1],[6],[9]-[11] Majeed A., Zia T.A., worked in area of Multi-set architecture for multi-applications for wireless sensor networks, Da Silva, Eduardo & Albini, Luiz Carlos P. did some research related to Middleware for mobile ad hoc networks , Sadhukhan P., Sen R., Das P.K. did some development for the dynamically deploy location based services onto heterogeneous mobile devices using bluetooth in indoor environment using middleware based approach and G. P. Silva; J. A. d. S. Borges worked in area of IoT for a Didactic Processor. But the most related work out of all reffered works was done by N. Tastan, A. Razaque, M. Ben Haj Frej, A. Saule Toksanovna, R. M. Ganda and F. Amsaad in his research on Burglary Detection Framework for House Crime Control. He designed and packaged framework for the reduction in the home or domestic thievery crime rate. Burglary Detection system includes the safe house application, and framework. Secure and safe house application includes two modes: protected and unprotected. First mode is activated when people are at home or property. Second one Unprotected mode is enabled when the house keeper are not far from the property. Whenever any prohibited member wants to enter the home, the messages are produced and sent to the landlords, In time intimation will help to caught the unauthorized person. The proposed Burglary Detection framework was tested an applied using ATmega328 microcontroller, Java technology and with Operating System Android.[2],[7],[12]-[14] Hsu I.-C., developed a Mobile ubiquitous attendance monitoring system using wireless sensor networks that is based on the API framework and middleware development . In the same area of middleware, Framework and APIs development M. Matijevic; V. Cvjetkovic resulted in their research Overview of architectures with ATmega328 boards as building blocks for data acquisition and control systems. A Single Board Computer with number of standard shields and sensors can be used as building blocks for fast development of network of intellectual devices with sensing, control and Internet access. Arduino family that has the ATmega328 microcontroller has high acceptance and big amount of sold things featuring open access, reliability, robustness, standard connections and low prices, possesses large potential for application of independent isolated measurement and control systems of various levels of intricacy. As Arduino boards can function self-sufficiently, they are complete small computer platforms that can achieve various tasks needing some types of interaction with the outer world. Arduino boards has the property of easily programmed by various types, and can be organized in various combinations creating some typical application designs. Finally the researchers abstracted, “Starting from basic and simple configurations, more advanced are gradually considered from the aspects of chosen way of programming and combining with other boards. Special attention is devoted to NodeJS as programming platform for Arduino boards and considerations of libraries used with Arduino boards like Johnny-Five, Galileo-io firmata equivalent, mraa library and other ways of program access to GPIO like Linux Sysfs”. Researchers selected special category board of ATmega328 the Arduino Uno, Arduino Due and Arduino.[3],[5]

Bellavista, Paolo; Montanari, Rebecca & Das, Sajal K., had completed some work in area of middleware development for Mobile social networking, similarly F. Baronti; A. Lazzeri; R. Roncella; R. Saletti on “Firmware/software platform for rapid development of PC-based data acquisition systems” and Killijian M.-O., Roy M. on “Data backup for mobile nodes: A cooperative middleware and an experimentation platform” and in the same area Tselikas N.D., Tselikis G.S., Sagias N.C. on Software and middleware technologies based on open APIs and protocols for modern service provision in telecoms but the most related work carried out by Lee, Dong-Kyu; Kim, Tae-Hyon; Jeong, Seol-Young & Kang, Soon-Ju resulted in research on a three-layer middleware design for supporting bi-directional location tracking approach for mobile asset. Due to the complexity and heavy traffic of the legacy location-awareness techniques, simultaneous locationing and tracking of plentiful mobile nodes in actual is difficult. For the solution of this problem, they suggest the three-tier middleware architecture called ubiquitous Mobile Asset Tracking Infra. In this proposed solution, all nodes (stationary and mobile) usually are under the standard IEEE 802.15.4 MAC protocol to promise the compatibility with the legacy wireless sensor network despite of mobile-stationary nodes co-existence network. To solve the problem bidirectional tracking in spite of the free mobility of the plentiful mobile nodes, they suggest a modest bidirectional location protocol”.

