ICCOTWT 2018

12 & 13 July 2018

Michigan,
United States of America

ISBN: 978-93-88122-00-9

Proceedings of the Third International Conference on Cloud of Things and Wearable Technologies 2018

A S D F

SUBRA GANESAN

International Conference on Cloud of Things and Wearable Technologies 2018

ICCOTWT 2018

International Conference on Cloud of Things and Wearable Technologies 2018

Volume 1

By
ASDF, North America

Financially Sponsored By
Association of Scientists, Developers and Faculties, India

Multiple Areas

12-13, July 2018
Michigan,
United States of America

Editor-in-Chief
Subramaniam Ganesan

Editors:

Daniel James, Kokula Krishna Hari Kunasekaran and Saikishore Elangovan

Published by

Association of Scientists, Developers and Faculties

Address: RMZ Millennia Business Park, Campus 4B, Phase II, 6th Floor, No. 143, Dr. MGR Salai, Kandanchavady, Perungudi, Chennai – 600 096, India.
Email: admin@asdf.org.in || www.asdf.org.in

International Conference on Cloud of Things and Wearable Technologies (ICCOTWT 2018)

VOLUME 1

Editor-in-Chief: **Subramaniam Ganesan**
Editors: **Daniel James, Kokula Krishna Hari Kunasekaran and Saikishore Elangovan**

ISBN-13: 978-93-88122-00-9
ISBN-10: 93-88122-00-3

TECHNICAL REVIEWERS

- Abdelrahman Elewah, Benha Faculty of Engineering, Egypt
- Abdulrazak Mohamed, School of Planning and Architecture, Vijayawada, India
- Abhishek Bajpai, Rajkiya Engineering College, India
- Achal Garg, Keppel Offshore and Marine, India
- Aede Hatib Musta'amal, Universiti Teknologi Malaysia, Malaysia
- Ahmed Mehany, Beni-suef Univeristy, Egypt
- Ahmed Mohamed, Beni Suef University, Egypt
- Ahmed Mohammed Kamaruddeen, University College of Technology Sarawak, Malaysia
- Alaa El-hazek, Faculty of Engineering At Shoubra, Benha University, Egypt
- Alagammal Mohan, Mepco Schlenk Engineering College, Sivakasi, India
- Ali Berkol, Baskent University, Turkey
- Allin Joe David, Kumaraguru College of Technology, India
- Ambavarm Vijaya Bhaskar Reddy, Universiti Teknologi Petronas, Malaysia
- Ambika Pathy, Galgotias College of Engineering College And Technology, India
- Ammar Jreisat, Al Ain University of Science and Technology, United Arab Emirates
- Amol Potgantwar, Sandip Institute of Technology & Research Centre Nasik, India
- Amr Helmy, The British University in Egypt, Egypt
- Anand Nayyar, Duy Tan University, Da Nang, Vietnam, Viet Nam
- Anbuoli Parthasarathy, Anna University, India
- Anil Dubey, Abes Engineering College Ghaziabad, India
- Aniruddha Thuse, Jgi's Jain College of Mba & Mca, Belagavi, Karnataka, India
- Anitha Natarajan, Kongu Engineering College, India
- Ankur Bist, Kiet Ghaziabad, India
- Anshul Garg, Taylor's University, Malaysia

- Appavu Alias Balamurugan Subramanian, E.G.S Pillay Engineering College, India
- Aravind C.k., Mepco Schlenk Engineering College, Sivakasi, India
- Ariffin Abdul Mutalib, Universiti Utara Malaysia, Malaysia
- Arockia Xavier Annie Rayan, Anna University, India
- Arshad Mansoor, Civil Defence, Riaydh, Saudi Arabia
- Arul Teen, University College of Engineering Nagercoil, India
- Arumugam Raman, Universiti Utara Malaysia, Malaysia
- Arun M R, SXCCE, Anna University, India
- Arun Sharma, Indira Gandhi Delhi Technical University for Women, Delhi, India
- Arun Pandian Jaya Pandian, M.A.M. College of Engineering and Technology, India
- Arupaddiyappan Sivasubramanian, V I T University, India
- Ashokkumar Nagarajan, Sree Vidyanikethan Engineering College, India
- Asokan Ramasamy, Kongunadu College of Engineering and Technology, India
- Ayyanadar Arunachalam, Indian Council of Agricultural Research, India
- Azwa Abdul Aziz, Universiti Sultan Zainal Abidin, Malaysia
- Bala Venkata Subrahmanyam, TKREC, India
- Balachandar Krishnamoorthy, Sastra Deemed to be University, India
- Balaji Kalaiarasu, Amrita Vishwa Vidyapeetham, Coimbatore, India
- Balakrishnan Kandasamy, Karpaga Vinayaga College of Engineering and Technology, India
- Balakrishnan Subramanian, Sri Krishna College of Engineering and Technology, Coimbatore, India
- Balamuralitharan Sundarappan, SRM IST, India
- Balamurugan N M, Sri Venkateswara College of Engineering, India
- Balamurugan Sivaramakrishnan, Coimbatore Institute of Technology, India

- Bhanu Prakash Kolla, Koneru Lakshmaiah Education Foundation, India
- Bhavani Anand, Mepco Schlenk Engineering College, India
- Bhavna Ambudkar, Dr. D.y Patil Institute of Technology, Pimpri, India
- C V Guru Rao Rajagopal Rao, Sr Engineering College, India
- Carlo Inovero, Polytechnic University of The Philippines, Philippines
- Chandrasekaran Muthial Chetty, Government College of Engineering, Bargur, Tamil Nadu, India
- Charles Weeraratna, Lanka Rainwater Harvesting Forum, Sri Lanka
- Chitra Krishnan, VIT Chennai, India
- Christopher Hill, The British University in Dubai, United Arab Emirates
- Dafni Rose, St. Joseph's Institute of Technology, India
- David Rathnaraj Jebamani, Sri Ramakrishna Engineering College, India
- David Wilson Devarajan, English/ Karunya Institute of Technology & Sciences, India
- Deepa Dhanaskodi, Bannari Amman Institute of Technology, India
- Deepa Jose, KCG College of Technology, India
- Deepa Rani T, India
- Deepali Sawai, Atss's Institute of Industrial and Computer Management and Research (IICMR), India
- Delampady Narasimha, Indian Institute of Technology Dharwad, India
- Devendra Kumar Rangasamy Natarajan, Sri Ramakrishna Institute of Technology, India
- Dharma Raj Cheruku, GITAM, India
- Diaa Salama, Faculty Of Computers and Informatics, Benha University, Egypt
- Dinkar Nandwana, Asm America Inc, United States
- Dishek Mankad, Shri P.K.M. College of Technology & B.Ed., India
- Djilali Idoughi, University A. Mira of Bejaia, Algeria

- Hariharan Vaggeeram, Kongu Engineering College, Erode, India
- Harikiran Jonnadula, Shri Vishnu Engineering College for Women, India
- Harikrishna Kumar Mohan Kumar, Kongu Engineering College, India
- Harikrishnan Santhanam, Adhi College of Engineering & Technology, India
- Hemanth Chandran, Vellore Institute of Technology Chennai, India
- Jagannath Mohan, Vellore Institute of Technology (VIT) Chennai, India
- Jagathy Raj V. P., Cochin University of Science and Technology, India
- Javier Dario Fernandez Ledesma, University Pontificia Bolivariana, Colombia
- Jayshree Sanjay Kumar Soni, JNVU, Jodhpur, India
- Jitendra Agrawal, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India
- Jitendra Singh, SRM University, India
- Jyothi Badnal, Chaitanya Bharati Institute of Technology, India
- Kala Iyapillai, SNS College of Engineering, India
- Kalaivani Anbarasan, Saveetha School of Engineering, India
- Kalpana Murugan, Kalasalingam Academy of Research And Education, India
- Kannan Gopal Radhakrishnan, PSNA College of Engineering and Technology, India
- Kannan Kilavan Packiam, Bannari Amman Institute of Technology, India
- Kareem Kamal A.ghany, Beni-suef University, Egypt
- Karthi Kumar Ramamoorthy, Kumaraguru College of Technology, India
- Karthik Subburathinam, SNS College of Technology, India
- Karthikeyan Jayaraman, Mangayarkarasi College of Engineering, Madurai, India
- Karthikeyan Parthasarathy, Kongu Engineering College, India
- Kavitha Kanagaraj, Kumaraguru College of Technology, India
- Kavitha R V R Venkatachalapathi, PES University, India
- Kiran Kumar Kudumula, Rajeev Gandhi University, India

- Marcia Pinheiro, IICSE University De, Australia
- Marikkannan Mariappan, Institute of Road and Transport Technology, India
- Marimuthu Nachimuthu, Nandha Engineering College, India
- Martin Sagayam Kulandairaj, Karunya Institute of Technology and Sciences, India
- Maslin Masrom, Universiti Teknologi Malaysia, Malaysia
- Mathivannan Jaganathan, Universiti Utara Malaysia, Malaysia
- Mathiyalagan Palaniappan, Sri Ramakrishna Engineering College, India
- Md. Haider Ali Biswas, Khulna University, Bangladesh
- Min Nie, Stevens Institute of Technology, United States
- Miroslav Mateev, American University in The Emirates, United Arab Emirates
- Missak Swarup Raju, GITAM Deemed to be University, India
- Mohamed Abd El-aal, Kanazawa University, Egypt
- Mohamed Abdo, Assiut University, Egypt
- Mohamed Ali, King Saud University, Saudi Arabia
- Mohamed Eid Shenana, Benha University, Egypt
- Mohamed Nayel, Assiut University, Egypt
- Mohamed Saleh, Harbin Institute of Technology, Egypt
- Mohamed Waly, Majmaah University, Egypt
- Mohammad Arif Kamal, Aligarh Muslim University, India
- Mohammed Ali Hussain, Kl University, India
- Mohammed Saber, Faculty of Engineering- Fayoum University, Egypt
- Mohd Helmy Abd Wahab, Universiti Tun Hussein Onn Malaysia, Malaysia
- Murali Muniraj, Sona College of Technology, India
- Murulidhar K S Shivaiah, K S I T, India
- Muthupandian Thangasamy, PSNA College of Engineering and Technology, India

