ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 2, February 2013

ZIGBEE BASED IMPLEMENTATION ON EXHAUST GAS DETECTION SERVICES ORIENTED SYSTEM RESEARCH

SK Sabiha Begum[1], B. UmaSankar[2]

[1]M.Tech Scholar, Dept. of ECE (VLSI & ES), VRS & YRN Engineering College, Chirala, AP, India

[2]Asst. Professor, Dept. of ECE, VRS & YRN Engineering College, Chirala, AP, India

ABSTRACT: **The vehicle exhaust gas emission is directly related with the quality of air. This paper describes the research and development of an ZIGBEE based exhaust gas detection system applying the service oriented architecture for the purpose of environmental protection. An edge engine supporting the EPC global ALE specification, a complex event processing engine dealing with the complex streaming ZIGBEE events and service bus, which are involved in the architecture of the system, are discussed both separately and in a whole picture. The edge engine is composed of four modules, which are kernel module, device management module, events filter module, and configuration module. It provides features to encapsulate the applications from device interfaces; to process the raw observations captured by the readers and sensors; and to provide an application-level interface for managing readers and querying ZIGBEE observations. Event processing engine consists of a network of event processing agents running in parallel that interact using a dedicated event processing infrastructure, which provides an abstract communication mechanism and allows dynamic reconfiguration of the communication topology between agents at run-time. Web Services and asynchronous APIs are pivotal system implementation for flexible components reuse, dynamic configuration and optimized performance. Service bus, through its service integration and management capabilities, is the backbone of the system. Lastly, the system is proven to be effective with high performance in the benchmark.**

Keywords- ZIGBEE; Exhaust Gas Detection; EPCglobal ALE; Service Oriented; Complex Event Processing

I. INTRODUCTION

There's a trend in automobile exhaust pollution on urban air quality impact of a growing pollutants into the atmosphere as the number of vehicles has increased year by year. In order to effectively improve the environment and air quality, we should take active and effective pollution control measures to service oriented architecture. First, we will take a look at the whole picture, the architecture and components of the system in the second section. The details of the edge engine supporting the EPCglobal ALE specification, complex event processing engine dealing with the streaming ZIGBEE events and the service bus, which are involved in the architecture of the system, are discussed separately in the section three. Lastly, the system is proven to be effective with high performance in the benchmark, and the conclusion is given.

II.ARCHITECTURE AND COMPONENTS

The system is conceptually comprised of event sources, event-driven application logic, and event sinks. The event sources, which are vehicles, sensors and ZIGBEE readers, generate streams of ordinary event data. The applications logic listens to the event streams, processes these events, and generates notable events. Event sinks, such as system management console, certain event subscribers and queries, receive the notable events, and take actions to do vehicle maintenance management according to vehicle emissions standards. Event sources, application logic, and event sinks are de-coupled from each other; one can add or remove any of these components without causing changes to the other components. This is an attribute of event driven architectures. Let's explain how the system works by a simple example. All the ordinary vehicles' emissions can be monitored by certain gas detection sensors. And if the emission is extra- standard, the warning event causes an operation of reading ZIGBEE tag attached on the vehicle. Then, all the emission data and vehicle will be sent to events repository for query, and alert the events subscribers to do the logical actions.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 2, February 2013