This is named as BLIDx (bidirectional location ID exchange) and its application into both types of movable and static nodes. On the other hand to protect the congestion due to the attentiveness of large number of mobile nodes into a specific location, they propose a solution of adding a particularly planned stationary node that is named as virtual sink (VS) node and mounting linked middleware mechanisms into the node. [4],[8],[15]-[17]

Need of Research and research gap identified*:* Research review concluded that the there are lots of research and framework has been made related to the title but exactly no solution was made about the development and research on physical computing, IoT and sensor communication with application program.

So that huge application can be developed in a short time and with less knowledge of electronics, electrical and hardware or sensors by software application developer. APIs of framework can be called by the application programmer and researchers to make the communication enable with sensors for different physical quantities.

III. BIT-BY-IT COMMUNICATION BASED SENSOR DESIGNS

Fundamentally selected methods of the firmata4J will be used for the development of FirmSens4J. Description of necessary methods for this implementation and design are given below:

FirmataDevice(): This method use one argument that is object of class Network Transport. This object is composed of two values IP address and port as per the standard of TCP/IP model. This method returns one device id of class IODevice. It is member of class org.firmata4j.firmata.FirmataDevice

Constructor prototype: FirmataDevice(TransportInterface transport)

NetworkTransport(): It method with two arguments. First arguments is IP address and second one is the port on which firmata is communication. It return a object of class NetworkTransport Which is to be used for further communication by passing this object to FirmataDevice method.

Constructor Prototype: NetworkTransport(InetAddress ip, int port)

start(): After setting all necessary parameters, this method is called with object of class IODevice by FimataDevice() method. It is used to start the device for operation of read write but before it on more method is used that is ensureInitializationIsDone.

Method prototype: void start()

ensureInitializationIsDone(): This method is used to ensure the initialization of device for operation. This method is also the member of class IODevice.

void ensureInitializationIsDone()

getPin(): This method is member of IODevice. It is used for the purpose of capture the specific pin of Microcontroller through firmata protocol. It takes one pin number as argument. It return pin id to an object of class Pin.

Pin getPin(int index)

setMode(): Whenever any pin is to be used. Mode of this pin is to be set. Mode means as input or output pin or analog pin. It is member of Pin class.

Mode getMode() and void setMode(Mode mode)

getValue(): This method is used to get the value on specified pin number. It is called with the object of Pin class.

long getValue()

setValue(): This method is used to set the value on specified pin number. It is called with the object of Pin class.

void setValue(long value)

1. Connect to node MCU: Node MCU working as microcontroller in the whole research. Algorithm for connecting the NodeMCU(Fig. 2)is representing the way of connecting of NodeMCU to application program. First of all, the firmata is to be setup on NodeMCU. Now, firmata device method is to be called. Firmata method has one argument that is class Network Transport. Constructor of this class has two arguments IP address and port. After this, it starts and ensures that the initialization is done. Finally, true or false will be returned if whether it is connected. This is very fundamental API to make the connection with ESP 8266.

Just after connection and basic setup of necessary hardware and software for the connection of NodeMCU. The above algorithm will be used for the connection. In connection algorithm, a high level

