Build A Web Application On Firebase

Twelve Python Tips For Beginners

₹120
ISSN-2456-4885

Volume: 11 | Issue: 01 | Pages: 100 | November 2022

OpenSource ForYou

THE COMPLETE MAGAZINE ON OPEN SOURCE

An EFY GROUP Publication

The World Of Connected Devices

How IoT Can Make The Workplace Intelligent

Industry 4.0 And Embedded Internet Of Things: An Introduction

IoT Analytics: Making Sense Of Big Data

A Complete Guide To Getting Started With Arduino

Open Source India 2022: Post-Show Report

Simplified Logging And Analysis With Parseable

An EFY GROUP EVENT

smartBHARAT

23-25 NOV, 2022 | KTPO (BENGALURU)

An Event to accelerate adoption of Smart Tech across Bharat

SMART TECH FOR SMART BHARAT

smart GOV | smart AGRICULTURE | smart INDUSTRY | smart VEHICLES | smart BUILDINGS | smart HEALTHCARE

The Summit

A 3 day summit where thought leaders from tech and different facets of Bharat will come together to share ideas, success stories, and challenges that still need to be addressed.

The Expo

SHOWCASING

- Latest solutions for smartBHARAT
- Enablers for developers of smartBHARAT solutions

To exhibit, please call +91-9811155335 or growmybiz@efy.in

To visit: www.smart-bharat.in

Does Your Antivirus Solution Provide You With Complete Protection?

Don't RELY on 20 years old technology to FIGHT current date viruses

TOP 10 things that your Antivirus solution should provide to tackle today's threat...

Ransomware File Protection	ATP- Advance Threat Protection	EDR- Endpoint Detection And Response	
Application Blocking	Deep Learning Malware Analysis	Exploit Prevention	URL Blocking
Disk and Boot Record Protection	Peripheral Control	Respond Investigate Remove (Root Cause Analysis)	

If it doesn't, contact us for a solution which can

Contact: Santosh on 9971696319 or Email at santosh.gupta@itsipl.com

ITS
A Total Solutions Company

I. T. Solutions India Private Limited
D-88/5, Okhla Industrial Area, Okhla Phase I, New Delhi -110020
Ph: 011-47695000 • Email: sales@ itsipl.com • www.itsipl.com

- Mumbai
- Jaipur
- Chandigarh

SOPHOS 2020 BEST NEXTGEN PARTNER | Delhi/ NCR
SOPHOS 2019 BEST NEXTGEN PARTNER | Delhi/ NCR
SOPHOS 2018 BEST SI PARTNER | Delhi

CONTENTS

NOVEMBER 2022 | ISSN-2456-4885

FOR U & ME

FOCUS

DEVELOPERS

ADMIN

www.OpenSourceIndia.in

19th Edition OPEN SOURCE INDIA

THANK YOU

Speakers

90+ Experts Share Knowledge here

For more details, call on +91-98111-55335 or email us at info@opensourceindia.in

EDITOR
RAHUL CHOPRA

EDITORIAL, SUBSCRIPTIONS & ADVERTISING
Delhi (HQ)
D-87/1, Okhla Industrial Area, Phase I, New Delhi 110020
Phone: +91-9811155335
E-mail: info@efy.in

MISSING ISSUES
Phone: +91-9811155335
E-mail: support@efy.in

BACK ISSUES
Phone: +91-9811155335
E-mail: support@efy.in

NEWSSTAND DISTRIBUTION
Phone: +91-9811155335
E-mail: efycirc@efy.in

ADVERTISEMENTS
NEW DELHI (HEAD OFFICE)
Phone: +91-9811155335
E-mail: efyenq@efy.in

MUMBAI
E-mail: rmowest@efy.in

BENGALURU
E-mail: rmosouth@efy.in

CHINA
Worldwide Focus Media
E-mail: china@efy.in

GERMANY
pms Plantenberg Media Service GmbH
E-mail: germany@efy.in

JAPAN
Tandem Inc.
E-mail: japan@efy.in

TAIWAN
J.K. Media
E-mail: taiwan@efy.in

UNITED KINGDOM
ASA Media
E-mail: uk@efy.in

UNITED STATES
E & Tech Media
E-mail: usa@efy.in

SUBSCRIPTION RATES			
Year	**Newstand Price (₹)**	**You Pay (₹)**	**Overseas**
Five	7200	4320	—
Three	4320	3030	—
One	1440	1150	US$ 120

Kindly add ₹ 50/- for outside Delhi cheques.
Please send payments only in favour of EFY Enterprises Pvt Ltd.
Non-receipt of copies may be reported to **support@efy.in**—do mention your subscription number.

CONTENTS

Google unveils TensorStore, a library to help manage large data sets

The open source C++ and Python framework TensorStore, developed by Google, aims to accelerate the design for reading and writing huge multi-dimensional arrays. Multi-dimensional data sets that cover a single large coordinate system are commonly used in contemporary computer science and machine learning applications. These data sets are challenging to work with because customers frequently wish to conduct investigations involving numerous workstations operating in parallel, and may receive and output data at unpredictable intervals and varied scales.