- Rafat Amin, Beni Suef University, Faculty of Science, Physics Department, Egypt
- Ragupathy Rengaswamy, Annamalai University, India
- Raj Mohan Radhakrishnan, Sastra Deemed University, India
- Raja Suzana Raja Kasim, Universiti Malaysia Kelantan, Malaysia
- Rajarajan Gopal, Hindustan Institute of Technology & Science, India
- Rajesh Keshavrao Deshmukh, S.s.i.p.m.t., Raipur, Chhattisgarh, India
- Rajesh Kanna Rajendran, Dr. N.G.P Arts and Science College, India
- Rajiv Kumar, Punjab Institute of Management and Technology, India
- Rajiv Selvam, Manipal University, India
- Rama Sree Sripada, Aditya Engineering College, India
- Ramasamy Pachaiyappan, Sri Balaji Chockalingam Engineering College, India
- Ramayah Thurasamy, School of Management/universiti Sains Malaysia, Malaysia
- Rames Panda, CSIR-CLRI, India
- Ramesh Balasubramanian, St. Joseph's Institute of Technology, India
- Ramesh Sengottuvelu, KCG College of Technology, India
- Rampradheep Gobi Subburaj, Kongu Engineering College, India
- Ramu Nagarajapillai, Annamalai University, India
- Rana Alabdan, Majmaah University, Saudi Arabia
- Randa Alarousy, National Research Center, Egypt
- Ravi Gulati, Veer Narmad South Gujarat University, India
- Ravikumar D.v., Adithya Institute of Technology, India
- Raviraj Pandian, GSSS Institute of Engineering & Technology for Women, India
- Rehab Nady, Faculty of Science - Beni-suef University, Egypt
- Reyhan Alhas, Mcmp/lmu Munchen, Germany
- Reza Gharoie Ahangar, University of North Texas, United States

- Roselina Sallehuddin, Universiti Teknologi Malaysia, Malaysia
- Ruchi Tuli, Jubail University College, Saudi Arabia
- Rupesh Dubey, IPS Academy Institute of Engineering & Science Indore, India
- S. Balamurugan, Quants Is & Cs, India
- S.V. kogilavani Shanmugavadivel, Kongu Engineering College, India
- Sabanayagam Angamuthu, Karpagam Institute of Technology, India
- Sadhik Basha J, International Maritime College Oman, Oman
- Sahar Badawy, Bue, Egypt
- Saikishore Elangovan, ,
- Saleh Altayyar, King Saud University, Saudi Arabia
- Sallehuddin Muhamad, Universiti Teknologi Malaysia, Malaysia
- Sandeep Bhat, SIT, Mangaluru, India
- Sangeetha Rengachary Gopalan, VIT University, India
- Sanjay Ghandhyambhai, LDRP-ITR, India
- Sanjay Kumbhar, Rajarambapu Institute of Technology, India
- Sanjay Pande, GM Institute of Technology, India
- Santhosh Kumar Balan, GMR Institute of Technology, India
- Sasikala Ramachandran, Kongu Engineering College, India
- Sasikala Senthamarai Kannan, Paavai Engineering College, India
- Sathish Babu, Department of Electronics and Instrumentation, India
- Sathish Gandhi Chinnasamy, University College of Engineering Nagercoil, India
- Sathish Kumar Nagarajan, Sri Ramakrishna Engineering College, Coimbatore, India
- Satish Sajja, V R Siddhartha Engineering College, India
- Sayed Gomaa, Al-azhar University and British University, Egypt
- Selvakumar Muthusamy, Sri Venkateswara College of Engineering, India

- Selvaperumal Sundara, Syed Ammal Engineering College, India
- Selvi Rajendran, KCG College of Technology, India
- Selvi Shanmugam, Institute of Road and Transport Technology, India
- Senthilkumar Kandasamy, Kongu Engineering College, India
- Senthilnathan Nattuthurai, Kongu Engineering College, India
- Shamsiah Banu Mohamad Hanefar, University of Nottingham Malaysia, Malaysia
- Shanmugapriyan Thiagarajan, ASDF International, India
- Shanmugasundaram O.l Lakshmanan, K.S. Rangasamy College of Technology, India
- Shanthakumari Raju, Kongu Engineering College, India
- Shanthi Radhakrishnan, Kumaraguru College of Technology, India
- Shekhar Ramaswamy, Alliance University, India
- Shidaling Matteppanavar, JNCASR Bangalore, India
- Shikha Maheshwari, JECRC, Jaipur, India
- Sivakumar Vaithilingam, Ramco Institute of Technology, India
- Sivaprakash Chokkalingam, Sri Sairam College of Engineering, India
- Sivaraja Muthusamy, N.S.N. College of Engineering and Technology, India
- Soonmin Drho, Inti International University, Malaysia
- Sreenivasa Rao Ijjada, GITAM University, India
- Sri Devi Ravana, University of Malaya, Malaysia
- Srinivasan Natrajan, Kongu Engineering College, India
- Subramaniam Ganesan, Oakland University, United States
- Subramanian Krishnamurthy, IGNOU/IIT, India
- Sudhakar Radhakrishnan, Dr. Mahalingam College of Engineering and Technology, India
- Sukamal Sanhita Deb, IGNOU, India
- Sukumar Ponnusamy, Nandha Engineering College (Autonomous), India

- Sundar Ganesh Chappani Sankaran, PSG College of Technology, India
- Sunita Daniel, Amity University Haryana, India
- Tamilarasi Angamuthu, Kongu Engineering College, India
- Tamilsalvi Mari, Taylor's University, Malaysia
- Tamilselvan K.s., Kongu Engineering College, India
- Tamizhselvan Perumal, Tamil Nadu Institute of Urban Studies, India
- Thamizhmaran Krishnamoorthy, Annamalai University, India
- Thangagiri Baskaran, Mepco Schlenk Engineerng College, India
- Thangamani Murugesan, Kongu Engineering College, India
- Thangavel Murugan, Thiagarajar College of Engineering, India
- Thangavel Subbaiyan, National Institute of Technology Puducherry, India
- Thenmozhi Rayan, Dayanandasagar College of Engineering, India
- Thilagamani Sathasivam, M. Kumarasamy College of Engineering (Autonomous), India
- Thirugnanam Gurunathan, Annamalai University, India
- Udhayakumari Duraisamy, Rajalakshmi Engineering College, India
- Uthirakumar Periyayya, Sona College of Technology, India
- Vadlamudi Parthasarathi Naidu, CSIR-national Aerospace Laboratories, India
- Valmiki Ramakrishna, Tumkur University, India
- Vasuki Arumugam, Kumaraguru College of Technology, India
- Vasunun Chumchua, Mahidol University, Thailand
- Veeraswamy Ammisetty, St.Ann's College of Engineering & Technology, India
- Venkata Subba Reddy Imma Reddy, GITAM (Deemed to be University), India
- Venkatanarayanan P.S., Hindustan Institute of Technology and Science, India
- Vijay Gupta, IITM, India
- Vijaya Deshmukh, National Institute of Fashion Technology, India

- Vijaya Kumar Y, Sri Sairam College of Engineering, India
- Vijaya Kumari Valsalam, Er.perumal Manimekalai Engineering College, India
- Vijayachitra Senniapppan, Kongu Engineering College, Perundurai, India
- Vijayan Gurumurthy Iyer, Koneru Lakshmaiah Education Foundation (KLEF), India
- Vijayaraja Kengaiah, KCG College of Technology, India
- Vikrant Bhateja, SRMGPC, Lucknow, U.p., India
- Vimala Vishwanath Achari, Avinashilingam Institute For Home Science and Higher Education for Women, Coimbatore, India
- Vinod Kapse, Gyan Ganga Institute of Technology and Sciences, Jabalpur, India
- Vipin Jain, Teerthanker Mahaveer University, India
- Visweswara Rao Pasupuleti, Universiti Malaysia Kelantan, Malaysia
- Vivekanandan Nellaiappan, Central Water and Power Research Station, India
- Vo Ngoc Phu, Duy Tan University, Viet Nam
- Walid Abdalhalim, Beni-suef University, Egypt
- Wan Hussain Wan Ishak, Universiti Utara Malaysia, Malaysia
- Xuan Wang, Utrgv, United States
- Yerra Rama Mohan Rao, Dr. Paul's Engineering College, India
- Yousef Farhaoui, Moulay Ismail University, Morocco
- Yousef Okour, Majmaah University, Saudi Arabia
- Yudi Fernando, Faculty of Industrial Management, Universiti Malaysia Pahang, Malaysia
- Zahira Rahiman, Tagore Engineering College, India
- Zahurin Samad, Universiti Sains Malaysia, Malaysia

Table of Content

Volume	01	ISBN	978-93-88122-00-9
Month	July	Year	2018

International Conference on Cloud of Things and Wearable Technologies 2018

Title & Authors	Pages
ABS System – Model in Loop Test (V & V) in Simulink by *Chandana Munireddy, Kaushik Avani Dixit Ganesh, Sreehari Mogulluri and Vidyasekhar Potluri*	pp01
Design and Validation of Digital Driving System for Semi-Autonomous Vehicle *by Bhanu Deepthi Sree Uddagiri, Kushagra Gupta, Saranya Sowrirajan and Samuel Ogunyemi*	pp01
Fault Tolerance of Multiple Processors with Divisible Loads Theory *by Yung-Hsuan Chen and Subra Ganesan*	pp02
Model-based Software Engineering Process *by Swathi Vadde and Subramaniam Ganesan*	pp02
GPU And Parallel Processing *by Akanksha Fadnavis and Yogini Nemade*	pp03
Hand Tracking and Gesture Recognition System for Human Computer Interaction *by Raphy Yaldo and Rita Kashat*	pp03
Model-Based Design, Validation and Verification of Automotive Embedded Systems—Infotainment *by Dingwang Wang*	pp04
Parallel Computing using Multicore Hardware and MATLAB *by Chandana Munireddy and Kaushik Dixit A G*	pp04
Verification and Validation of Shift-By-Wire Actuators *by Vinod Shankershetty*	pp05

Snoop Based Multiprocessor Design *by Arjun Musham, Khushboo Manek and Sai Priyanka Palla*	pp05
Image Filters Using CUDA on NVIDIA *by Tedros Berhane and Namit Nagpal*	pp06 – pp11
User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators *by Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh*	pp12 – pp20

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9
Website	iccotwt.com
Received	14 – May– 2018
Article ID	ICCOTWT001

VOL	01
eMail	iccotwt@asdf.res.in
Accepted	02 - June – 2018
eAID	ICCOTWT.2018.001

ABS System – Model in Loop Test (V & V) in Simulink

Chandana Munireddy[1], Kaushik Avani Dixit Ganesh[2], Sreehari Mogulluri[3] and Vidyasekhar Potluri[4]

Abstract— *This Mini project is to show our understanding on Verification and Validation of Antilock braking system in which we took the working principle model of ABS designed in MATLAB SIMULINK and applied the verification and validation methods by simulating different inputs of ABS System and monitoring its outputs against the expected results.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9
Website	iccotwt.com
Received	20 – May– 2018
Article ID	ICCOTWT002