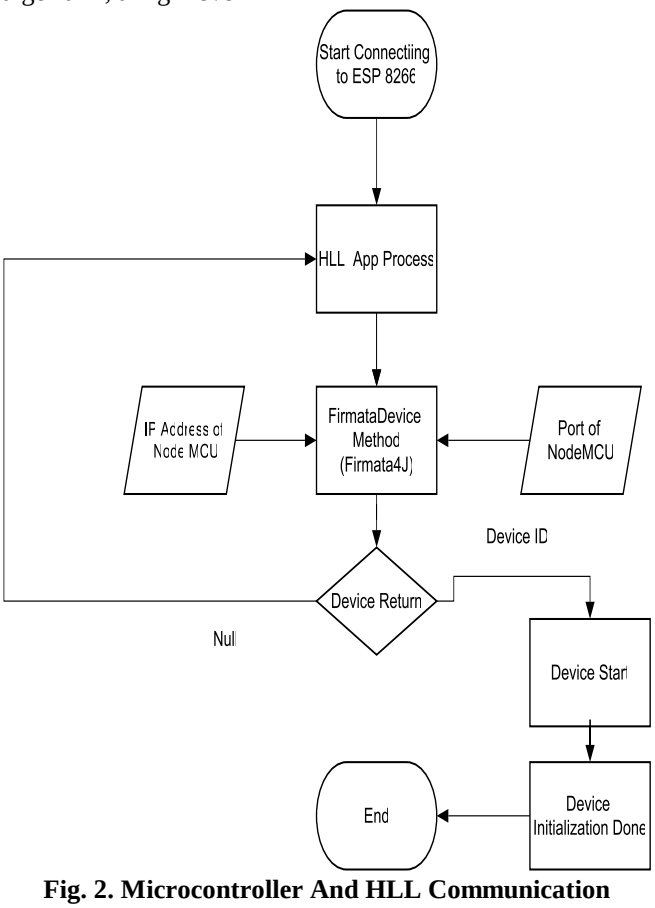

Fig. 2. Microcontroller And HLL Communication

language program(HLL) or process will call the firmata device method with TCP layer port and IP layer address of the target NodeMCU. If it return false, number of attempts will be made else the device ID will be returned.

After it the device start process will be initiated. And connection establishment will be confirmed. This algorithm is only to make the connection with NodeMCU, not for sensor connection and data acquisition.

2. Connecting to specific sensor: After making connection to node MCU, firmata will help to access specific pins of the NodeMCU. A sensor will be connected to specified pins of the NodeMCU. The first method that is be called, will be NodeMCUConnect(). Now, Node MCU will get physical connectivity on IEEE802.11 standards over Wi-Fi. If it return true, pin number will be specified. After it, pin will be set in to specific mode like Input, output or analog. Now pin can be read or write. After that a value formation method will be called to get the value in to human understandable format. Algorithm in Fig. 3 is demonstrating this method.

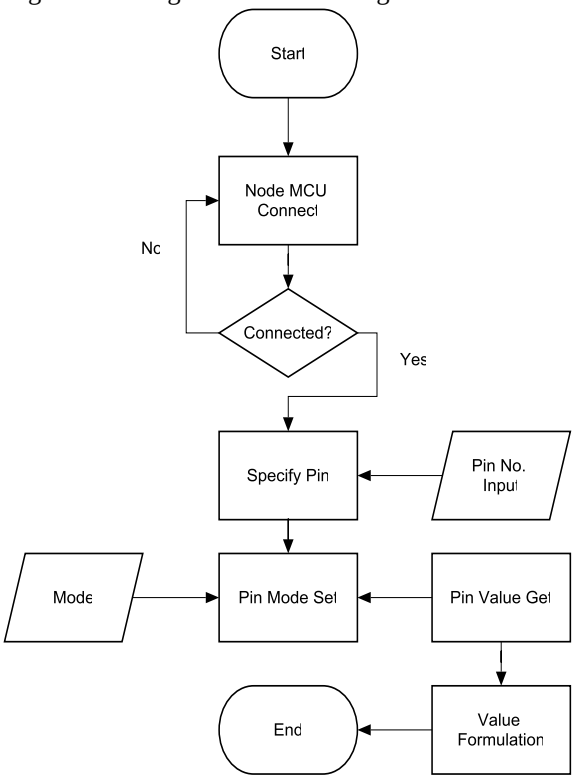

Fig. 3. Connecting To Specific Sensor

As we discussed in the previous algorithm of making connection with NodeMCU, the basic connectivity was made for bit by bit communication. Fig. 3 shows an algorithm for making communication with individual sensor LM35, MQ7, MQ6 etc. Pin number on which the pin was connected, the pin number is to be passed as argument to specified method according to the sensor type and number of connecting pins in the sensor for data acquisition. After mention the pin number the mode of pin is to be set. The mode like input output etc. Mode is particularly for the purpose to get or set data. After reading the value of sensor parameter the value will be sent back to the caller method. For the web enablement of the process the method is to be called by JSP code.