Google Research has developed TensorStore, a library that provides users with access to an API that can manage huge data sets without the requirement for sophisticated hardware, to address issues with data storage and manipulation. Numerous storage systems, including local and network file systems, Google Cloud Storage, and others are supported by this library.

To load and work with enormous arrays of data, TensorStore provides a simple Python API. Any arbitrary big underlying data set can be loaded and updated without having to store it in memory, because no actual data is read or kept in memory until the precise slice is required.

This is made possible by the indexing and manipulation grammar, which is quite similar to the syntax used for NumPy operations. TensorStore supports virtual views, broadcasting, alignment, and other sophisticated indexing features like data type conversion, down sampling, and haphazardly created arrays.

Additionally, it includes an asynchronous API that enables read or write operations to work concurrently.

Legit Security announces new tool to secure GitHub configurations

Legitify, an open source security tool to secure GitHub implementations, has been announced by Legit Security, a cyber security company with an enterprise platform to secure an organisation's software supply chain. Security teams and

FIWARE Foundation and Red Hat join hands to develop a smart city solution

Red Hat, Inc., has announced a partnership with the FIWARE Foundation, a non-profit organisation that promotes the use of open standards in the creation of smart solutions, to create an integrated platform that will use data to make cities all over the world more resilient and improve the wellbeing of their residents. Red Hat Open Innovation Labs collaborated with the FIWARE Foundation and Human Oriented Products (HOPU), a solution provider involved in the FIWARE community, over the course of a six-week residency to develop an enhanced smart city solution that is simple to deploy, fully scalable, reliable and runs on Red Hat OpenShift.

Cities around the world are dealing with escalating and complicated environmental, economic, and social issues that have an impact on the daily lives of their residents. City officials and public sector decision-makers need eco-smart technology solutions to solve these issues so they can use data to assist them make decisions that are good for the environment and for their constituents.

The smart city platform powered by FIWARE is able to derive insights from data processed by HOPU air quality sensors in order to adopt wiser decisions for the welfare of inhabitants. It can be deployed on any private or public cloud. This platform and its base application for air quality monitoring have been implemented in Las Rozas and Cartagena, two Spanish cities, as pilot use cases.

CERN develops a detailed open science policy

CERN (European Council for Nuclear Research) established a new open science policy for itself, which comes into immediate effect, during its 209th session in September this year. The goal of the policy is to make all CERN research completely open, inclusive, democratic, and transparent for other researchers as well as for the general public. The Open Science Strategy Working Group (OSSWG), which has representatives from each CERN department, developed this policy. A web page dedicated to detailing all the open scientific projects at CERN has been published together with the policy statement.

The finalised policy was developed in response to the European Strategy for Particle Physics' 2020 update, which emphasised the value of open science, and the UNESCO Recommendation on Open Science, which was released in 2021. Right since the CERN Convention was ratified at UNESCO in 1952, open science has been one of the organisation's guiding principles.

All research publications, experimental data, and research software and hardware will be made publicly available under the new policy, which also incorporates the previous rules for open access, open data, and open source software and hardware. Additionally, it combines other open science tenets such as training, outreach, and citizen science, which aim to inform and engage the public and the next generation of researchers.

DevOps engineers can monitor and enforce their GitHub configurations in a secure and scalable manner with the aid of Legitify, a GitHub misconfiguration checker. With support for Windows, Mac, and Linux, Legitify is a cross-platform security solution that offers only a small portion of the features offered by the larger Legit Security platform.

Software developers all across the world utilise GitHub, a very well-liked source code management (SCM) solution at the centre of many enterprises' software supply chains. However, because GitHub is a complicated product, managers may fail to notice incorrect configurations and insecure default settings, which can lead to security flaws that jeopardise an organisation's software supply chain.

Due to the distinct configurations and protections necessary for each repository, ensuring security across big GitHub installations was challenging and time consuming prior to the advent of open source Legitify. It took a lot of manual work and the efforts of a large GitHub group to consistently enforce security were prone to error.

Along with Legitify, Legit Security has given back to the online security community by responsibly disclosing more GitHub flaws found by its in-house security research team. Legit Security also participates actively in groups like OpenSSL and the Linux Foundation, where it helps to advance the security of the software supply chain and safe software development in general.

Managing security issues in the open source world is important

Extensively used open source projects are today being maintained by a limited number of engineers. These engineers are unable to provide voluntary time and effort completely to the job since they also need to pay their bills.

For more complex open source projects, this can be a problem. As an illustration, the Linux kernel project consists of more than 30 million lines of code, contains hundreds of flaws that need to be resolved, and has close to 2000 active developers. Each active developer has written more than 15,000 lines of code.

According to recent research from the Linux Foundation, an application has an average of 5.1 significant vulnerabilities that are still open, and 41 per cent of enterprises lack confidence in the security of their open source software. To make matters worse, only 49 per cent of businesses have an open source security policy.