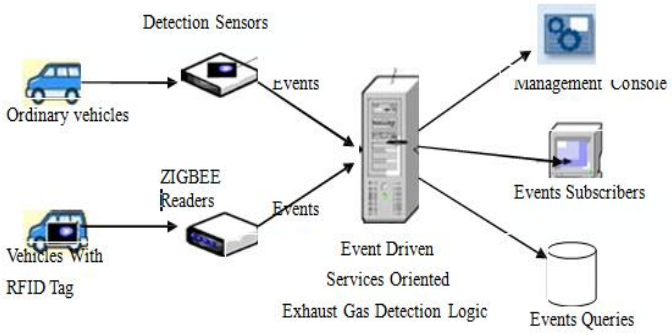

Figure 1. Conceptual Overview of the Exhaust Gas Detection System

Improve vehicle exhaust pollution. Monitoring vehicle emissions is an effective method that we can adopt. Through detecting and monitoring vehicles on the road, we can enforce owners of the selected excessive amount of vehicles exhaust to do vehicle maintenance management according to vehicle emissions standards for vehicle emissions control purposes.For the purpose of environmental protection, this paper Exhaust Gas Detection Logic describes our effort at research and development of an ZIGBEE based exhaust gas detection system applying the implementation architecture consists of the edge engine, the complex event processing engine and service bus. The edge engine is composed of four modules, which are kernel, device management, events filter, and configuration module. It provides features to encapsulate the applications from device interfaces; to process the raw observations captured by the readers and sensors; and to provide an application-level interface for managing readers and querying ZIGBEE observations. As shown in the Figure 2, the web services, that edge engine exposes, canbe invoked synchronously or asynchronously by service bus and application logic components, receive request specifications and return gathered events.

In other hand, the gathered events will be sent to complex event processing engine asynchronously because of the ad-hoc appearance of raw events. The complex event processing engine communicates with service bus and the application logic components synchronously. Application logic components can call the proxy service provided by service bus either waiting the response or in unblocked way.

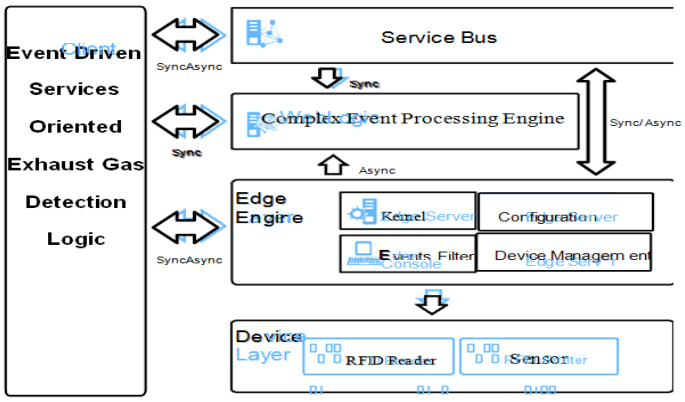

Figure 2. System Architecture and Components

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 2, February 2013

III.DESIGN AND IMPLEMENTATION

The system is implemented mainly using Java, XML, and Web Service technologies. Java enables the system independent to any specific operating system; XML makes the event and data values been transferred in a uniform and self- explanation way; Web Service enable the logic functions can be called from heterogeneous implementation asynchronously. All of the design and implementation keep some core principle: standardized, performance, and scalability.

A. Edge Engine

The core function of edge engine is implementation of ALE specification. It means that the kernel of edge engine receives an ECSpec and return an ECRport, which are both XML fragments. An ECSpec is a complex type that defines how an event cycle is to be calculated. There are two ways to cause event cycles to occur. one or more clients may subscribe to that ECSpec using the subscribe method. The ECSpec will generate event cycles as long as there is at least one subscriber. The second way is that an ECSpec can be posted for immediate execution using the immediate method.

As summarized in Figure 3, an ECSpec is said to be active if reads are currently being accumulated into an event cycle based on the ECSpec. Standing ECSpecs that are requested using subscribe may transition between active and inactive multiple times. ECSpecs that are requested using poll or created using immediate will transition between active and inactive just once.