IV. FRAMEWORK:

On the basic of above algorithms the following APIs has been developed in Java. For the following sensors:

1. Hall Magnetic: This is used to get the value of the magnetic field for any specific device. It is measured in the term of output voltage that is directly proportional to value of magnetic field. In the wheel illustrated with two equal distance magnets, the voltage from the sensor will be peak twice for each revolution.

2. LM35: LM35 is a device that belongs to the category of integrated analog temperature sensor that's resulted voltage/power is measured and calibrated for the temperature's unit degree centigrade. It has been designed in such a way that no any calibration/trimming is required to get fine correctness for the LM35 sensor. Due to small output impedance, direct output and precise inbuilt calibration the interfacing of it become very easy.

3. MQ3: MQ3 sensor is also a gas sensor. It is used for alcohol detection. . It is a low-cost semiconductor sensor that can detect the presence of alcohol gases in the concentration of 0.05 mg / L to 10 mg / L. It is using SnO2 material. The conductivity of SnO2 is low in clean air, and it is increasing as the concentration of alcohol increases.

4. MQ6: The MQ-6 sensor is used in devices that detect gas leakage in the family and industry. This module has high sensitivity to LPG, Butene, Propane and LNG. It can also be used to detect the presence of alcohol, cooking smoke and cigarette smoke. The sensor return the concentration of gases in as equivalent volt that is to calibrated and normalized. The electronic circuit of is responsible for calibration and normalization.

5. MQ7: It is a s carbon monoxide gas sensor. It detects presence of CO in the air. Sensor produces the respective voltage for the CO presence. After processing by microcontroller/microprocessor the result can be used. MQ7 gas sensor return the gas concentration value in PPM from 10 to 10000. The temperatures limitations for the MQ7 sensor is in range of 10 to 50 degree Celsius and consume less than 150 mA at 5 V.

6. Soil Moisture Sensor: It works on the concept of measurement of the capacitance to find the water content of soil. To use it, simply introduce this sensor into the soil to test the moisture after proper connection with microcontroller and display. Finally the water presence in soil or moisture will be is displayed on the LCD in percent.

A. Class LM35

By applying the above algorithms a framework has been developed in Java. That is having classes for each sensor. Each class has number of methods. Every method has the implemented concept of all above algorithms for communication and data acquisition. The following document detail is having the prototype or signature of methods and the inherited classes like Object Table 1 representing the method signature of proposed framework.

- java.lang.Object
- mobilemiddleware.LM35

public class **LM35**
extends java.lang.Object

- *Constructor Summary*

Constructors

Constructor and Description
LM35()

- *Method Summary*

All MethodsInstance MethodsConcrete Methods

Modifier and Type	Method and Description
double	**rawLM35**(int pin, org.firmata4j.IODevice device)
double	**rawLM35ScaleC**(int pin, org.firmata4j.IODevice device, double fact)
double	**rawLM35ScaleF**(int pin, org.firmata4j.IODevice device, double fact)
double	**tempLM35C**(int pin, org.firmata4j.IODevice device)
double	**tempLM35F**(int pin, org.firmata4j.IODevice device)

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- *Constructor Detail*

- **LM35**

public LM35()

- *Method Detail*

- **rawLM35**

- public double rawLM35(int pin,
- org.firmata4j.IODevice device)
throws java.io.IOException

Parameters:
pin - This the Pin Number on Node MCU
device - It is Device NodeMCU or Arduino
Returns:
It Returns the raw value
Throws:
java.io.IOException

- **tempLM35C**

- public double tempLM35C(int pin,
- org.firmata4j.IODevice device)
throws java.io.IOException

Parameters:
pin - This the Pin Number on Node MCU
device - It is Device NodeMCU or Arduino
Returns:
It returns temperature in Degree centigrade
Throws:
java.io.IOException

- **tempLM35F**

- public double tempLM35F(int pin,
- org.firmata4j.IODevice device)
throws java.io.IOException

Parameters:
pin - This the Pin Number on Node MCU
device - It is Device NodeMCU or Arduino
Returns:
It returns temperature in Degree Fahrenheit
Throws:
java.io.IOException