VOL	01
eMail	iccotwt@asdf.res.in
Accepted	15 - June – 2018
eAID	ICCOTWT.2018.002

Design and Validation of Digital Driving System for Semi-Autonomous Vehicle

Bhanu Deepthi Sree Uddagiri[1], Kushagra Gupta[2], Saranya Sowrirajan[3] and Samuel Ogunyemi[4]

Abstract— *Nowadays, automobiles are being developed by electrical parts for efficient operation. This project will present the development and implementation of a digital driving system with park aid and wiper control for a semi-autonomous vehicle to improve driver vehicle interface. The design utilizes two external sensors connected to Dragon 12 Board for rain and relative distance sensing between the car and the obstacle, servo motor for wiper control and LEDS/buzzer in the board for achieving park aid. First step is to define the requirements of the system for the above said features. According to our software design the park aid feature is achieved through Modelling in MATLAB and the Wiper control is achieved through programming the board using embedded C in CodeWarrior. Validation of the software components is done through unit test and Model-In-Loop test. The overall system function is verified against the system requirements using Hardware-In-Loop testing.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9
Website	iccotwt.com
Received	05 – August– 2018
Article ID	ICCOTWT003

VOL	01
eMail	iccotwt@asdf.res.in
Accepted	05 - June – 2018
eAID	ICCOTWT.2018.003

Fault Tolerance of Multiple Processors with Divisible Loads Theory

Yung-Hsuan Chen[1] and Subramaniam Ganesan[2]

[1,2]Oakland University

Abstract— *Today the scientific computation problems requiring intense problem-solving capabilities for problems arising from complex research and industrial application has driven in all global institution and industry segments the need for dynamic collaboration of many ubiquitous computing resources to be able to work together. The problem of minimizing the processing time of extensive processing loads originated from various sources presents a great challenge that, if successfully met, could foster a range of new creative applications. Inspired by this challenge, we sought to apply divisible load theory to the problem of grid computing involving multiple sources connected to multiple sinks. So far research in this area includes where tasks arrive according to a basic stochastic process to multiple nodes and presents a first step technique for scheduling divisible loads from multiple sources to multiple sinks, with and without buffer capacity constraints. The increasing need for multiprocessing systems and data-intensive computing has created a need for efficient scheduling of computing loads, especially parallel loads that are divisible among processors and links. During the past decade, divisible load theory has emerged as a powerful tool for modeling data-intensive computational problems. The purpose of this research is to obtain a closed form solution for the finish time, taking into consideration the adverse effect of the fault Single Installment with FIFO (First In, First Out) and LIFO (Last In, First Out) result allocation on a homogenous system. The system under consideration in this research utilizes job scheduling of a Divisible Load scheme that entails distributing arbitrarily divisible computational loads amongst eligible processors within a bus based distributed computing environment. Including, the aspect of Single Installment scheme of Divisible Load Theory (DLT), along with the Results Collection Phase. In this distributed system, there is a primary processor and a backup processor. All the processors periodically checkpoint their results on the backup processor. If any processor fails during the task execution, the backup processor takes over the failed process by rolling over to the time of the last check pointing. The study assumes that only one processor faults during a lifetime of single task execution. It is believed that the outcomes of this research may be beneficial to embedded system designers on how to approach fault-tolerant methods and performance improvement for Load distribution.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9
Website	iccotwt.com
Received	20 – May– 2018
Article ID	ICCOTWT004

VOL	01
eMail	iccotwt@asdf.res.in
Accepted	05 - June – 2018
eAID	ICCOTWT.2018.004

Model-based Software Engineering Process

Swathi Vadde[1] and Subramaniam Ganesan[2]

[1,2]Oakland University

Abstract— *The increasing complexity of ECU (Electronic Control Unit) in Automotive has created new challenges in ensuring the need of higher quality design, reliability, competitive cost, customer satisfaction. This report describes the process followed by Software Engineers during Model based Algorithm Development in the automotive industry. This report discusses also on the necessary tools used to achieve this.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9
Website	iccotwt.com
Received	08 – May– 2018
Article ID	ICCOTWT005

VOL	01
eMail	iccotwt@asdf.res.in
Accepted	13 - June – 2018
eAID	ICCOTWT.2018.005

GPU And Parallel Processing

Akanksha Fadnavis[1] and Yogini Nemade[2]
[1,2]Oakland University

Abstract— *Parallelism is the future of computing and is been used in many domains such as high-performance computing (HPC), graphic accelerators, many large control and embedded systems, automotive with great success. Graphics Processing Unit (GPU) is a highly effective utilization of parallel processing which provides a vast number of simple, data-parallel, deeply multithreaded cores and high memory bandwidths. GPUs were originally hardware blocks optimized for a small set of graphics operations. As demand arose for more flexibility, GPUs became ever more programmable. Early approaches to computing on GPUs cast computations into a graphics framework, allocating buffers /arrays and writing shades /kernel functions. Several research projects looked at designing languages to simplify this task; in late 2006, NVIDIA introduced its CUDA architecture and tools to make data parallel computing on a GPU more straightforward. Not surprisingly, the data parallel features of CUDA map well to the data parallelism available on the NVIDIA GPUs. GPU architectures are vastly programmable, they offer high throughput and data intensive operations.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9
Website	iccotwt.com
Received	03 – May– 2018
Article ID	ICCOTWT006

VOL	01
eMail	iccotwt@asdf.res.in
Accepted	20 - June – 2018
eAID	ICCOTWT.2018.006

Hand Tracking and Gesture Recognition System for Human Computer Interaction

Raphy Yaldo[1] and Rita Kashat[2]

Abstract— *Human and computer interaction is used in our daily lives, one instance is the use of mice and keyboards or touchscreen as a method to interact with computers. As new and improved technology develops so do new machine-man interfaces as a result. In this project we are using minimally priced hardware to achieve finger and hand tracking. We use a simple system but it proves highly efficient in allowing us to track hand movement while still enabling us to ignore the effect that complex backgrounds, which may include movement, have on movement tracking. This system enables us to translate the detected hand motions and gestures to be used as multiple function inputs to interface with our applications, as well as provide the necessary outputs needed for hand motions or gestures. This project will not rely on simple mechanical devices, such as the standard mouse and keyboard, but instead on hand and finger tracking instead.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9
Website	iccotwt.com
Received	14 – May– 2018
Article ID	ICCOTWT007

VOL	01
eMail	iccotwt@asdf.res.in
Accepted	02 - June – 2018
eAID	ICCOTWT.2018.007

Model-Based Design, Validation and Verification of Automotive Embedded Systems—Infotainment

Dingwang Wang

Abstract— *Infotainment and multimedia applications are an increasingly important factor when consumer deciding to purchasing a new car. The current technology for automobile infotainment systems is the Media Oriented Systems Transport (MOST) network. This MOST protocol can allow passengers enjoy sophisticated infotainment applications through the high-speed networking. However, the increasing complexity and interface of automotive infotainment functions and the infotainment related applications are changing with each passing day bring a big challenge for automobile design and development. So, we need to develop fast to keep up with the pace of development. The quality of these systems must be ensured as the functionality constantly grows while the software development costs must be driven down continuously. One approach that has been proven successfully here is model-based software design, validation and verification development.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9
Website	iccotwt.com
Received	20 – May– 2018
Article ID	ICCOTWT008

VOL	01
eMail	iccotwt@asdf.res.in
Accepted	05 - June – 2018
eAID	ICCOTWT.2018.008

Parallel Computing using Multicore Hardware and MATLAB

Chandana Munireddy[1] and Kaushik Dixit A G[2]

[1,2]Oakland University

Abstract— *The parallel computing project is a two-week project. Its overall goal is to develop a parallel computing environment for the programs and analyse its behaviour in software as well as hardware. The main objective is to reduce the time required for the execution of the programs indeed reducing the cost. To achieve this overall goal, several key objectives are achieved.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9	**VOL**	01
Website	iccotwt.com	**eMail**	iccotwt@asdf.res.in
Received	28 – August– 2018	**Accepted**	02 - June – 2018
Article ID	ICCOTWT009	**eAID**	ICCOTWT.2018.009

Verification and Validation of Shift-By-Wire Actuators

Vinod Shankershetty[1]
[1]Oakland University

Abstract— *The Shift-By-Wire (SBW) actuator system converts mechanically actuated automatic transmissions into electronically shifted transmissions. The actuator control module mounts directly to the transmission shift shaft. The actuator incorporates an electric motor and gearbox, along with a Hall-effect sensor for position feedback. Verification and Validation of Shift by wire actuator is completed as a project phase since this activity is independent of development, focusing and mapped on the right side of the V-model as per the course ECE-5734 Embedded systems verification and validation.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9	**VOL**	01
Website	iccotwt.com	**eMail**	iccotwt@asdf.res.in
Received	20 – May– 2018	**Accepted**	01 - June – 2018
Article ID	ICCOTWT010	**eAID**	ICCOTWT.2018.010

Snoop Based Multiprocessor Design

Arjun Musham[1], **Khushboo Manek**[2] **and Sai Priyanka Palla**[3]
[1,2,3]Oakland University

Abstract— *Multi core architectures have become popular in mobile SoCs; for CPU's as well as for mobile GPUs. With the overview of OpenCL for mobile GPU architecture, the SoCs are able to become more influential than before. All this was possible because the programs that were previously performed on the CPU are now being performed on the GPU at much faster rate. Along with this the need for cache coherence protocols has also been introduced. In any multicore system, snoop-based cache coherence protocols2 characteristically tend to have wide coherence traffic on the bus. All this traffic leads to tag lookups in remote data caches. In order to solve cache coherence problem how the snoopy based protocols can be used is well explained in this report. Along with the working of snoopy protocol, what are the advantages and disadvantages of having snoop-based designs are illustrated in the report. Basic idea behind this report is to explain what can be the design issues when implementing snoopy protocols in single level cache and multilevel caches and what are the solutions to avoid the problems. A simple method to support snoop cache coherence it to use it to transfer snoop messages. But with implementation of this arises the problem of having longer response time in case of snoop operations. In order to tell this problem, the report suggests Flexible Snooping algorithms consisting of forward and filter snoop algorithms.*

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9	**VOL**	01
Website	iccotwt.com	**eMail**	iccotwt@asdf.res.in
Received	14 – May– 2018	**Accepted**	02 - June – 2018
Article ID	ICCOTWT011	**eAID**	ICCOTWT.2018.011

Image Filters Using CUDA on NVIDIA

Tedros Berhane[1] and Namit Nagpal[2]

[1,2]School of Engineering and Computer Science, Oakland University, Rochester, Michigan, USA

Abstract— *This paper explains our team project on GPU Accelerated Parallel programing using CUDA and OpenCV open source available development tool and was completed at Oakland University, Rochester, MI Engineering labs. The project compared two commonly known as image filters, Median and Bilateral. The two filters were also compared with basic Gaussian filter, which has normal distribution, the team applied sorting algorithm methods in CUDA programming to perform parallel computing and optimization*

Keywords: Computer Vision (CV), OpenCV, CUDA, Median Filter, Bilateral Filter, Algorithm, Benchmarks, Shared Memory, Results.