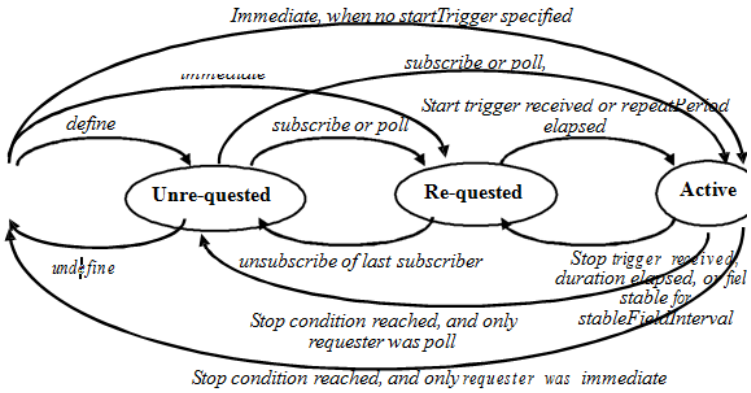

Figure 3. States transfer diagram of kernel module in edge engine

Events Filter in implemented by package filter and package filtertypes. It can work at both include mode and exclude mode. When the ECRport returns, it must match at least one pattern in the "include pattern" list, and not match any pattern in the "exclude pattern" list. It can be described mathematically as follow:

$$F(R) = \{epc|epc \in R \ \& \ (epc \in includePattern1 \mid epc \in includePattern2 \mid ... \mid epc \in includePatternN) \ \& \ !(epc \in excludePattern1 \mid epc \in excludePattern2 \mid ... \mid epc \in excludePatternN)\}$$

Device management module devotes to be the single point of device access and administration with various physical readers and sensors from different providers. Having the device management module, the system expansion with adding new devices support becomes more convenient. Implementation is organized into three packages, reader, eventhandler and readertypes. Package reader contains interfaces and classes representing the physical and logic readers. Package eventhandler contains interfaces and classes representing the reader event and event handlers. Implementation classes of physical and logic readers are stored in package readertypes, for example, XXXPhysical Reader is the code for XXX physical reader.

Configuration module set up the system configuration and managing the kernel, events filter and devices management modules in declaration way with dynamical modification, which are stored in physical XML files. The configuration items are organized in a hierarchical way, and ConfigManager is the central controller of management, which is implemented by a singleton class ConfigManagerImpl.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 2, February 2013

B. Complex event processing engine

The complex event processing engine is a low latency framework for our event driven system. It is a light weight application server which connects to high volume data feeds and match events based on user defined rules.

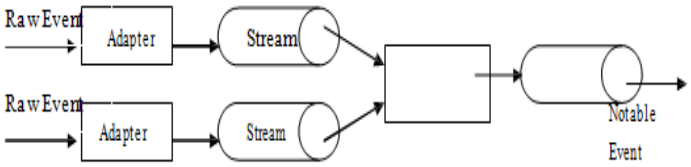

Figure4. Overview of complex event processing engine

As shown in Figure 4, the complex event processing engine comprises of three main component types. Adapters interface directly to the inbound event sources. Adapters understand the inbound protocol, and are responsible for converting the event data into a normalized data that can be queried by a processor. Adapters forward the normalized event data into Streams. Streams are event processing endpoints. Among other things, streams are responsible for queuing event data until the event processing agent can act upon it. The event processing agent removes the event data from the stream, processes it, and may generate new events to an output stream. The application logical code registers to listen to the output stream, and is triggered by the insertion of a new event in the output stream. The system makes use of a set of external services, such as web service and file writers, to forward the generated events to external event sinks. Events are represented as POJOs following the JavaBeans conventions, in which properties are exposed through getter methods. A SQL-like language is used in the continuous events processing model, which has SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY clauses.

C. Service Bus

The service bus provides an abstraction layer implementation of our complex architectures via an event- driven system on top of a standards-based messaging engine. Ideally, we should replace all direct contact with the logic components, so that all communication takes place via the bus In order to achieve this objective, the bus encapsulates the functionality offered by component applications into web services built around the exchange of XML messages in a platform-independent manner. XQuery implementation is used to query and operate collections of XML data. Although multiple transport protocols such as JMS/FTP are supported by our service bus framework, we use HTTP for better performance and scalability, because the service bus is the backbone of our system.