- **rawLM35ScaleC**

- public double rawLM35ScaleC(int pin,
- org.firmata4j.IODevice device,
- double fact)
throws java.io.IOException

Parameters:
pin - This the Pin Number on Node MCU
device - It is Device NodeMCU or Arduino
fact - It is a calculated value to calibrate the sensor
Returns:
It returns temperature in Degree centigrade
Throws:
java.io.IOException

- **rawLM35ScaleF**

- public double rawLM35ScaleF(int pin,
- org.firmata4j.IODevice device,
- double fact)
throws java.io.IOException

Parameters:
pin - This the Pin Number on Node MCU
device - It is Device NodeMCU or Arduino
fact - It is a calculated value to calibrate the sensor
Returns:
It returns temperature in Degree Fahrenheit
Throws:
java.io.IOException

Table 1: Method Signature Summary of framework

All APIs were developed as per the standard process of Java development. They are having the details of arguments, variables and return type. It provides the full run time help when any scientist or engineer or researcher will be using framework.

B. Class Diagram for sensors classes:

A class diagram of the one class LM35 sensor for the reference to the framework is given here following diagram. It is in Unified Modelling Language has been used to design the class diagram. That represents the design of a system. It also indicate the relationships between system classes, their characteristics, operations (or methods), and objects. It represents the number of methods in the side the class with full signature. (Fig. 4 and Fig. 5)

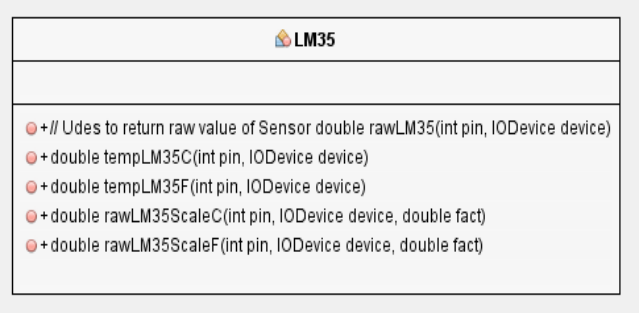

Fig. 4. LM35 Class Diagram

Above one for the LM35 class design. It is having five numbers of methods for different purposes. It different methods are returning the results in different forms either the raw reading that can be calibrated accordingly. But few methods are returning temperature in specific unit.

Three more designs for the MQ3, MQ6 and MQ7 are also given in the following UML diagram. Diagram is representing the number of methods of three more classes.

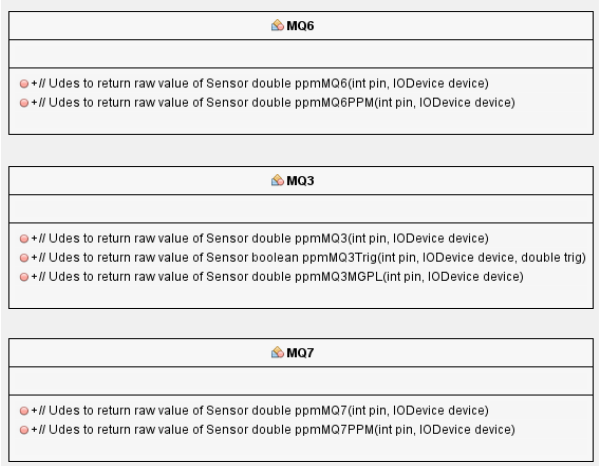

Fig. 5. MQ6, MQ3, MQ7 Class Diagram

V. RESULTS

For the evaluation of framework API the following parameters were considered:

1. Memory Utilization
2. CPU Utilization:
3. Garbage Collection
4. Thread and Classes loaded

The whole research and development is based on the JVM, Tomcat or any other Java technology server or containers. Some other servers may be JBoss, Abyss and others. The tomcat is containers for the web applications in JSP etc. JVM is containers for the Tomcat server itself and framework.

