IoT in Traditional OS

Mr. S. Mohan[1], Dr.M. Varatharaj[2], Dr R. Murugadoss[3]
[1]*Assistant professor, Department of computer science, V.S.B. College of Engineering Technical Campus, Coimbatore, Tamil Nadu, INDIA.*
[2]*Associate professor, Department of EEE, V.S.B. College of Engineering Technical Campus, Coimbatore, Tamil Nadu, INDIA.*
[3]*Head of the department of AI&DS, V.S.B. College of Engineering Technical Campus, Coimbatore, Tamil Nadu, INDIA.*

***Abstract*—In Traditional OS is not sufficient suitable for IoT. because it is so many library constraints with its operations. But lake of constraint in IoT like windows, Linux and MAC OS all are having general- purpose OS In traditionally. this is power consuming and tight time constrain in IoT. In traditional OS not enough constrain in IoT device or machines. The IoT means IT+OT= IoT. IT means information technology; OT means operational technology are combined with create 'internet of things '(IoT). without intervention of human. because data transfer, process all are doing by constrain .so special OS needed to IoT with special constrain of the OS in traditional OS consider general – purpose OS, because lake of constrain in its. We will discuss about it.**

***Index Terms*—OS, IT, OT, IoT,**

I. INTRODUCTION

In traditional OS like these used in desk top and server are designed to run as powerful hardware and required significand memory and processing power. but IoT particularly low-end. Often have the limited resources in

IoT have the battery power simple task often performance by IoT devices. these are all process. of operating systems while all are having the connect of the devices or gadgets. In general purpose OS are so many libraries function (built-in) constraints. But not suitable for IoT gadgets in traditional OS are not ideal for the IoT device. These traditional OS are not generally optimized for the strict time requirements of many IoT applications. In IoT OS are designed to be light weight and required minimal resources. Many IoT as real-time features handling the to regards quickly to event and data streams, because it making them earn to display and manage resource constraint devices. So, we will discuss special OS for IoT (real-time OS)-RTOS or bare metal.

RTOS (real-time operating system):

It is popular open-source real – time operating systems used in wide range of IoT application .it design for small micro controller and used in various enable systems so many field RTOS used now autonomous medical and industrial application. because it is safety platform known it is for real-time performance.

Application of RTOS it is used in -aircraft control systems, autonomous driving system, medical decision, robotics, industrial autonomous and defense.

MICRO CONTROLLER:

That is integrated circuit. that circuit microprocessor along with memory and associated with the circuit and control. That all of the function of electronic device for specific task to control the particular systems. micro controller generally doesn't with operating systems but run with bootloader. that program lives in the put of the controller. bootloader is a firmware.

FIRMWARE:

The form of micro code embedded in to hardware devices to help them operating efficiency like camera, printer, router, scanner, television remote.it built in their memory function.

Firmware is the fundamental software that either device of function, communication and interact with the broader ecosystem. Its operation collecting and transmitting data. Firmware is layer between hardware and software system that control and monitoring IoT devices.

AN EMBEDDED SYSTEM IN IoT:

This is specialized computing systems that is power allow to the interact with the internet of each other. It is brain of IoT device functionality to sensor, actuator and connect.

II. MICRO PROCESSOR

That means small unit of computer that contain all the function of the control processing unit .it is also brain of the computer or electronic devices.it is small integrated circuit (a Chip). that performs all functionality task by following instruction .it think of it is a tiny computer on a chip. that control all process of the computer it connects of only central processing unit (CPU).

BARE METAL:

The bare metal is the programmable that is real time-operating system and traditional os represent different approaches to software development on embedded system bare metal it is real time scheduling capabilities. ideal for the high resources constraints devices and application with strict real-time requirements. Were directly hardware control in memory. The traditional os often broader range of features but with more activated. It is writing code run on directly on the hardware without os layer in between.

TRADITIONAL OS:

like as WINDOWS AND LINUX are designed for general -purpose computing it is wide range of features and capability it has many foot print significant proving overhead and showing response then less efficient resource utilization.