INTRODUCTION

The document is final project report on GPU Accelerated Computing, in which two graduate students at Oakland University developed optimization and speed up results using CUDA and OpenCV. Parallel computing in this project has been achieved using CUDA programing and was perform in NVIDIA computing platforms environment. Parallel computing techniques recently has been practiced in the academia and industries to solve complex problems through thread and kernel launches. NVIDIA has Computer Vision (CV) open source and customized libraries where user can custom with their application from facial recognitions, contour of the physical image appearances, or smoothing blur image outcome [4].

To achieve this result, we accessed the wide-ranging of memory management, and parallelly optimization techniques of CUDA, the concepts and approaches of shared memory was used. In addition, thread scheduling by using tiling approaches and Kernel launches from CPU was enhanced as per performance results shows in next sections.

The image filtering was reached using Central Processing Unit (CPU) and Graphical Processing Unit (GPU) in OpenCV to compare and see the speed performance related to serial and parallel programming methods. In the program code, higher programming languages C/++ that are oftentimes compatible with CUDA and OpenCV were used to run the algorithms developed by the team. In this case, C++ was used in filtering images. Most importantly OpenCV Application Peripherals Interfaces (API) and struct class type of data organization approach was precisely practice in importing image matrix and outputting data results from OpenCV. Over all, the Massive parallel programming and computing procedure for median and bilateral filters differs on the pseudo code of the functions, as the result different blurring intensity has been displayed. The following four images portrays median and Bilateral filtering. Two figures below are sample of Median and Bilateral filter results ruined by the team.

Cite this article as: Tedros Berhane and Namit Nagpal. "Image Filters Using CUDA on NVIDIA". *International Conference on Cloud of Things and Wearable Technologies 2018*: 06-11. Print.

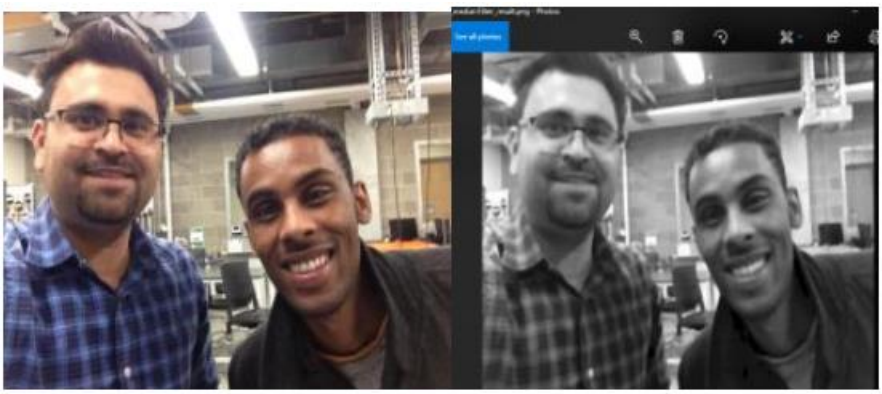


Figure 1: Median Filter

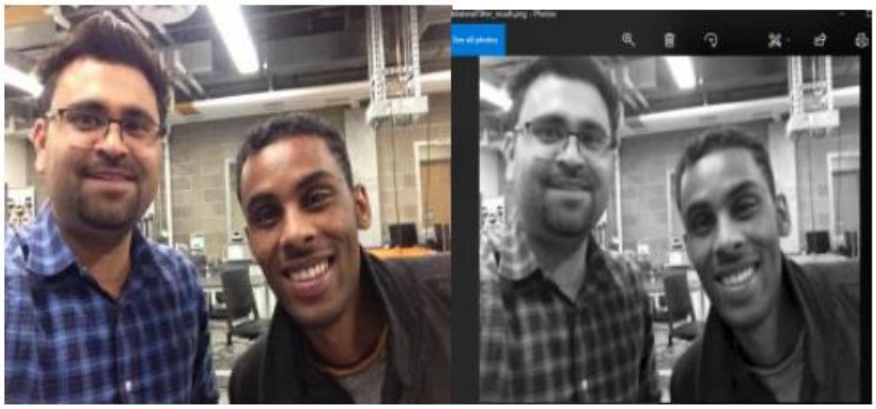


Figure 2: Bilateral Filter

COMPUTER VISION (CV)

Computer Vision (CV) is modern computing software that enables and helps in detection, classification, and recognition of objects by the body size, shapes, and appearances [6]. Nowadays, Computer vision are using heavily in autonomous or self-driving cars, where the it detects the appearances of stop lights, types of cars, and buildings. This computer software helps in guiding self-driving cars through image and video surveillances cameras. Computer visions can also detect and classify human faces and body parts. The team used Computer vision to blur two types of image filters.

OPENCV

OpenCV is open sources software supported by NVIDIA CUDA and has higher level functional Application Peripheral Interfaces (APIs), where the user or developer can customize inputs and outputs of their source's files [4]. Thereafter, the team inserted 2-Dimensional (2D) image through CV strands. OpenCV, can be programmed with C, C++, Python, and Java and has 250 built functions. When CUDA is used with OpenCV the memory models can give 5x-100x times speed faster.

CUDA

The final thing the team approached better optimization and enhance good image blurring product was implementing the sequential code at first hand and allied it with parallel code. Naïve approach of host code was compatible with OpenCV libraries and functions, but to reach better optimization and speed up tiling approaches and shared memory access of basic parallel massive programming method was utilized. The speed performances of the host and device has been displayed below in result section.

A. Median Filter

Median Filter is one type of many image or signal processing filters that commonly used in graphics to remove pixel intensity, while preserving the image edges. Pixel intensity removal is done by sorting instead of Gaussian normal distributions, the neighboring

Cite this article as: Tedros Berhane and Namit Nagpal. "Image Filters Using CUDA on NVIDIA". *International Conference on Cloud of Things and Wearable Technologies 2018*: 06-11. Print.

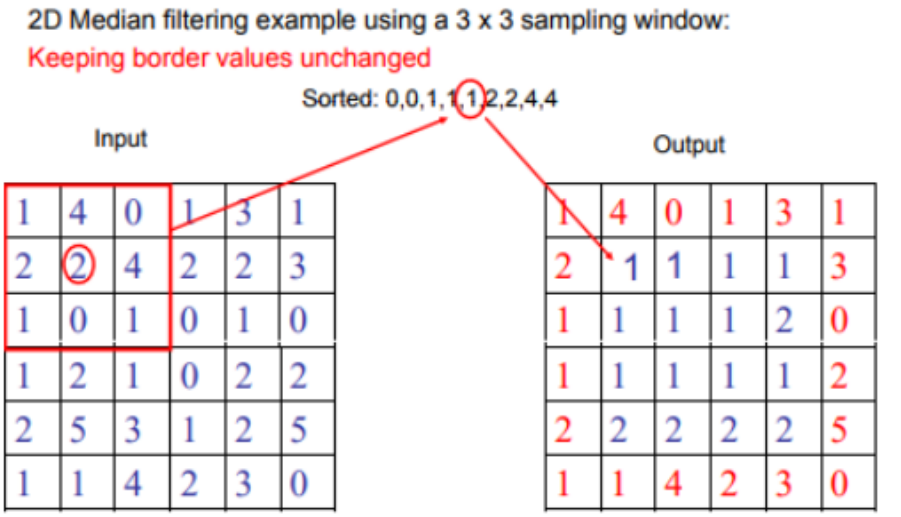
elements within sub matrix gets the median distribution and the median kernel is then replaces the element size [2]. As non-linear type of function, Median filter has cons that when sorting of median distribution is applied the average value doesn't sum up or make the whole pixel value. However, this gives better speed up performance when implemented in programming due to algorithm or logic nature of it.

B. Bilateral Filter

Bilateral is another type nonlinear image filtering model, where filter or blurring image is done by the weight average of intensity value from neighing elements, while preserving sharp edge of the pixel [3]. Bilateral has two filters, spatial, and local. Spatial filter measures the geometric closeness between the current point and nearby point, while Local filter measures the photometric similarity between the pixel value of the current point and nearby point [3]. Bilateral image blurring has higher latency due to the functional mode of the filter and the weighted average gives better distorting upshot.

C. Algorithm

Median filter algorithm used in the project by moving the image pixel by pixel, replacing each pixel with the median value of neighboring pixels. We used median of nine neighboring elements in the project. The pattern of neighbors is called the "window", which slides, pixel by pixel over the entire image pixel. The median is calculated by bubble sort all the pixel values from the window to numerical order, and then substituting the pixel actuality considered with the median pixel value for example.




```
allocate outputPixelValue[image width][image height]
allocate window[window width * window height]
edgex := (window width / 2) rounded down
edgey := (window height / 2) rounded down
for x from edgex to image width - edgex
    for y from edgey to image height - edgey
        i = 0
        for fx from 0 to window width
            for fy from 0 to window height
                window[i] := inputPixelValue[x + fx - edgex][y + fy - edgey]
                i := i + 1
        sort entries in window[]
        outputPixelValue[x][y] := window[window width * window height / 2]
```

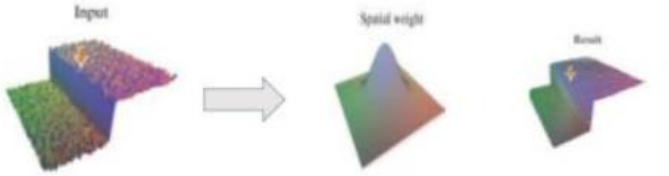
Bilateral filter algorithm used in the project by moving the image pixel by pixel the intensity of each pixel in the image is substituted by a weighted average of intensity values nine nearby pixels. This weight can be constructed on a Gaussian distribution for instance, the range variance like the color intensity.