When the tomcat server is started the all the parameters Memory Utilization, CPU Utilization, Garbage Collection and Thread and Classes loaded are increasing continuously. After started, the tomcat all parameters come to the stable state and reach to the saturation. In Fig. 6 graph is representing the value of all such parameters after the completion of start of tomcat.

Space time utilization of Methods for MQ7:

For the MQ7 Memory Utilization, CPU Utilization, Garbage Collection and Thread and Classes loaded graphs are given the Fig. 7. CPU utilization of MQ7 utilization is first increasing and reaches to the maximum value and then decreasing. When the curves going up it predict that the connection establishment started and when it achieve the max value the data, this is acquisition and reading of data. When it is coming down, it is showing completing, releasing of threads classes and closing the connection. So the total width of the curve is the time to read and data acquisition. And the maximum level shows the maximum possible utilization of processor of CPU.

If it is increasing continuously, it indicate some error like the network not available or memory leak, Or some failure of the sensor or microcontroller. Memory Utilization graph shows the maximum available heap and the utilized heap. If it is increasing continuously, it predict some memory leak. Up and down in the graph shows that the object are being created and destroy and memory is released as standard practice. Fig. 7, Fig. 8 and Fig. 9 are representing the same for different sensors.

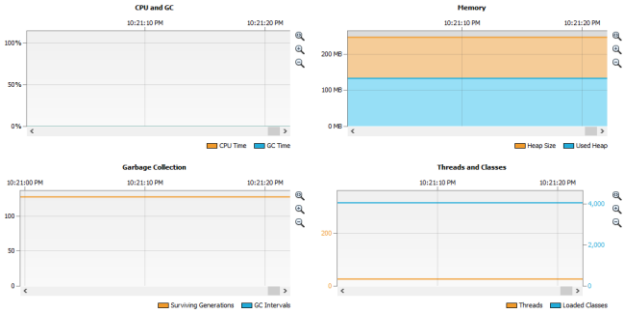

Fig. 6. Standard Tomcat Telemetry

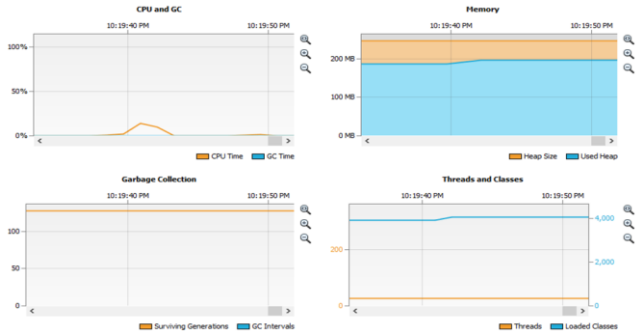

Fig. 7. MQ7 API Space Time Utilization

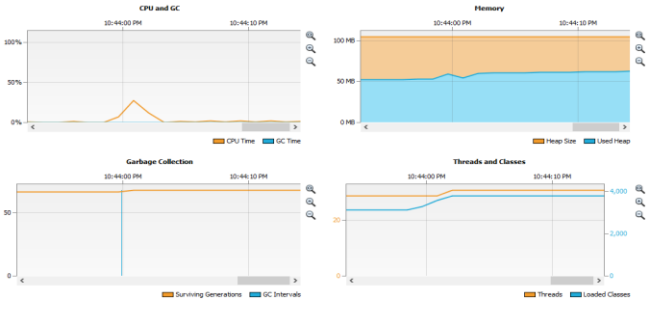

Fig. 8. MQ6 API Space Time Utilization

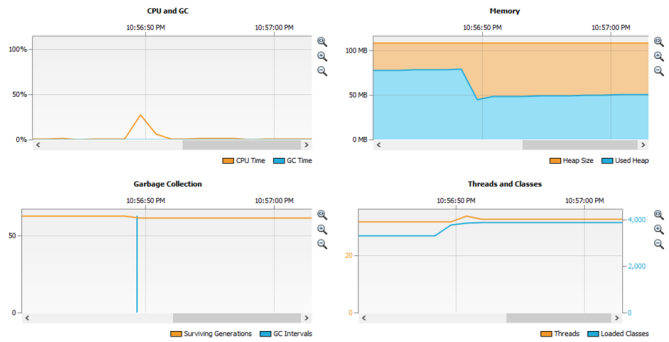

Fig. 9. MQ3 API Space Time Utilization

There are many similarities and differentiation in different graphs for different sensors. All such parameters are dependent to many quantities and variables likes time of making connection, number of request at the server ate that specific time etc.