It has not designed for distributing level for balancing p [potentially carrying up predictable res [ponce time .it is application with complex requirement a need for a wide range of features and when real time performance is not permanent.

It is not a primary close, especially for resources-constraint device. IoT is particularly low-end one. Rely need real-time os (RTOS). because it is light weight OS. but traditional os is designed such a constraint and can be resource inedible. making them unsuitable for these devices.

ALTRANATE FOR THE TRADITIONAL OS:

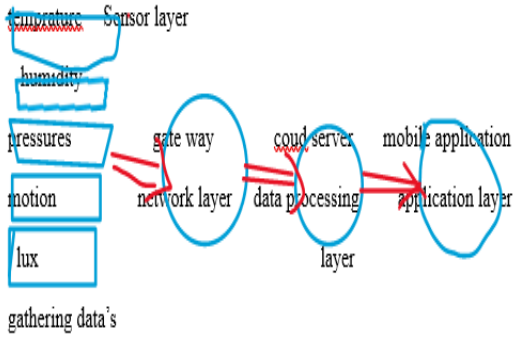
Light weight OS: RIOT<tiny and Contiki are light weight OS and low end IoT devices focus of minimal resource usages.

Embedded Linux: like ubuntu care and android things are also popular clear for specific IoT application and platforms more processing power.

OTHER OPTION.:

MICRO PYTHON, M bed and android things while Linux other traditional os are not absence for the IoT land scape then are often over shad over specialized as and RTOS designed unique requirement of the device the Chaice of as depends on specific application and constrain of the shareware.

IoT architecture:



IoT divided in to 4 different layers

1.Sensing layer:

2.Network layer

3.Data processing layer

4.Applicatiom layer

1. collecting data from different resources. Its inclusive sensor actuators, wired and wireless communication protocol connect these devices to the network layers

2.it is responsible for the processing communication and connecting devices and the IoT devices.

3.it refer the hardware and software components that are responsible for collecting, analyzing and integrating data from IoT devices. receive the raw data from the devices processing it further analyzing action, machine learning.

4. it is end-user for user friendly information's and functionality enable used to across control the IoT devices. because it is data visualization tool.

For instants the traditional OS to manage it sever network connecting and communication with the general server. it collects the data communication with the network and allow user control routing via the application.

What is different to use RTOS:

It is used to real time task management it making systems to mention, implements and design. it is

lower end program and higher hardware interaction might increase the complexity. because it is easy target to the cyber-attack, Leake sub tense inputs. Hacker can control the device remotely dependence and internet connection automation can replace the manual job. Reshaping the word renewable way, offering in parallel communication efficiency and innovations.

III. USE TRADITIONAL OS IN IoT

While possible to use traditional operating system like windows or mac OS in an IoT device it generally not the most of efficient or suitable approach for the most IoT application. IoT devices low end-once, limited resources (memory, power, processing) and unique requirements like low energy connection and real time processing traditional or often not optimized for these constraints resource and can be resource intensive. resource constraints eminent with the limited memory, processing power and battery life. In traditional general-purpose computing can be too large and resources intensive to in efficient and then device.

WHY SPECIAL IoT OS ARE PREPARED:
1.optimized resource constrain
2.real-time capability
3.security
An IoT OS allows to with communicate with cloud services across the global networks. Today connected world IoT. It transferring various industries, it refers vast network connected devices and sensors that shares, process and analysis data to enable smooth connected and automation.
Seamless connecting and verbal exchange of IoT gadgets in traditional OS and iOS have been not designed for the unique need and skill of IoT gadgets lake of memory constrains for low energy intakes separate working device specially design for IoT devices allowed for optimum performance and optimization.
When it comes IoT connecting to most important factor to consider without communication between the various components of IoT ecosystem (sensor, computing engine, data hub….) no proper process can be executed IoT device are connected through radio waves, blue tooth, and WIFI, we can use various protocol of internet connectivity across IoT eco system and industries…
Data used to important business insight and drive the important business decision. we develop machine learning, /deep learning data to obtained valuable insights.