Since OSS updates and patches must be implemented to secure systems, this requirement can bring unique difficulties. Updating mission-critical software could result in functionality loss and/or unplanned downtime if the solution depends on a certain software version. When a situation is business-critical, it may be more elegant to hire a specialist to backport the patch, and maintain a version for longer than the larger community is willing to.

Contributing to open source as it was intended, improving the code and publishing it for others, or hiring professionals to manage the OSS code and debug it as necessary are options. However, the industry cannot afford to not contribute anything at all.

HFCL introduces open source Wi-Fi 7 access points

The world's first open source Wi-Fi 7 access points were introduced by HFCL Limited, in collaboration with Qualcomm Technologies, Inc., on October 1, 2022, at the India Mobile Congress in Pragati Maidan, New Delhi.

The access points are based on IEEE 802.11be, a ground-breaking Wi-Fi technology that is intended to give extremely high throughput (EHT), increased spectrum efficiency, better interference mitigation, and support for real-time applications (RTA). In order to provide a better user experience while using less power, Wi-Fi 7 uses faster connections with 320MHz and 4kQAM, numerous connections with multi-link operation, and adaptive connections for adaptive interference puncturing.

Wi-Fi 7 promises a significant technological advance over all prior Wi-Fi standards, providing a more immersive user experience and paving the way for a more robust digital future. The peak data speeds supported by HFCL's Wi-Fi 7 APs will exceed 10Gbps, and they will have latency under 2ms as opposed to the 5Gbps and 10ms of existing Wi-Fi 6 products.

Lyra audio codec gets an update

The open source Lyra audio codec, which Google first released last year, has been updated with Lyra V2, which has a new architecture that enables scalable bitrate capabilities, greater performance, higher audio quality, and compatibility with more systems.

Lyra V2's internal architecture is based on the SoundStream end-to-end neural audio codec, which has a 'residual vector quantizer' (RVQ) sitting before and after the transmission channel and the ability to adjust the amount of quantizers to be used at any time to alter the audio bitrate. The three available bitrates are 3.2kbps, 6kbps, and 9.2kbps. Lyra V2 makes use of artificial intelligence, and a TensorFlow Lite model makes it possible for it to function on Linux and Android smartphones. Support for Mac and Windows is experimental. At this time, iOS and other embedded platforms are not supported; however, things might change later.

As compared to other audio codecs like Lyra (V1) and Opus, the new audio codec offers a higher quality (MUSHRA score) than those at a given bitrate. Lyra V2 at 9.2kbps offers roughly the same quality as Opus at 14kbps.

This second generation codec's latency has improved from 100ms to 20ms, making it similar to Opus for WebRTC, which typically has delays of 26.5ms, 46.5ms, and 66.5ms. In order to provide real-time audio encoding/decoding and reduce power consumption, Lyra V2 also encodes and decodes five times quicker than Lyra V1.

Scout APM releases TelemetryHub

Application performance monitoring (APM) service provider Scout APM has announced the launch of TelemetryHub via its Early Access Program (EAP). By utilising the open source initiative OpenTelemetry, TelemetryHub offers observability solutions for small- to medium-sized engineering teams.

Scout, which was first developed as a server monitoring tool in 2009, has since expanded to offer insight into the performance of applications for a growing number of supported languages. It is currently developing into an observability platform that will allow teams to diagnose problems and shorten the mean time to resolution (MTTR) by providing a straightforward, reasonably priced, and practical single pane of glass view across distributed systems.

Users can ingest telemetry data from platforms instrumented with the OpenTelemetry CNCF initiative using TelemetryHub. As an early adopter of OpenTelemetry, TelemetryHub has the support of a project that has amassed over 100,000 contributors from the community. TelemetryHub is dedicated to evangelising and supporting the project through contribution and education. This gives users the chance to employ an ever-evolving, vendor-neutral solution that gives them complete control and freedom with their observability data.

TelemetryHub will enable users with a stake in the stability and health of their systems to take advantage of the comprehensive diagnostic tools offered by the new platform to increase understanding and decrease MTTR thanks to usage-based pricing and no-seat licences.

Being the first Wi-Fi 7 access points in the market to embrace open standards, all Wi-Fi 7 variants will come pre-installed with open source software. The introduction of Wi-Fi 7 will primarily support the country's upcoming 5G rollout, particularly for enhancing inside coverage. Additionally, the technology will make it easier to construct a variety of apps because of its increased throughput, dependable network performance, and lower latency.

Elon Musk may turn Twitter into an open source platform that encourages free speech

Elon Musk's Signal text exchanges, which were made public in advance of the trial, reveal his goal to turn Twitter into an open source, democratically beneficial platform for free speech. Elon Musk had conversations with a few people in an effort to determine the value of the company before making the US$ 44 billion offer to purchase Twitter.

His Signal text communications with a number of well-known figures in business and technology have just been made public. Mathias Döpfner, the CEO of Axel Springer, the parent company of *Business Insider*, texted Elon Musk at one point, pleading with him to buy Twitter and promising to "manage" the firm on his behalf.