Cite this article as: Tedros Berhane and Namit Nagpal. "Image Filters Using CUDA on NVIDIA". *International Conference on Cloud of Things and Wearable Technologies 2018*: 06-11. Print.

```
3. For each pixel p in the input_image
      p_value = 0;
      weight = 0;

      for each pixel q inside the range of spacial kernel
         p_value += Gaussian_spacial(|p-q|)*Gaussain_range(|Ip - Iq|)*Iq;
         weight += Gaussian_spacial(|p-q|)*Gaussain_range(|Ip - Iq|);
      end

      //normalize
      p_value /= weight;
```



D. Results

we tried using different type of sorting techniques but got the same result then for the final programming we use only bubble sort.

```
-bash-4.2$ ./build/filters_gpu
GPU Accelerated Median Filter took 1.11111 ms.
CPU Accelerated Median Filter took 37.7778 ms.          Linear sort
GPU Accelerated Bilateral Filter took 2.22222 ms.
CPU Accelerated Bilateral Filter took 114.444 ms.
-bash-4.2$ make
nvcc -I. -arch=sm_52 -c src/median_gpu.cu -o build/median_gpu.o
nvcc -I. -arch=sm_52 -c src/bilateral_gpu.cu -o build/bilateral_gpu.o
g++ -o build/filters_gpu src/main.cpp build/median_gpu.o build/bilateral_
 opencv` `pkg-config --libs opencv` -L/usr/local/cuda/lib64 -lcudart
-bash-4.2$ ./build/filters_gpu
GPU Accelerated Median Filter took 1.11111 ms.
CPU Accelerated Median Filter took 36.6667 ms.          Bubble sort
GPU Accelerated Bilateral Filter took 2.22222 ms.
CPU Accelerated Bilateral Filter took 114.444 ms.
```

Result with Image size 450 x 300

```
-bash-4.2$ ./build/FinalProject
GPU : Time taken by GPU Median Filter 2.22222 ms.
CPU:  Time taken by CPU Median Filter 10 ms.
GPU : Time taken by GPU Bilateral 2.22222 ms.
CPU : Time taken by CPU Bilateral 25.5556 ms.
```

Result with Image size 1450 x 1300

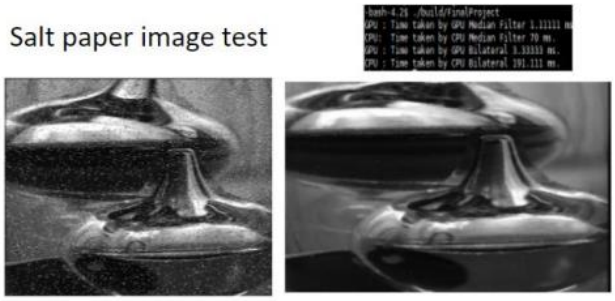
```
GPU : Time taken by GPU Median Filter 4.44444 ms.
CPU:  Time taken by CPU Median Filter 125.556 ms.
GPU : Time taken by GPU Bilateral 3.33333 ms.
CPU : Time taken by CPU Bilateral 357.778 ms.
```

Result with Image size 700 x 400

```
-bash-4.2$ ./build/FinalProject
GPU : Time taken by GPU Median Filter 2.22222 ms.
CPU:  Time taken by CPU Median Filter 20 ms.
GPU : Time taken by GPU Bilateral 1.11111 ms.
CPU : Time taken by CPU Bilateral 54.4444 ms.
```

Median filtering is a widely used image in improvement method for eliminating salt and pepper noise. Since this filtering is not as much of sensitive than linear techniques to dangerous changes in pixel values, it can eliminate salt and pepper noise without knowingly reducing the sharpness of an image.

Cite this article as: Tedros Berhane and Namit Nagpal. "Image Filters Using CUDA on NVIDIA". *International Conference on Cloud of Things and Wearable Technologies 2018*: 06-11. Print.

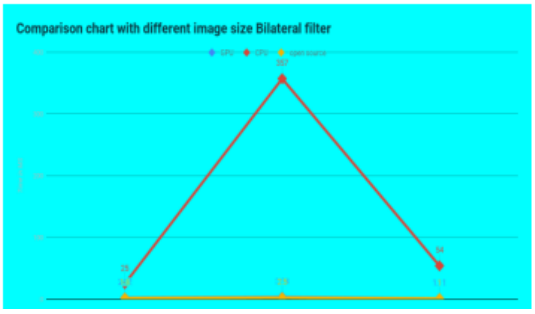


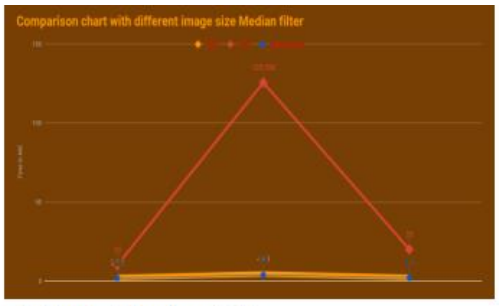
PROFILING RESULTS

Profiling results shows the usage of memory and performance of the application. It also tells us that which api used how much time and number of times it called.

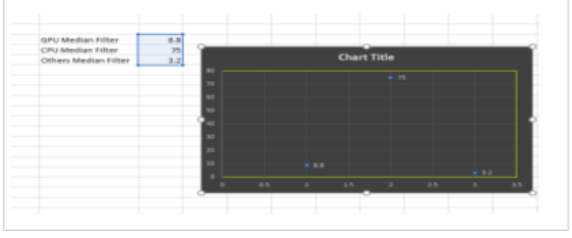
```
==9264== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 51.32%  5.7067ms        20  285.33us  285.12us  285.51us  [CUDA memcpy HtoD]
 48.68%  5.4128ms        20  270.64us  270.56us  270.69us  [CUDA memcpy DtoH]

==9264== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 95.52%  506.56ms        40  12.664ms  412.62us  487.73ms  cudaMalloc
  2.29%  12.164ms        40  304.09us  132.08us  1.1035ms  cudaMemcpy
  1.07%  5.6801ms        20  284.00us  259.57us  292.29us  cudaDeviceSynchronize
  0.94%  4.9942ms        40  124.85us  100.47us  177.78us  cudaFree
  0.08%  415.61us        91  4.5670us     341ns  160.71us  cuDeviceGetAttribute
  0.07%  378.18us         1  378.18us  378.18us  378.18us  cuDeviceTotalMem
  0.01%  41.628us        80     520ns     288ns  6.2120us  cudaSetupArgument
  0.01%  32.665us         1  32.665us  32.665us  32.665us  cuDeviceGetName
  0.00%  24.082us        20  1.2040us     870ns  3.7600us  cudaConfigureCall
  0.00%  22.857us        20  1.1420us     926ns  1.7480us  cudaLaunch
  0.00%  6.1370us         3  2.0450us     476ns  4.4310us  cuDeviceGetCount
  0.00%  2.3980us         3     799ns     499ns  1.3110us  cuDeviceGet
-bash-4.2$
```





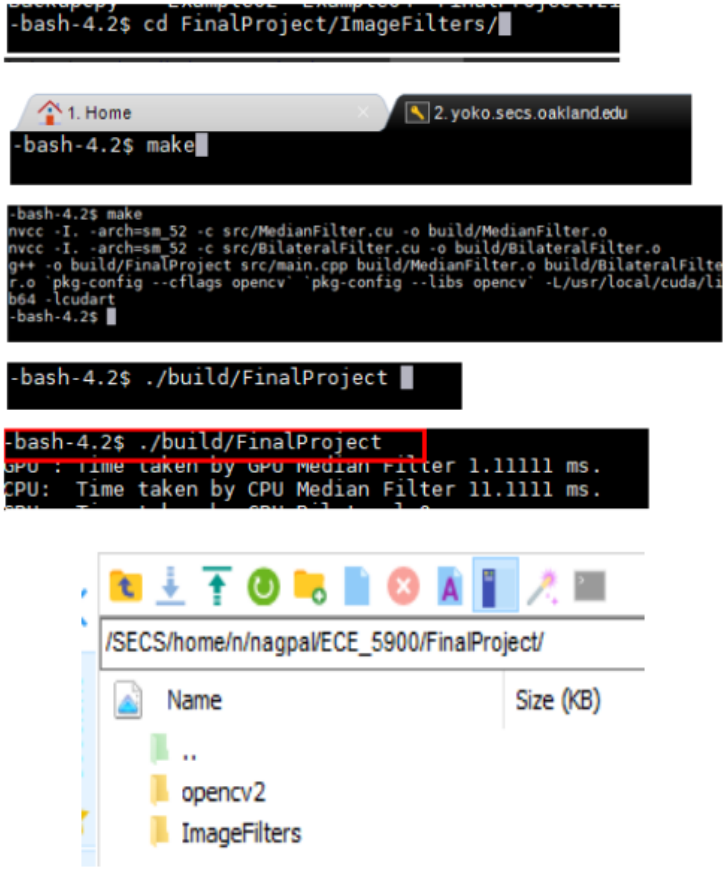
BenchMark Median filter



Cite this article as: Tedros Berhane and Namit Nagpal. "Image Filters Using CUDA on NVIDIA". *International Conference on Cloud of Things and Wearable Technologies 2018*: 06-11. Print.

How to Build the Project

OpenCV 3.3.1 is used in the project/Go to the project Directory/Type make Then. /build/Final Project/Folder Structure



Performance and Optimization (Shared Memory)

Shared Memory used to calculate the median of window in the median filter and change of the intensity in the Bilateral filter. Device constant Used to define the window size

CONCLUSIONS

Overall, we learned how to use both sequential and parallel programs through image filtering and processing in OpenCV. The most important part of the learning process was access OpenCV available libraries and functions to set up an image in in C++ programming language and resizing it. We learned how to customize freely available of both NVIDIA CUDA and OpenCV. We learned how to compare performance results with core processors and Graphical Processors. Eventually, the team has learned how to approach 2D image through tiling and shared memory accesses.

REFERENCES

1. Yipin Zhou (zyp), cs129/hdr. [Online]. Available: http://cs.brown.edu/courses/cs129/results/proj5/zyp/. [Accessed: 07-Dec-2017].
2. Median filter Wikipedia, 05-Nov-2017. [Online]. Available: https://en.wikipedia.org/wiki/Median_filter. [Accessed: 07-Dec-2017].
3. Bilateral filter, Wikipedia, 29-Oct-2017. [Online]. Available: https://en.wikipedia.org/wiki/Bilateral_filter. [Accessed: 07- Dec-2017].
4. OpenCV, NVIDIA Developer, 27-Apr-2017. [Online]. Available: https://developer.nvidia.com/opencv [Accessed: 08-Dec-2017].
5. Parallel computing, Wikipedia, 06-Dec-2017. [Online]. Available: https://en.wikipedia.org/wiki/Parallel_computing. [Accessed: 08-Dec-2017].
6. Computer vision, Wikipedia, 09-Oct-2017. [Online]. Available: https://en.wikipedia.org/wiki/Computervision. [Accessed: 08-Dec-2017]

Cite this article as: Tedros Berhane and Namit Nagpal. "Image Filters Using CUDA on NVIDIA". *International Conference on Cloud of Things and Wearable Technologies 2018*: 06-11. Print.