Garbage collection graph predict that freeing the heap for unreferenced objects. If this curve is increasing regularly or exponentially means API performance is not up to the mark. The complete framework not only tested for all above parameters, but tested with some piolet project and Unit testing of Java.

Thread and classes graph indicating the number of thread and classes created and loaded at the time of calling of API in HLL program.

Table 2: Execution time and processor utilization

Reading No. at Different Timings	Time (T) Sec	Highest CPU Utilization (%) of Gas Sensors		
		MQ7	MQ6	MQ3
1	40	0		
2	41.212	14.08		
3	43.408	0		
Time Taken in Execution(MQ7)	3.408 Sec			
4	44.00		0	
5	46.307		24.978	
6	48.529		0	
Time Taken in Execution(MQ6)	4.529			
7	50.0			0
8	51.282			24.196
9	53.504			0
Time Taken in Execution(MQ3)	3.504			

Table 2 shows the CPU utilization and execution time at different time. MQ7, MQ6 and MQ3 processor utilization is 14.08, 24.978 and 24.196 respectively. Time T can be used to get the execution time of API for specific sensor. Execution time includes the time of communication with sensor for data acquisition, connection via NodeMCU.

Exactly in the same way the result for the LM35 and Soil and moisture sensor are predict. Differentiation between all such sensor is only data reading and control time. Time of making the connection will be approximately same.

API performance has been tested on multithread creation and time consuming. It is given in Fig. 10.

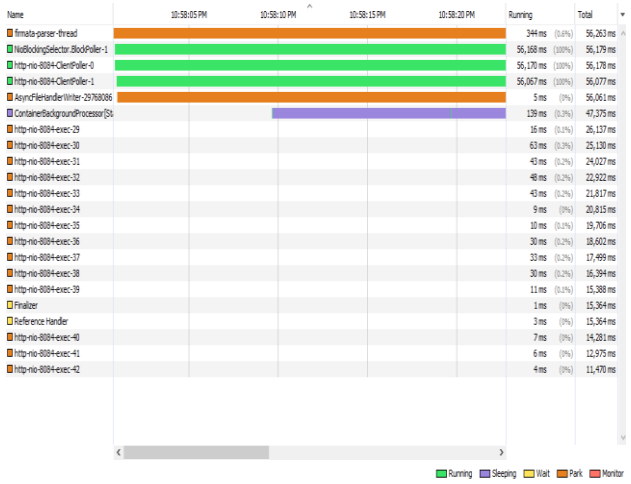

Fig. 10. Multithread Execution Time

VI. CONCLUSION

The developed framework will provide the easy to use and rapid application development of IoT and physical computing software. Big applications can be developed in the short time i.e. support in rapid application development. Developers with the minimum knowledge of hardware and microcontroller or electronics might be developing the physical computing software easily. Framework can be extending by the addition of more APIs for new type of sensors. APIs developed for the many gas sensors like MQ3, MQ6 and MQ7 were tested with the standard NetBean telemetry over Tomcat server for the parameters like CPU time, GC time, Memory utilization, Garbage collection and Thread load timings and executions.

ACKNOWLEDGMENTS

Authors do thanks to the CSE Departments of Mody University, CPU University, GJU Hisar authorities for providing the research facilities, environment and necessary platform for this research.