IV. PERFORMANCE ANALYSIS

To evaluate the performance of this system, we did a benchmark in emulation environment. The system was deployed on three Dell Latitude D620 with 2GHz dual-core CPU and 2GB memory. Simulator of ZIGBEE readers and sensors was used to generate physical events, and the workload was added gradually. System behaves to be effective with high performance in this benchmark, it works well at an injection rate of 1000 events/sec, and the average event process latency was 367 microseconds. The event driven pattern, asynchronous interaction pattern in the service oriented architecture are proven to make the system run fast and scalable.

V. CONCLUSIONS

Some relevant study and exploration works has been done by many universities and research groups at present. But most of them are just having interest at one of the technologies we used, such as event driven, ZIGBEE or SOA without a application scenario. Solution provided by the commercial software vendors are mostly based on their own products or technology applications and in the early stages of its exploration, and hard to extend. And, our work is also at its very beginning, most of the works are at experimental stage, and have limitations. There are many works to be done in our event driven ZIGBEE based exhaust gas detection system applying the service oriented architecture our just discussed. We are working at

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 2, February 2013

supporting full EPC network standard protocols including EPCIS and ONS, and extend the system to leverage the power of wireless sensor networks to gather environment information more effectively and flexibly.

To sum up, our exhaust gas detection system has a very bright application perspective, and will be advanced through evolution adopting the event driven SOA and wireless sensor networks technology.

VI.ACKNOWLEDGMENT

This work is carried out by Tianjin University and BUPT. Thanks a lot to our colleagues from TJU and BUPT contributing to this project and giving suggestions to this paper.

REFERENCES

[1].EPC global Inc, "The EPCglobal network: overview of design, benefits, and security", Available: http://www.epcglobalinc.org.

[2]. Kambhampati, S., An Event Service for a Rule-Based Approach to Component Integration, M.S, Thesis, Dept. of Computer Sci. and Eng., Arizona State University, 2003

[3]. Karl Aberer , Manfred Hauswirth , Ali Salehi,A middleware for fast and flexible sensor network deployment, Very Large Data Bases (VLDB) Seoul, Korea, 2006.

[4]. Luckham D C, Frasca B. Complex Event Processing in Distributed Systems[R]. Stanford University, Technical Report: CSL-TR-98-754, 1998.

[5]. Mahadevan, A. Barker, J. The beginning of a sensor network era: an inter-enterprise ZIGBEE-agent enabled EPCglobal supply network architecture and the matrix modelDigital Object Identifier: 10.1109/ASENSE.2005.1564538

[6]. Neil Gershenfeld, Raffi Krikorian, Danny Cohen, The Internet of Things, Scientific American Magazine;October 2004

[7]. Sanchez Lopez, Daeyoung Kim, Taesoo Pak, "A service framework for mobile ubiquitous sensor networks and ZIGBEE", Wireless Pervasive Computing, 2006, pp 1~6.

[8]. S. Perera, C. Herath, J. Ekanayake, E. Chinthaka, A. Ranabahu, D. Jayasinghe, S. Weerawarana, and G. Daniels. Axis2, middleware for next generation web services. In Proc. Int. Conf. on Web Services (ICWS'06), pages 833–840

[9]. Xingxin Gao, "An approach to security and privacy of ZIGBEE system for supply chain", E-Commerce Technology for Dynamic E-Business, 2004. IEEE International Conference on ,2004 pp:164 – 168.

[10]. Wang F, Liu S, Liu P,et al.Bridging physical and virtual worlds:complex event processing for ZIGBEE data streams[C]//The 10th International Conference on Extending Database Technology (EDBT),Munich,Germany,2006:588-607.

BIOGRAPHY

SK SABIHA BEGUM received her B.Tech Degree in Electronics and Communications from JNTUK University of Technology, Andhra Pradesh in May 2008. Currently she is pursuing her M.Tech Degree in VLSI & ES from JNTUK University of Technology, Andhra Pradesh.

B. Uma Sankar received his B.Tech Degree in Electronics and Communications from JNTUH University of Technology, Andhra Pradesh in May 2007, and received his M.Tech Degree in Applied Electronics from ANNA University Chennai.