International Conference on Cloud of Things and Wearable Technologies 2018 [ICCOTWT 2018]

ISBN	978-93-88122-00-9	**VOL**	01
Website	iccotwt.com	**eMail**	iccotwt@asdf.res.in
Received	26 – May– 2018	**Accepted**	26 - June – 2018
Article ID	ICCOTWT012	**eAID**	ICCOTWT.2018.012

User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators

Trusit Shah[1], S Venkatesan[2], Harsha Reddy[3], Somasundaram Ardhanareeswaran[4] and Vishwas Lokesh[5]

[1,2,3,4,5]Department of Computer Science, The University of Texas at Dallas, Richardson, Texas, USA

Abstract— *Many IoT devices have been designed, built and deployed enabling users to monitor and control their systems through the Internet. Many of these IoT devices are rigid with no easy ways for users to customize. We have designed an IoT architecture to enable IoT devices to enable user customization. Customization is performed in two ways: hardware customization and software customization. In hardware customization, a user can add/remove IoT sensors/actuators, which are made self-aware by our technique, to/from the IoT system without knowing any hardware details of those sensors/actuators. For software customization, the user can create tasks without writing any code. A self-aware sensor/actuator is a sensor/actuator with an additional processing element and related components. For software customization, we have designed a rule engine, which converts user-desired actions into computer code. We have tested this architecture in several real-life scenarios.*

INTRODUCTION

Advancements in embedded technology has led to several IoT solutions being offered to various vertical markets. These offerings have resulted in many advantages for the users. Going ahead, many more devices will become "Web enabled" and join the IoT ecosystem [1]. Current IoT solutions, offered by many different providers, typically have a proprietary system including User interfaces (both smart phone and web). The lack of standardization for the IoT products makes them rigid and limits the user's ability to customize the product. The heterogeneous environment for different applications and interoperability between different types of network protocols are some of the major challenges in standardizing IoT products [3].

Many researchers have proposed different ways to standardize IoT products. Some attempts in this effort include protocol standardization. IETF has proposed protocols such as 6LoWPAN and Co AP for the constrained devices used in the IoT environment [2]. Some cloud based IoT architectures have also been recommended by the researchers to standardize the IoT architecture using cloud-based services [4] [5]. In these solutions, the cloud service handles all the types of sensors and actuators in a standardize way. Most of these IoT architectures are more focused towards standardization of IoT systems (and not customization of the IoT system at the user level). As these architectures follow some standards, they provide flexibility to the developers to rapidly create new IoT systems.

In the current IoT systems, if the user wants to customize the software part of the IoT system, the user must reprogram that IoT system. For example, if a user currently using a proprietary system to monitor temperature at a warehouse needs to monitor humidity also, he/she would have to either buy a humidity sensor from the same provider and ask them to reprogram the IoT system supporting new sensor or buy an entirely new system.

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.

In this paper, we describe a way to build an IoT system that is user customizable. Our architecture consists of four components: Self-aware sensors/actuators, an IoT gateway device (or IoT device), a server and a user interface. The self-aware sensor/actuator is connected to the IoT device using a known user interface such as USB, Wi-Fi or Bluetooth. The server communicates with IoT device and the user interface either via REST calls or over MQTT service. Any sensor/actuator can be converted into self-aware sensor/actuator using our methodology.

Our architecture focuses on user level customizations: if the user wants to deploy a new IoT system or make a change to an existing system, the user can do it without having any knowledge of and expertise in hardware or programming.

Our architecture enables the user to build an IoT system using few clicks from the user interface provided by us. For example, the user can setup a task such as "If the temperature is greater than 50 °F and humidity is 75%, turn on the Fan controller" and make changes to the IoT system easily whenever requirements change.

The paper is arranged in the following way. First, we discuss the related work in section 2. Section 3 covers the overall system architecture followed by hardware customization of our architecture in Section 4. In section 5, we describe the software customization. Section 6 describes the implementation details of the architecture. Section 7 presents a performance analysis of our architecture.

PREVIOUS WORK

There have been several ways that researchers have conceptualized the idea of user customizable IoT architecture. Kortuem, G. et al. discuss the awareness of smart IoT objects [17]. In that paper, the authors describe how a normal sensor or actuator can be aware of its surrounding and can perform tasks according to the environmental change. They have presented ways using which a developer can pre-program the IoT sensors/actuators with environmental constraints. The user doesn't have any control over customization of an IoT device.

Several approaches have been presented on the unification of IoT devices. JADE architecture is an example of one such architecture where the developer can configure and customize the IoT service [18]. JADE provides an easy way to create an IoT device using their framework. The developer needs to write some code to define the IoT system and JADE creates the IoT system from the defined script. A restful service is created by Stirbu et. al. [19] for unified discovery, monitoring and controlling of smart things. Sarar et. al [21] have introduced an IoT architecture with virtual object layer. This virtual object layer is responsible for unifying heterogeneous IoT hardware. Alam et. al [22] have proposed an architecture named SenaaS, which creates a virtual layer of IoT on the cloud. Here, IoT sensors are considered as sensor as a service and it hides all the hardware specific details of sensor to the user. Kiran et. al [23] have designed a rule based IoT system for remote healthcare application. They have created a virtual software layer to execute rules on the sensor values. As their work focuses only on a single application (healthcare), they don't have any virtualization on hardware. There has been similar work done on rule-based IoT systems. An If-Then based rule implementation architecture is explained by Zeng, D. et. alb [20]. The authors discuss the user configurable triggers for IoT systems.

Popular cloud services such as Amazon AWS, IBM Watson, ATT M2x have created a sensor as service cloud platform for IoT systems [15, 16]. These architectures are customizable at developer level. A user has the ability to configure few thresholds but the user cannot customize full IoT System.

SYSTEM ARCHITECTURE

Researchers have proposed different ways to develop an IoT architecture. These architectures can be classified into two types. In the first type, the IoT sensors/actuators directly communicates with the server over the internet [12]. In the second type, all the sensors/actuators are connected to a gateway device and that gateway device communicates with the server using a WAN interface [6,7,8,9]. The first type is suitable for the application where the number of sensors and actuators are low and/or when the network connectivity is poor.

We have designed two different architectures to demonstrate user customizable IoT system: IoT gateway-based architecture and standalone self-aware architecture. The IoT gateway-based architecture has four main components: One or more self-aware sensor(s)/actuator(s), an IoT device, a server (on the cloud) and a user interface. Fig. 1 shows the different components and their interactions.

Each self-aware sensor/actuator consists of a sensor/actuator with an additional processing element and related support components. The processing element stores basic details about the sensor/actuator, such as the id of sensor/actuator, type of sensor/actuator, parameters of sensor/actuator, etc. When the self-aware device is connected to the IoT device, all the details of sensor/actuator is communicated to the IoT device. The IoT device uploads these details to the server and obtains further instructions from the server on

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.

how to handle the newly connected sensor/actuator. Any number of self-aware sensors/actuators can be connected to the IoT device. The user can monitor all the connected sensors/actuators through the user interface and create tasks for those connected sensors/actuators.

The standalone self-aware architecture doesn't have the IoT gateway. The self-aware device (which can be either a self-aware sensor or a self-aware actuator) directly communicates with the server. The self-aware device must contain a network interface such as Wi-Fi/Cellular/Satellite modem to communicate with the server. In this case, the software (that processes the task/rule) runs on the server. Fig. 2 shows the representation of this architecture.

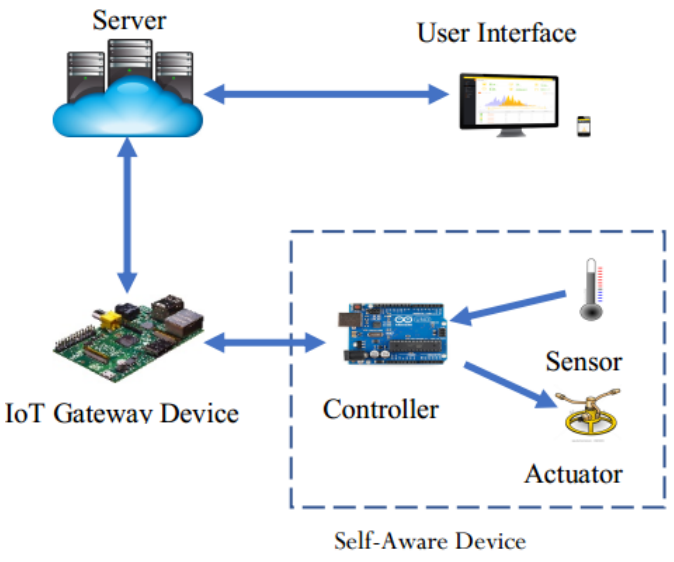


Figure 2 Gateway based self-aware architecture

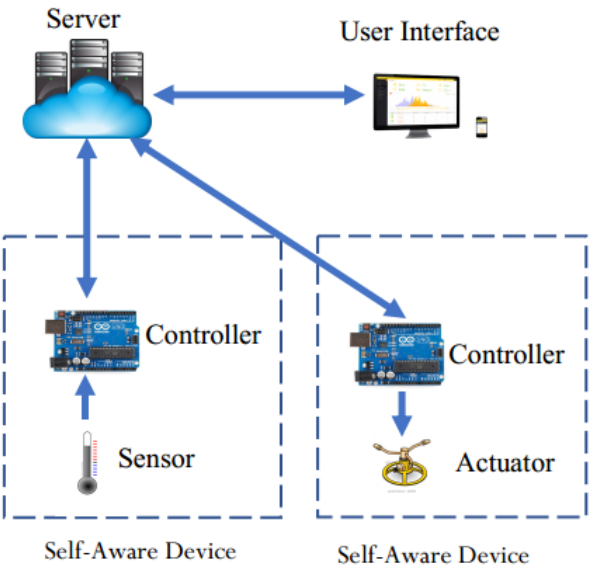


Figure 1 Standalone self-aware architecture

HARDWARE CUSTOMIZATION

We present the details of hardware customization assuming the gateway-based architecture is used. The goal of hardware customization is to enable the user to add or remove sensors/actuators as the needs change (instead of replacing the whole IoT system). To achieve such a goal, every time a new self-aware sensor/actuator is connected to the IoT gateway, the self-aware sensor/actuator should be able to identify itself to the gateway. The sensor/actuator knows the basic details about itself and, once connected to the IoT gateway, it communicates those details to the IoT gateway device. The following section gives the details of self-aware sensor/actuator. We use the term developer to denote a person who is an expert in hardware/software and can create a self-aware device (sensor or actuator). The term user refers to the end user of the IoT system with no expertise in hardware/software.

A. Self-Aware Sensor/Actuator

A self-aware sensor/actuator is basically a sensor/actuator with additional components to make that sensor/actuator self-aware. A self-aware sensor/actuator typically has the following components.