Since Musk works to promote freedom of speech in the broadest sense, it is obvious that these remarks are exactly what got him. People who raise crucial problems are frequently silenced on platforms like Twitter. Many Twitter users who backed Musk noticed that their tweets were being deleted. These tweets earned a small number of likes and responses despite interaction statistics that were high (often in the thousands). Additionally, it was observed that some tweets, particularly those that were critical of Twitter, were displayed to a group of Twitter users entirely unrelated to the concerns or problems they were discussing and who were not followers of the account that sent them.

Musk and former Twitter CEO Jack Dorsey had another text conversation that showed Musk was sincerely dedicated to making Twitter a high-quality and open platform. Dorsey reportedly proposed that Twitter employ an open source protocol, similar to that used by the Signal messaging platform. Musk promised Dorsey that, if possible, he would work towards it.

Red Hat rolls out OpenStack Platform 17

OpenStack, an open source Infrastructure-as-a-Service (IaaS) cloud, is still seen by some as being obsolete. They are utterly mistaken. OpenStack is the most widely used open source cloud, according to *Statista*. The fact that OpenStack is the lifeblood of the telecom industry, including Verizon cellular and 5G networks, is even more significant for Red Hat. So it should come as no surprise that Red Hat highlighted the relationship between OpenStack and telecom while announcing Red Hat OpenStack Platform (RHOP) 17 at the Mobile World Congress in Las Vegas this year.

Red Hat also emphasised how closely it is collaborating with its Kubernetes based OpenShift release with this release. Some people believe that OpenStack and Kubernetes are competitors. They aren't. Red Hat is demonstrating how they may cooperate to benefit from both virtual machines and containers.

"We have found over the last couple of years that the conversation is not a religious dispute between the two platforms but rather where their workloads are served best," said Maria Bracho, lead product manager for OpenStack. Therefore, a virtualized control plane that uses OpenShift is now supported by OpenStack version 17. This makes it possible for customers to run virtual machines (VMs) and Kubernetes based cloud native services on the same cloud.

Dynamic resource allocation is another feature of the revised OpenStack 17 that supports "lighter" deployments. These are deployments that don't need as many services, such as edge computing.

The new RHOP will primarily appeal to the telecom industry, but it is still a reliable platform for other use cases. These consist of edge deployments, private clouds, and public clouds. Anyone who prefers to manage their own cloud over entrusting their services to a hypercloud provider like Amazon Web Services, Azure, or Google Cloud should give RHOP some thought.

Google launches OpenXLA, an open source ML compiler project

Google Cloud has introduced the OpenXLA open source machine learning compiler ecosystem at its Next '22 conference. A community-led and open source ecosystem of ML compiler and infrastructure projects, OpenXLA has been created by Google in collaboration with AMD, Arm, Meta, NVIDIA, AWS, Intel, Apple, and other AI/ML developers. By allowing developers to select the frameworks and hardware they want for their ML projects, OpenXLA attempts to address the problem of framework and hardware incompatibilities, which can hinder the development of machine learning algorithms.

The OpenXLA project's goal is to enable effective lowering, optimisation, and deployment of machine learning models from the majority of major frameworks, including PyTorch and TensorFlow, to any hardware backend (particularly CPUs, GPUs, and ML ASICs), through collaboration with major ML frameworks and hardware vendors.

Its objectives are: promote XLA industry collaboration and create a thriving OSS community; exchange ideas and solicit input on OpenXLA's technical direction to make sure it satisfies the requirements of its key users and contributors; create a new XLA repository or organisation that is independent of hardware and framework, has infrastructure to more easily accept PRs, and has independent build and test; and ensure that the removal of XLA from TensorFlow causes the fewest possible disruptions for current users and contributors.

The XLA compiler, which was created to simplify modelling in TensorFlow by accelerating the training process and lowering overall memory usage, will be the first goal of the new community project, said Sachin Gupta, Google VP and GM of

The Clearing House joins OIN as a community member

The Clearing House (TCH) has joined as a community member of the Open Invention Network (OIN), which was founded to protect open source software (OSS) and is currently the largest patent non-aggression group in history. TCH is reaffirming its commitment to OSS as an enabler of cutting-edge financial services payment platforms. It runs US-based payments networks that clear and settle more than US$ 2 trillion in payments daily through wire, ACH, check image, and real-time payments.

Through royalty-free cross-licensing of Linux System patents, the OIN community upholds patent non-aggression in core Linux and related open source technology. Similar royalty-free licences are granted for Open Invention Network patents to any company that pledges not to oppose the Linux operating system with its own patent claims. On *http://www.j-oin.net*, the OIN licence can be signed electronically

"Advancements in financial services and FinTech increasingly rely on open source technologies," said Keith Bergelt, CEO of Open Invention Network. "As the most experienced payment company in the US, and a keystone for the financial services industry, we are pleased that The Clearing House is committed to patent non-aggression in core Linux and adjacent open source technologies."

"We remain focused on integrating the best technologies in our payment networks, which help to deliver services that our banks' customers value and trust. We enable all financial institutions to offer safer, faster, and smarter digital transaction services, and open source is a significant innovation driver in these offerings," said Sean Reilly, senior vice president and associate general counsel at The Clearing House. "We are pleased to join the Open Invention Network and support its role in protecting open source software."