REFERENCES

1. Jonas Wåhslén, Thomas Lindh,(2018). A Javascript Web Framework for Rapid Development of Applications in IoT Systems for eHealth. IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom),
2. N. Tastan, A. Razaque, M. Ben Haj Frej, A. Saule Toksanovna, R. M. Ganda and F. Amsaad(2019). Burglary Detection Framework for House Crime Control, 19th International Conference on Computational Science and Its Applications (ICCSA), Saint Petersburg, Russia, 2019, pp. 152-157
3. M. Matijevic; V. Cvjetkovic(2016). Overview of architectures with Arduino boards as building blocks for data acquisition and control systems. IEEE Conferences 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)
4. Lee, Dong-Kyu; Kim, Tae-Hyon; Jeong, Seol-Young & Kang, Soon-Ju(2011). A three-tier middleware architecture supporting bidirectional location tracking of numerous mobile nodes under legacy \WSN\ environment . Journal of Systems Architecture
5. Hsu I.-C(2010). Mobile ubiquitous attendance monitoring system using wireless sensor networks. 2nd International Conference on Education Technology and Computer, ICETC 2010,22 June 2010 through 24 June 2010,Shanghai .

6. Martin S., Castro M., Talevski A., Potdar V.(2010). A middleware for mobile and ubiquitous learning ecosystems based on a reconfigurable plug-and-play architecture: Application to mashups. 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010,20 April 2010 through 23 April 2010,Perth
7. Majeed A., Zia T.A.(2010). Multi-set architecture for multi-applications running on wireless sensor networks. 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010,20 April 2010 through 23 April 2010,Perth
8. Bellavista, Paolo; Montanari, Rebecca & Das, Sajal K. (2013). Mobile social networking middleware: A survey . Pervasive and Mobile Computing
9. Amarandei D., El-Said M(2010). Implementation of wideband VoIP middleware using embedded systems. IEEE International Conference on Electro/Information Technology, EIT2010,20 May 2010 through 22 May 2010,Normal, IL
10. Martin, Sergio; Diaz, Gabriel; Plaza, Inmaculada; Ruiz, Elena; Castro, Manuel & Peire(2011). State of the art of frameworks and middleware for facilitating mobile and ubiquitous learning development . Journal of Systems and Software
11. Pasricha, Sudeep; Donohoo, Brad K. & Ohlsen, Chris(2015). A middleware framework for application-aware and user-specific energy optimization in smart mobile devices. Pervasive and Mobile Computing
12. Da Silva, Eduardo & Albini, Luiz Carlos P (2014). Middleware proposals for mobile ad hoc networks . Journal of Network and Computer Applications.
13. Sadhukhan P., Sen R., Das P.K.(2010). A middleware based approach to dynamically deploy location based services onto heterogeneous mobile devices using bluetooth in indoor environment. 2nd International Conference on Advanced Communication and Networking, Miyazaki.
14. G. P. Silva; J. A. d. S. Borges(2018). A Didactic Processor and Simulator for IoT. IEEE Conferences 3rd International Conference of the Portuguese Society for Engineering Education (CISPEE)
15. F. Baronti; A. Lazzeri; R. Roncella; R. Saletti(2010). Firmware/software platform for rapid development of PC-based data acquisition systems. 17th IEEE International Conference on Electronics, Circuits and Systems
16. Killijian M.-O., Roy M(2009). Data backup for mobile nodes: A cooperative middleware and an experimentation platform. International Conference on Dependable Systems and Networks,Lisbon.
17. Tselikas N.D., Tselikis G.S., Sagias N.C(2010).Software and middleware technologies based on open APIs and protocols for modern service provision in telecoms.14th Panhellenic Conference on Informatics, Tripol

AUTHORS PROFILE

Mr. Prevesh Kumar Bishnoi received his M.Tech CSE degree from Guru Jambheshwer University, India in 1999. This author is. member of many national and International Research/Standard/Societies like IEEE, ISTE, CSI, ACM etc and working as Assistant Professor at Mody University Lachhmangarh, Rajasthan

Prof. Dharmender Kumar completed his Bachelor of Technology in Computer Science & Engineering (CSE) from CRSCE Murthal, GJUS&T, Hisar, Haryana, Master of Technology in CSE from Kurukshetra University, Kurukshetra and Ph.D in CSE from GJUS&T, Hisar, Haryana, India. Currently he is working as Professor in Department of Computer Science and Engineering in GJUS&T, Hisar. His main research work focuses on Artificial Intelligence, Data Mining and Big Data Analytics.