- Sensor/Actuator
- Processing Element
- Driver circuit for an actuator
- External memory (if processing element doesn't have sufficient memory)
- Communication interface to communicate with IoT gateway device
- Power supply (if sensor/actuator can't be powered by IoT gateway device)

B. Converting a Sensor/Actuator into a Self-Aware Sensor/Actuator

Steps to convert an arbitrary sensor/actuator into self-aware sensor/actuator:

1. Select suitable processing element.
2. Interface sensor/actuator to processing element.

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.

3. Store parameters of sensor/actuator in the memory.
4. Create a communication interface for communication between a self-aware device and the IoT gateway device connected to it.
5. Create and embed self-aware device discovery code on IoT gateway device and self-aware device to detect connected nearby (if wirelessly connected) self-aware devices.

Select Suitable Processing Element

Depending on the type of sensor/actuator, the developer selects the processing element. For example, if the type of sensor is analog, it is advisable to select a processing element which has an inbuilt analog to digital converter. Some sensors such as crack detection sensor or fingerprint sensor require a complex algorithm to read sensor value, and hence a processing element with significant memory and processing power is preferable. The power consumption of processing element is also important, as some installations may require a battery powered solution. For those applications, processing elements with power saving capabilities should be selected.

Interface Sensor/Actuator to Processing Element and Program Parameters of Sensor/Actuator into Memory

The developer interfaces the sensor/actuator to the processing element depending on the type of the sensor/actuator. After that the developer programs the parameters of sensor/actuator into the memory. The parameter can be of two types: fixed parameters and user configurable parameters. Fixed parameters don't change over time (such as the unique id of a sensor/actuator, type of a sensor/actuator or manufacturing date of sensor/actuator). User configurable parameters are the parameters which the user can define according to the application (for example, the time interval between two consecutive sensor readings). The user can define these user configurable parameters from the user interface.

Communication Interface between Self-Aware Device and IoT Gateway Device

The communication interface between the self-aware device and the IoT device should be user-friendly and known to the user. WiFi, USB, and Bluetooth are some examples. The processing element is connected to one such communication interface to communicate with self-aware sensor/actuator. The IoT device and self-aware sensor/actuator communicate using two methods: query-response method and interrupt method. In the query- response method, the IoT device queries the self-aware sensor/actuator and self-aware sensor/actuator responds it. In the interrupt method, the IoT device turns on the interrupt mode where the self-aware device sends messages to the IoT device without any query. We have developed a library for the query response model in embedded-c which is suitable for most embedded hardware used for self-aware device.

Self-Aware Device Discoverable Code for IoT Gateway Device

An IoT gateway device should be able to discover all the self-aware devices near it (for wireless connection) and all the devices connected to it (for wired connection). For different types of communication protocols, the method of discovering self-aware sensor/actuator changes. For example, for USB, the developer just needs to check the/dev/tty USB ports for checking connected USB device. For Wi-Fi, a multicast signal with a certain message can be sent and all the nearby self-aware devices respond back. The developer should write code to enable discovery for all the communication interfaces available on that IoT device. The developer also writes the code for self-aware device, so that it responds back to the IoT device's device discovery query.

C. Self-Aware Virtual Sensor

Virtual sensors are sensors that are not physically connected to the IoT system. For example, weather feed from the weather API and time from the NTP server are some of the virtual sensors we have incorporated into our implementation. We have created a virtual sensor API in our architecture, which takes virtual sensors as input and attaches a unique id, sensor type and other parameters to make it self-aware.

D. Validation of Sensor/Actuator Value

Validation of sensor values is an important feature of the self-aware architecture. As the sensors/actuators are self- aware, they know their typical range of readings of the sensor outputs. These will also be stored as part of the sensor-specific data. If any sensed value is out of this range, the self-aware sensor notifies the user about the deviation. For example, a ds1820 temperature sensor is connected to the self-aware device and the maximum value ds1820 temperature sensor can have is 125 ℃ [10]. If the sensor reads more than 125 ℃, the self-aware sensor itself needs to generate an error message (in addition to possibly sending the error reading to the server). Another example of self-validation is related to a self-aware actuator. Suppose we have a relay that controls a compressor as the

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.

actuator, which has been made self-aware. For compressor longevity, the compressor should not be turned on and off frequently. There should be a minimum elapsed time between two successive times when it is turned on [11]. The self-aware actuator knows this constraint and, if it receives too many commands for turning on and turning off the relay (actuator), it disregards the received commands and generates an error to inform the user. This property makes sure that neither the user nor the server needs to be aware of actuator-specific constraints. Instead, the self-aware device has the knowledge of the constraints and has checks and balances built in.

E. Working Status of Sensor/Actuator

It is important to know if the connected sensors/actuators are actually working or not. For a sensor, we can detect its working condition by its current value. If a sensor stops sending values or sends values that are out of range, we can conclude that that sensor is not working properly assuming that the communication channel is not faulty. This type of property only works on sensors which give analog values (e.g. temperature sensor or pressure sensor which gives output over a range). It is not possible to detect working status for some of the digital sensors (for example touch sensor which gives either 1 or 0 as its output). We cannot decide whether the sensor is stuck at a single value or it is providing normal input. We can define such analog and digital sensors in the device property part of the unique id of the sensor and notify the user whether the user can get the working status of the sensor or not.

Checking the health of an actuator is harder than that of a sensor. Many actuators may not provide any feedback to the processing element. However, this can be rectified by using auxiliary parts. For example, we can connect an appropriate new sensor to the actuator and retrieve values from the sensor. Using the output of the new sensor, we can check whether the actuator is working or not. Failure of the feedback sensor can raise a false alert. Fig.3 explains the working of this feedback mechanism.

V. Software Customization

Our architecture provides user customization of software where the user can create tasks/rules and set the user defined parameters for self-aware sensors/actuators. For example, the user can set how frequently the user wants to read the sensor values as a user-defined parameter.

Rules are divided into two main components:

- Trigger Condition: When the trigger condition is true, the rule is executed by either IoT gateway device or the server. A single rule can have multiple trigger conditions. When a rule has more than one trigger condition, the rule is executed when all the trigger conditions are true or a specific combination of triggers occurs.
- Actions: This represents the actions to be taken when the conditions are true. Some examples of actions our architecture supports are as follow.
    1. Send Text Message/App Notifications.
    2. Send Email
    3. Make phone calls and play notification message
    4. Turn on/off Actuator
    5. Turn on/off Main Power

More actions are user customizable and can be added to the system.

A single rule can have more than one action. All the actions are executed when the rule is true. An example of full rule is: "If (temperature > 82 && humidity > 40) then "turn on" the fan controller and send message & email" This rule will turn on fan controller (a self-aware actuator) and send out the notifications when the value of temperature (a self-aware sensor) is more than 82 and the value of humidity (a self-aware sensor) is more than 40. While this is a simple if then else type of rule more complex rules are also possible.

A. Execution of Rule

A rule can be executed either on the server or on the IoT device. For the standalone self-aware architecture, a rule must run on server because the standalone self-aware devices may not be capable of running the rule engine. For the gateway-based architecture, a rule can run either on the server or on the IoT gateway device. When all the self-aware sensors/actuators attached with the rule are connected to the same IoT gateway device, the rule can be executed on the IoT gateway device. When the self-aware sensors/actuators are connected to different devices, the rule is executed on the server. When a rule is executed on the server, slow or intermittent internet connectivity can cause problems in execution of the rule. When a user creates a rule, the rule is stored in our database in the form of different tables. Execution of the rule is done as follows:

If the server is executing the rule, it fetches the rule from the database and sends it to the rule engine. The rule engine is a service in

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.

our architecture, which takes the rule as input, fetches sensor values related to the rule and outputs appropriate commands to the actuator. If the IoT device is executing the rule, it fetches the rule from the server using REST calls and after that, it sends the rule to the rule engine running on the IoT device. The architecture of Rule Engine is shown in Fig. 4.

The code executed by the IoT device or the server is generated automatically by our backend. The user doesn't write any code.

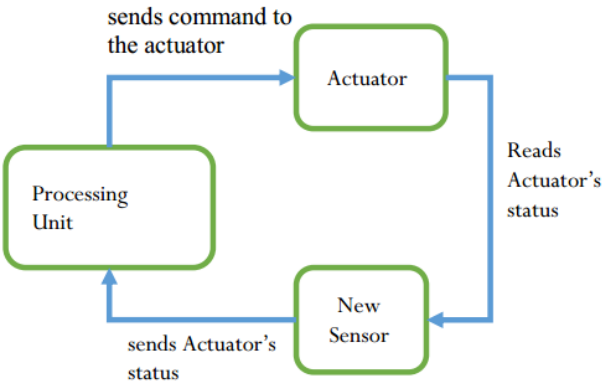


Figure 3 Feedback system for an actuator

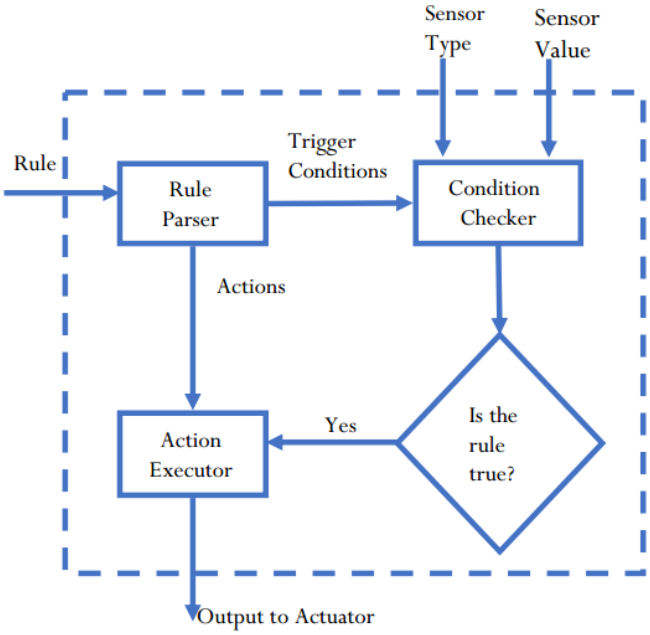


Figure 4 Rule execution block diagram

IMPLEMENTATION

We have implemented a basic prototype for such a self-aware sensor/actuator architecture. This implementation consists of four basic components: IoT gateway Device, self-aware sensor/actuator, server and user interface.

A. Server

We have used a server running CentOS Linux by Digital Ocean. We have used JAVA Spring MVC framework as our main web server. REST calls are used to communicate with the server. All the self-aware sensor and actuator details are stored in the database at our server, once they become active. Hibernate platform is used to communicate with the database from the web server.

B. IoT Device

We have used a Raspberry Pi as the IoT device. We have created four threads to perform the following tasks independently:

- Detect any new USB device connection or removal of a self-aware device.
- Communicate with connected self-aware sensors/actuators via USB. When there are more than one sensors/actuator, the IoT device communicates with them one by one.
- Get push notification from the server via MQTT.
- Run the rule engine.