Intel and DataStax partner to enhance the performance of Apache Cassandra and Apache Pulsar

Real-time data startup DataStax has announced a strategic partnership with Intel to improve the performance of systems based on the Apache Cassandra database and the Apache Pulsar messaging and streaming technology on the most recent Intel Xeon Scalable processors. In addition to providing enterprise and cloud solutions based on these potent open source initiatives, the companies intend to make improvements to each project. DataStax has been chosen for the Intel Disruptor Initiative, in which Intel works with a limited group of cutting-edge businesses that are redefining innovation.

Largely scalable databases, such as Apache Cassandra, function best on machinery designed specifically to support database workloads. Through this partnership, Intel and DataStax are working together to provide Apache Cassandra with CPUs that can support this utilisation in order to contribute to its long-term success. Apache Cassandra and Apache Pulsar's strength, along with Intel's long-lasting processors, will pave the path for efficient database operations and quick access to crucial data.

The Intel-DataStax partnership will also concentrate on introducing the most recent Cassandra and Pulsar technologies to academic institutions around the world in order to speed up innovation and train the next generation of programmers and data scientists on these potent open source initiatives.

With well-known products like Astra DB, DataStax's cloud database built on Apache Cassandra, the company is already a significant supporter of the Cassandra and Pulsar open source projects. Astra DB makes it simple for customers to access and utilise all of Cassandra's capabilities without having to deal with the hassle of managing infrastructure. End-to-end security, global scalability, and streamlined operations are features of this serverless platform.

infrastructure, in a blog post. In order to create StableHLO, a portable ML compute operation set that serves as a portability layer across machine learning frameworks and compilers, OpenXLA will endeavour to isolate the compiler from TensorFlow.

Open RAN platform for simulation of 5G networks is now available

The first open source platform for end-to-end simulation of 5G wireless networks, powered and optimised by Open Radio Access Network (Open RAN) components and ns-O-RAN (Open Radio Access Network Simulator), has been made available to the public by researchers at the Institute for the Wireless Internet of Things (WIoT) at Northeastern University.

The O-RAN standards-compliant Near-RT RIC is interfaced with using the ns-O-RAN emulator, which combines ns-3 with O-RAN-compliant features including the E2 application protocol and service models. This project is the product of a partnership between Mavenir, a member of the WIoT Industry Consortium and the Open6G initiative led by WIoT, and the Institute for the Wireless Internet of Things (WIoT) of Northeastern University.

The widely used open source network simulator ns-3, which simulates all the layers of the RAN protocol stack as well as core network operations and mobile traffic applications, and its 5G module from the University of Padova and NYU, which enables the LTE-5G NR dual connectivity RAN protocol stack on ns-3, serve as the foundation for the ns-O-RAN simulator.

With the help of extensive modelling of the entire 3GPP RAN protocol stack and large-scale 5G simulations based on 3GPP channel models, ns-O-RAN has been conceived and developed to enable the integration of O-RAN software, such as Mavenir's Near-RT RIC. This enables RAN key performance metrics (KPMs) data collection at scale, in various simulated circumstances, and with various applications (e.g., multimedia streaming, web browsing, wireless virtual reality, etc). Two E2 service models (E2SM), E2SM-Key Performance Metrics (KPM) monitoring and E2SM-RAN Control (RC), which enable a closed-loop control, are implemented by ns-O-RAN and support an O-RAN-compliant E2 interface (for example, of traffic steering and mobility).

Dutch minister in favour of using open source for digital ID wallets

The Dutch minister for digitalization, Alexandra van Huffelen, recently spoke about the need for open source in promoting ID wallets as secure and user-friendly.

During an appearance on news outlet *Euractiv's* 'The Tech Brief' podcast, Van Huffelen stated that The Netherlands has reached an agreement on the usage of open source in most or all components of a digital ID wallet.

Everyone involved in a digital wallet programme gains trust as a result of the transparency. The software is superior to closed source programmes since anybody may access the source code and make changes. Open source software may be safer for the same reason. According to van Huffelen, it is also more effective. She said that she "believes in not being super-naive," echoing decades of prior justifications for open source.

Microsoft Research open sources Project FarmVibes

In an effort to increase farmers' yields while lowering costs and assisting the agriculture sector in accepting responsibility for its role in climate change, Microsoft Research is open sourcing its agricultural data and networking technology. The technology platform, known as Project FarmVibes, incorporates the cloud provider's most recent research in precision and sustainable agriculture, according to a blog post by Microsoft's Jake Siegel. This builds on data analysis and integration work done with Microsoft's clients like Land O' Lakes and Bayer.

The algorithms are intended to serve as a decision-making tool at each stage of farming, from sowing the seeds through reaping the harvest. According to Siegel, this involves forecasting temperature and wind speeds, determining the ideal depth to plant seeds based on soil conditions, advising on crop types, and following best practices to keep carbon sequestered in the soil. It also involves determining the right amount of fertiliser or herbicide to use on which areas of the fields.