IV. CONCLUSION

will discuss about why need to special OS for IoT. Because already so many traditional OS each and every OS separately field decision made by human which this paper, we field id is which OS is best fit. but in IoT without human intervention of devices is not human-to-human or human-to-machine because IoT means machine-to-machine specially allow to execute some specific OS. That OS are optimized of device eco systems all are combined with the traditional OS but lake of constraints so needs to special for IoT.in traditional OS vast purpose and not especially one .so many library constraints -even though not sufficient to the IoT eco systems.

REFERENCES

[1] W. Dong, C. Chen, X. Liu, and J. Bu, “Providing os support for wireless sensor networks: challenges and approaches,” Communications Surveys & Tutorials, IEEE, vol. 12, no. 4, pp. 519–530, 2010.
[2] L. Saraswat and P. S. Yadav, “A comparative analysis of wireless sensor network operating systems,” International Journal of Engineering and Technoscience, vol. 1, no. 1, pp. 41–47, 2010.
[3] no. 13, pp. 2521–2533, 2006. [29] B. Hughes, R. Meier, R. Cunningham, and V. Cahill, “Towards realtime middleware for vehicular ad hoc networks,” in Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, ser. VANET ’04. New York, NY, USA: ACM, 2004, pp. 95–96. [Online]. Available: http://doi.acm.org/10.1145/1023875.1023894
[4] J.-H. Hoepman and B. Jacobs, “Increased security through open source,” Commun. ACM, vol. 50, no. 1, pp. 79–83, Jan. 2007. [Online]. Available: http://doi.acm.org/10.1145/1188913.1188921

[5] A. Castellani, G. Ministeri, M. Rotoloni, L. Vangelista, and M. Zorzi, “Interoperable and globally interconnected Smart Grid using IPv6 and 6LoWPAN,” in Communications (ICC), 2012 IEEE International Conference on, June 2012, pp. 6473–6478.

[6] J. Eriksson, A. Dunkels, N. Finne, F. Osterlind, and T. Voigt, “Mspsiman extensible simulator for msp430-equipped sensor boards,” in Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session, 2007, p. 27.

[7] Emul8. [Online]. Available: http://emul8.org/ [39] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, “Network simulations with the ns-3 simulator,” SIGCOMM demonstration, vol. 14, 2008.

[8] A. Dunkels, B. Gr¨ onvall, and T. Voigt, “Contiki- a lightweight and flexible operating system for tiny networked sensors.” in LCN. IEEE Computer Society, 2004, pp. 455–462. [Online]. Available: http://dblp.uni-trier.de/db/conf/lcn/lcn2004.html#DunkelsGV04

[9] E. Baccelli, O. Hahm, M. G¨unes, M. W¨ahlisch, and T. C. Schmidt, “RIOT OS: Towards an OS for the Internet of Things,” in 32nd IEEE INFOCOM, IEEE. Turin, Italy: IEEE, 2013.

[10] E. Baccelli, O. Hahm, H. Petersen, and K. Schleiser, “RIOT and the Evolution of IoT Operating Systems and Applications,” ERCIM News, vol. 2015, no. 101, 2015. [Online]. Available: http://ercim-news.ercim.eu/en101/special/ riot-and-the-evolution-of-iot-operating-systems-and-applications

[11] H. Will, K. Schleiser, and J. H. Schiller, “A real-time kernel for wireless sensor networks employed in rescue scenarios,” in IEEE LCN, 2009.

[12] “CCN Lite: Lightweight implementation of the Content Centric Networking protocol,” 2014. [Online]. Available: http://ccn-lite.net

[13] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, “TinyOS: An Operating System for Sensor Networks,” in Ambient Intelligence, W. Weber, J. M. Rabaey, and E. Aarts, Eds. Berlin Heidelberg: Springer-Verlag, 2005, ch. 7, pp. 115–148. [Online]. Available: http://dx.doi.org/10.1007/3-540-27139-2 7 [51] P. Levis,