When any newly connected device is detected by the first thread, the thread checks for the type of new device connected to it and if the new device is a self-aware device, it configures it and adds it to the list of connected devices.

According to the constraints of self-aware sensor/actuator, the second thread will communicate with the sensor and upload data through the services provided by the server.

If the user wants to manually turn on/off the actuator, the user can send a request to the server via the user interface. The server sends the same request to the appropriate IoT device using MQTT notification. The third thread running on IoT device receives this request

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.

and sends the appropriate command to turn on/off the actuator.

If all the self-aware sensors and self-aware actuators related to a rule are attached to the same IoT device, the last thread of IoT device code executes the rule and sends appropriate actions to the self-aware actuator.

C. Self-Aware Device

For proof of concept, an Arduino board has been used to build the self-aware sensor (or actuator). [A custom hardware will make the device minimal and aesthetically appealing, but will be time-consuming to build]. Temperature sensor ds1820 is connected to it. The sensor ds1820 works on 1 wire protocol. When the Arduino is connected to the Raspberry Pi it sends all details such as unique id, sensors connected to it, the range of the sensor, sensor's other constraints, etc. All these details are stored in nonvolatile memory of Arduino. After getting all these values, the Raspberry Pi will periodically request values from Arduino using AT commands. The Arduino will read the value from ds1820 via 1 wire protocol, and send the value to the pi. Before sending data to the pi, the Arduino performs preliminary check on the data to ensure that the data being provided to the Raspberry Pi is valid.

D. User Interface

We have created a simple web interface. It consists of following web pages.

- Summary Page: It displays basic details of the user. It also displays all the IoT devices owned by the user. The user can import a new IoT device and delete an existing IoT device from this page.
- Rule Page: User can create a new rule from this page. The user can select the IoT device to "program", see what sensors and actuators are connected to it, and then from a set of pull-down menus, select the desired behavior. This page is responsible for taking all the data from the user to create the needed tables which will be used by our backend to automatically create code to implement the rule.
- Data Display Page: All the sensor values are displayed on this page.

E. Actual Implementation

We have converted the following sensors and actuators into self-aware sensors and self-aware actuators.

Sensors:

- Temperature sensor
- Soil moisture sensor
- Water level sensor
- Ultrasonic sensor
- Barcode reader

Actuators:

- Sprinkler controller
- Fan controller
- Magnetic Lock controller

We have also deployed one system at a local organic farm named Profound Microforms and the system has been operating well for the past 9 months. The system consists of two self-aware sensors measuring air and water temperature respectively. The user can customize text message alert or email alert on this system. The farmer at Profound Microforms have customized alerts according to their requirements. For example, during winter they set an alert for the air temperature value greater than 70 °F and during summer they set an alert for the air temperature value greater than 85 °F.

PERFORMANCE ANALYSIS

We measured timing evaluation and power evaluation for the performance analysis on an actual proof of concept implementation. As the self-aware architecture requires additional processing element, it delays the end to end communication and it also requires more power. We performed our analysis on a Raspberry Pi as the IoT device and Arduino as a self-aware processing element. We used Java as the programming language for Raspberry Pi and C for Arduino.

A. Timing Analysis

We checked timing analysis by performing end to end communication on following two architectures. Basic IoT architecture, where

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.

the sensor is connected directly to IoT device and self-aware IoT architecture, where the sensor is connected to IoT gateway device through the self-aware device. Self-aware architecture introduces an additional communication delay between IoT device and self-aware device. This additional delay varies based on data rate used between the IoT device and the self-aware device. In our analysis, we have connected the IoT device with the self-aware device using USB protocol. We performed timing analysis on temperature sensor which contains a payload of 3 bytes from self-aware device to IoT device. We performed timing analysis for 100 cycles and took an average for analysis. Fig. 5 shows the analysis.

B. Power Analysis

Power consumption is divided into 3 main parts. Power consumed by IoT device, power consumed by self-aware device and power consume by sensor/actuator. Power consumed by self-aware devices is an additional power consumed in our architecture. The power consumption of self-aware device depends on the clock frequency of the self-aware sensor/actuator, communication data rate between the self-aware device and the IoT device, communication frequency (how often) between the self-aware device and IoT device and power saver mode used in the self-aware device. We theoretically calculated the power consumption using the following assumptions: the self-aware device has sleep mode and it wakes up periodically to sense the reading and sends it to the IoT device. Communication happens over USB with 9600 baud rates. Clock Frequency and voltage used are 16MHz and 5V respectively. For our configuration, current consumption during wake-up mode is 19.9 mA and sleep mode is 3.14 mA [13, 14]. Power consumptions for various duty cycles are shown in Table 2. Power consumption is computed using the following equation:

$$P = \frac{ts * Vin * I(sleep) + tw * Vin * I(wake\ up)}{ts + tw}$$

Vin = input voltage
ts = sleep time
tw = wake up time
I(sleep) = current consumption during sleep time
I (wake up) = current consumption during wake-up timeP = Power consumed

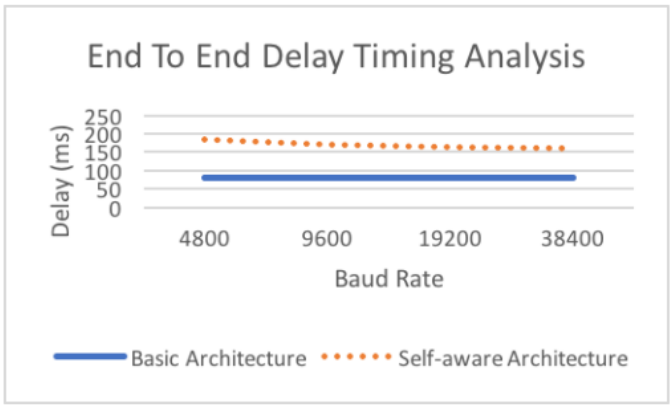


Figure 5 Timing Analysis

Wakeup time	Sleep time	Power consumption
1 second	59 seconds	17.09 mW
1 second	29 seconds	18.49 mW
1 second	9 seconds	24.08 mW
1 second	0 second (orno sleepmode)	99.5 mW

Table 1 Power Consumption

CONCLUSION

We have designed and implemented a user configurable IoT architecture using which a user can add (or remove) one or more self-aware sensor/actuator to (or from) the IoT system without knowing any hardware knowledge. The user can also define rules that will govern the execution of the IoT system, without having to write any software. This architecture enables a novice user to build a highly customized IoT system. We have built several proof of concept prototypes to validate the idea.

Security is an import requirement of IoT systems. Numerous approaches are possible for this task. Implementing a suitable security protocol for our framework is an important next step. We are working on this and related issues.

REFERENCES

1. Khan, R., Khan, S. U., Zaheer, R., & Khan, S. (2012, December). Future internet: the internet of things architecture, possible applications and key challenges. In Frontiers of Information Technology (FIT), 2012 10th International Conference on (pp. 257-260). IEEE.
2. Ishaq, Isam, et al. "IETF standardization in the field of the internet of things (IoT): a survey". Journal of Sensor and Actuator

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.

Networks 2.2 (2013): 235-287.
3. Bandyopadhyay, Debasis, and Jaydip Sen. "Internet of things: Applications and challenges in technology and standardization". Wireless Personal Communications 58.1 (2011): 49-69.
4. Tao, F., Cheng, Y., Da Xu, L., Zhang, L., & Li, B. H. (2014). CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system. IEEE Transactions on Industrial Informatics, 10(2), 1435-1442.
5. Alam, Sarfraz, Mohammad MR Chowdhury, and Josef Noll. "Senaas: An event-driven sensor virtualization approach for internet of things cloud". Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on. IEEE, 2010.
6. Tan, Lu, and Neng Wang. "Future internet: The internet of things". 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). Vol. 5. IEEE, 2010.
7. Datta, Soumya Kanti, Christian Bonnet, and Navid Nikaein. "An IoT gateway centric architecture to provide novel M2M services". Internet of Things (WF-IoT), 2014 IEEE World Forum on. IEEE, 2014.
8. Ren, J., Guo, H., Xu, C., & Zhang, Y. (2017). Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing. IEEE Network, 31(5), 96-105.
9. Hada, Hisakazu, and Jin Mitsugi. "EPC based internet of things architecture". RFID-Technologies and Applications (RFID-TA), 2011 IEEE International Conference on. IEEE, 2011.
10. https://datasheets.maximintegrated.com/en/ds/DS18S20.pdf
11. http://www.ni.com/white-paper/2774/en/
12. Yashiro, T., Kobayashi, S., Koshizuka, N., & Sakamura, K. (2013, August). An Internet of Things (IoT) architecture for embedded appliances. In Humanitarian Technology Conference (R10-HTC), 2013 IEEE Region 10 (pp. 314-319). IEEE.
13. http://www.home-automation-community.com/arduino-low-power-how-to-run-atmega328p-for-a-year-on-coin- cell-battery/.
14. http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf
15. https://m2x.att.com/
16. https://aws.amazon.com/iot/
17. Kortuem, G., Kawsar, F., Sundramoorthy, V., & Fitton, D.(2010). Smart objects as building blocks for the internet of things. IEEE Internet Computing, 14(1), 44-51.
18. Ghosh, Debashish, Fan Jin, and Muthucumaru Maheswaran. "JADE: A unified programming framework for things, web, and cloud". Internet of Things (IOT), 2014 International Conference on the. IEEE, 2014.
19. Stirbu, Vlad. "Towards a restful plug and play experience in the web of things". Semantic computing, 2008 IEEE international conference on. IEEE, 2008.
20. Zeng, D., Guo, S., Cheng, Z., & Pham, A. T. (2011, September). IF-THEN in the internet of things. In 2011 3rd international conference on awareness science and technology (iCAST) (pp.503-507). IEEE.
21. Sarkar, C., Nambi, S. A. U., Prasad, R. V., & Rahim, A. (2014, March). A scalable distributed architecture towards unifying IoT applications. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 508-513). IEEE.
22. Alam, S., Chowdhury, M. M., & Noll, J. (2010, November). Senaas: An event-driven sensor virtualization approach for internet of things cloud. In Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on (pp. 1-6). IEEE.
23. Kiran, M. S., Rajalakshmi, P., Bharadwaj, K., & Acharyya, A. (2014, March). Adaptive rule engine based IoT enabled remote health care data acquisition and smart transmission system. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 253-258). IEEE.

Cite this article as: Trusit Shah, S Venkatesan, Harsha Reddy, Somasundaram Ardhanareeswaran and Vishwas Lokesh. "User Customizable IoT Systems Using Self-Aware Sensors and Self-Aware Actuators". *International Conference on Cloud of Things and Wearable Technologies 2018*: 12-20. Print.