The artificial intelligence (AI) component of Microsoft's FarmVibes project includes features like Async Fusion, which combines sensors and satellite imagery to create maps of soil moisture and nutrient heat; SpaceEye, which uses AI to remove clouds from satellite imagery; DeepMC, which uses sensor data and weather forecasts to predict the temperatures and wind speeds of a farm's microclimate, and a "what if" analysis tool that speculates on how farming practices might change.

Andrew Nelson, a fifth-generation farmer and software engineer, has tested the Project FarmVibes suite on a 7,500-acre farm under his management. Nelson collaborated with Microsoft Research to evaluate the technology portfolio. He said that the project's AI will improve damage control capabilities while also saving him time and money in agricultural settings.

Nelson is now piloting FarmVibes. Connect, which will utilise the unused TV white space spectrum to provide internet connectivity to rural and remote locations. He is also experimenting with FarmVibes.Edge to cleverly compress the enormous volumes of data collected by drones. This approach helps to focus on one specific problem, such as weeds in a field, while ignoring other issues.

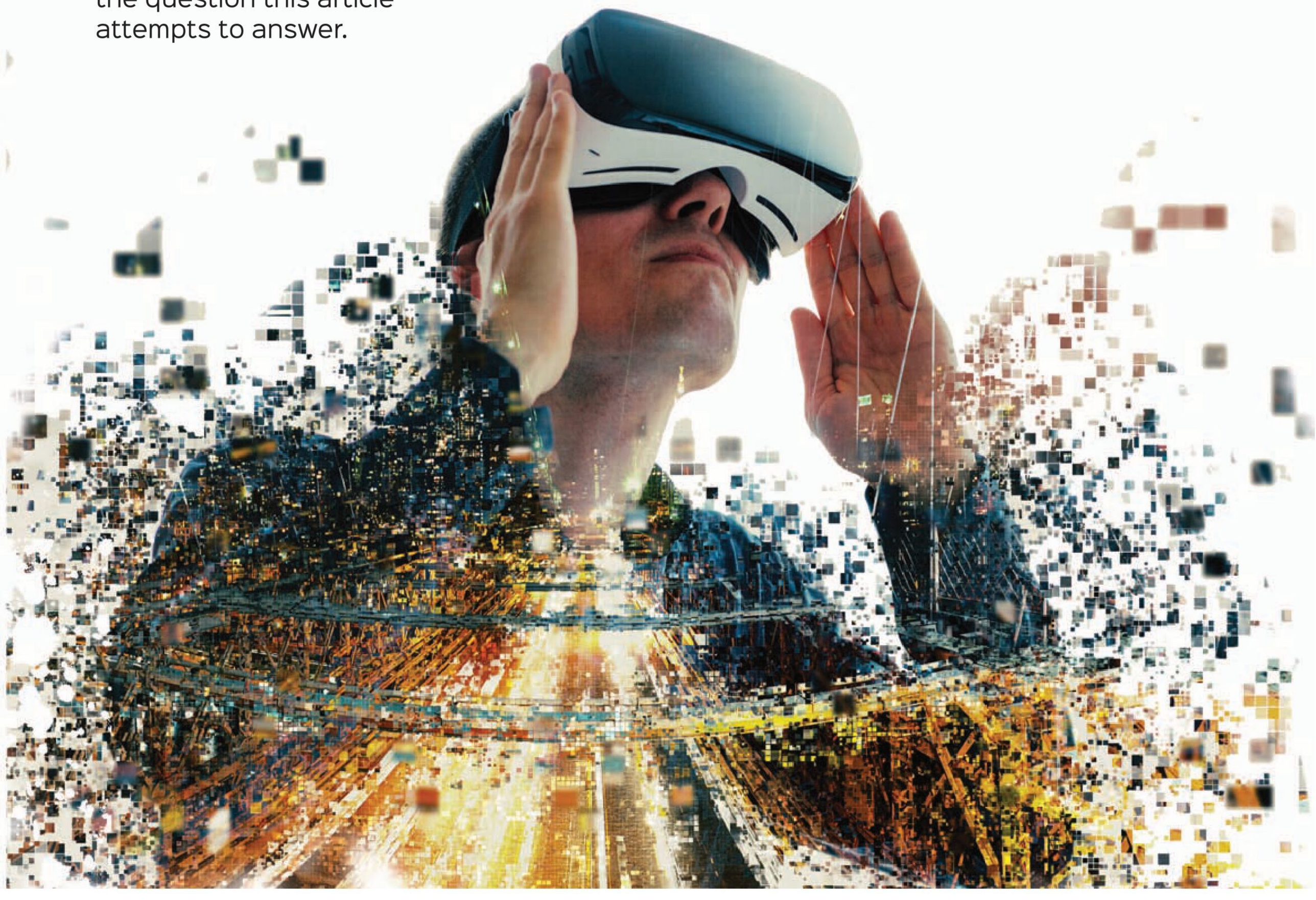
Microsoft intends to open source additional FarmVibes parts related to edge and connectivity services in the future, in addition to the AI.

For more news, visit
www.opensourceforu.com

The Importance of Open Source in the Metaverse

The Metaverse is no longer fiction...this virtual world is set to take over our lives, making people all over the globe connect with each other in ways never experienced before. But how relevant or important is the role of open source in this new virtual world? That is the question this article attempts to answer.

The Metaverse has four pillars, namely, virtual spaces, virtual assets, virtual interactions and digital payments. In this new virtual world, co-workers or students can gather for a chat over coffee irrespective of what part of the globe they belong to. Musicians or artists can also interact with fans from around the world in a large digital venue, which would be impossible in the physical world. Today, conferences can reach new audiences unlike before. All this can be achieved due to 5G, augmented reality (AR), virtual reality (VR), and edge computing. These ultra-reliable, low latency services ensure the rendering of 3D graphics and content, and make virtual experiences faster. "Open source will play a huge role in the Metaverse, and will help it evolve faster," says Dr Lokesh Boregowda, director, Samsung R&D Institute India.



A need for standards

"It becomes very interesting for us to witness this seamlessly connected virtual world as part of the Metaverse. It opens the door to a wide number of possibilities and we have yet to see how far we can truly go with it. It is important to adopt at scale; otherwise, it all falls flat. We have seen some of the standards that ensure interoperability, such as Razer's open source virtual reality (OSVR). We have seen that it encourages experimentation with the development of headsets as well. But if you look at the flip side, on the device layer you will see the OpenXR standard being badly adopted by manufacturers," says Vinay Rajagopal, ISV technology lead architect for partner ecosystem at Red Hat.

"Encouraging the development of a decentralised, distributed and interoperable virtual world built on the foundation of open standards also creates multiple opportunities. We have already started seeing a proliferation of 3D environments that have been created by the users themselves. This is where the blockchain has become imperative for the details surrounding a transaction and the digital assets or the avatars. It has become important for us to put these assets on multiple ledgers, while making sure that you understand who has built, sold or purchased a particular digital asset. This is important from a legal aspect too," he adds.

"Open source is the ground zero for the development of technologies like the Metaverse. All the biggest projects in the last five to ten years that have been able to garner the most attraction are the ones that are open sourced — artificial intelligence, blockchain, and containerisation, to name a few. Open source comes with a great number of benefits. It allows you to collaborate with the smartest minds from every corner of the world; your work gets constantly reviewed and used, and there is a constant feedback loop," says Gandhali Samant, director, cloud solution architecture, digital and app innovation, Microsoft, India. "A good aspect of working with open source is that there will always be components ready to be used, which saves a lot of time as you don't need to start from scratch. As the Metaverse continues to gain traction, more tools and platforms will appear to assist the developers."

"Even though open source comes with a lot of benefits such as quick development and large communities, it is important to note these communities own the software itself. In 2014, a bug was detected in Unison. Its source code had been open for so long that it made it hard to estimate the time hackers had been exploiting it," says Dr Sumit Kalra, assistant professor in the Department of Computer Science and Engineering, Indian Institute of Technology, Jodhpur. "Today whatever we do with the Metaverse is going to have a tremendous impact on the future of its development. If we are going to continue Metaverse development with open source then it is crucial for us to adopt open standards."

Dr Lokesh Boregowda, director, Samsung R&D Institute India

Vinay Rajagopal, ISV technology lead architect for partner ecosystem at Red Hat.

Gandhali Samant, director, cloud solution architecture, digital and app innovation, Microsoft

Dr Sumit Kalra, assistant professor in the Department of Computer Science and Engineering, Indian Institute of Technology, Jodhpur

The right hardware

Syed Mohammed Khasim, senior manager, software development - Sitara ARM MPU, Texas Instruments, notes: "In the last couple of decades I have seen the evolution of open source, starting from Linux. It all essentially started for shared software development. The cost of developing software turns out to be a big challenge

Syed Mohammed Khasim,
senior manager, software development - Sitara ARM MPU, Texas Instruments

Biju K. Nair,
founding partner, Legalitech

as it requires a unique skillset. You need to have proper architecture to build software, a design and the necessary components that will allow you to get started with the coding. You then have to validate and commercialise everything before it goes into the hands of developers and, ultimately, the consumers themselves. It all boils down to sufficient investment. But there is a limit to which everyone can invest in software and often it is only for a particular period of time."

"Open source is no free lunch as somebody is indeed paying for it. When we use the term open source, we mean that we are enabling a shared software development model where we want to collaborate with similar entities that want to invest in the development of software. Hardware is also a fundamental aspect of building software, as without proper hardware in place software cannot be developed. We need low cost hardware that can be afforded by everybody. This is why the Intel ecosystem picked up, because everyone used a personal computer and these systems did not require a huge investment. Over time, embedded systems have also evolved and reduced a lot of the burden of software development. With the entry of Android all the necessary components of open source development have been stitched together," he says.

"Today, things have evolved tremendously. We are seeing multi-core architecture and multi-core programming even in a simple smart watch today. It requires a great amount of skill to do it all. Not only are you not going to work for free while working on open source software, but your contribution to open source will also provide you with the relevant validation and introduce you to new features. Hence, companies always try to build an ecosystem for open source development as it tends to be more robust and reliable. It is time to define the hardware ecosystem that we are going to enter into, and the kind of software skillsets that are required, in order to assess what we need to build the community around. We also need to find whether it can scale and sustain on its own without a huge sum of money being pumped from big organisations," adds Khasim.

The legal angle

Biju K. Nair, founding partner, Legalitech, says: "There is also a very important legal aspect to open source from an intellectual property (IP) perspective. IP is different in the Metaverse from how it is considered in the traditional ecosystem. So this creates opportunities to maintain and safeguard the IP within the Metaverse. In the last two years the whole globe has woken up to the vulnerability issues. Everyone today, including the US government, is trying to spend more money on the detection of vulnerabilities and securing them. Working together as a community can allow us to solve these issues together."

The advantages of open source in the Metaverse

So what is the main foundation and strength of open source, which will play a key role in the development of the Metaverse?

"Open source is both flexible and agile. It is important to note that the advantages of using open source far outweigh the disadvantages. GitHub is the largest open source community in the world or, you can say, it is the home of open source itself. At GitHub our focus is to continuously improve the platform and add features so that the collaboration and open source projects can thrive," says Gandhali Samant.

"Today, every new security feature that we build is available for free for public repositories. It is the main source of developing open source software. If you look up the top 10 Metaverse projects on Google, you will find that they are all available on GitHub. The role of GitHub in the Metaverse will be very similar to the role that GitHub has played in the large scale success and adoption of technologies like TensorFlow for AI, or Docker and Kubernetes for containerisation, orchestration, Hyperledger, etc. GitHub will continue to provide the best platform for software development," she adds.

But are there any limitations and concerns for the collaborators?

According to Samant, "On one side we have multiple open source projects, while on the other we have tech giants like Google and Microsoft that are already investing big dollars in open source. No one company holds the monopoly for any component today. They are bound to come together and work, as each one

brings a different value to the table. We will see more standardisation across some of the underlying fundamental blocks to ensure compatibility. For instance, Linux started as an open source project while Microsoft Windows was totally closed source; yet, they still have a number of partnerships."

Data privacy

There are already a lot of copyright and patent infringement cases around data privacy as many open source projects are being used without the consent of the original creator. Although we have better control over such scenarios in the real world, this is not the case in the Metaverse as the person is not physically present there. It is not clear yet if the virtual property in the Metaverse will have any analogy in the physical world.

"When it comes to hardware, technology has evolved exponentially in terms of security, power and features. Every smart device today is connected to a network, which is also exposing it to various security threats. So semiconductor companies, too, are focusing on resolving these security threats," says Khasim.

There could be trademark issues too. Biju Nair says, "Let us say you have a trademark in the physical world but have no presence in the virtual world. Then the chances of you getting an injunction for trademark misuse are very low. Such a case has already been brought before a US court — E.S.S. Entertainment versus Rock Star Videos. The court ruled that the depiction of a physical world trademark in any video game does not infringe on the original trademark, as the video game is an artistic expression protected by the First Amendment. So you may not be protected in the Metaverse unless you have some sort of presence there."

Nair adds, "Companies need to revisit their existing agreements, see what rights have been provided to them, and plan accordingly. They also need to check if they have filed for the trademark for virtual goods or not, because if they don't do that they will be at risk of losing the rights for those goods in the Metaverse workspace."

There are a lot of legalities that need to be looked at and addressed separately in the Metaverse. Nonetheless, the way forward is exciting and interesting as there is still a lot to be explored and developed in this new virtual world. END

This article is put together from a panel discussion at 'Open Source Summit: Frontiers of Innovation 2022', an event co-hosted by Samsung and IEEE Bangalore.

Transcribed and curated by
Laveesh Kocher

Brain Inspired Computing
Moving from Known to the Unknown Worlds

Over the last few decades technology has integrated itself into our daily lives in ways which make us totally reliant on it for our day-to-day tasks. Undoubtedly, technology has only gotten smarter. Considering the trajectory it currently follows, it is only a matter of time until it is able to mimic the way we think and perform our daily tasks. Continue reading to know more.

We are living in an extraordinary time when we are trying to recreate a human brain in machine. Human brain is the most sophisticated organ that nature has given us. Building something which can equal that is a very difficult feat to achieve.

Over the last few years, a lot of studies and findings have been done but that does not seem to be enough by any means. Our limited understanding of this complex organ makes it even harder to mimic all of its functions precisely. Yet, we are determined to make something that can at least be close to the real thing.

Technology landscape

Figure 1 shows an analogue from the perspective of the technology landscape. It shows that a personal computer and a wired internet connection can be used to connect to the external world.

The horizontal dimension in the figure shows that with the help of larger systems we are able to integrate data altogether on a large scale storage and process that data through cloud. The vertical direction contains the devices and technologies that provide a plethora of functionalities.

For example, our cell phones can sense, display, communicate via wireless systems and the internet while providing a great computational power for data processing applications. The computing domain can change from a static and virtual environment to the real world which is highly dynamic in nature. By combining these two dimensions we envision intelligent systems.

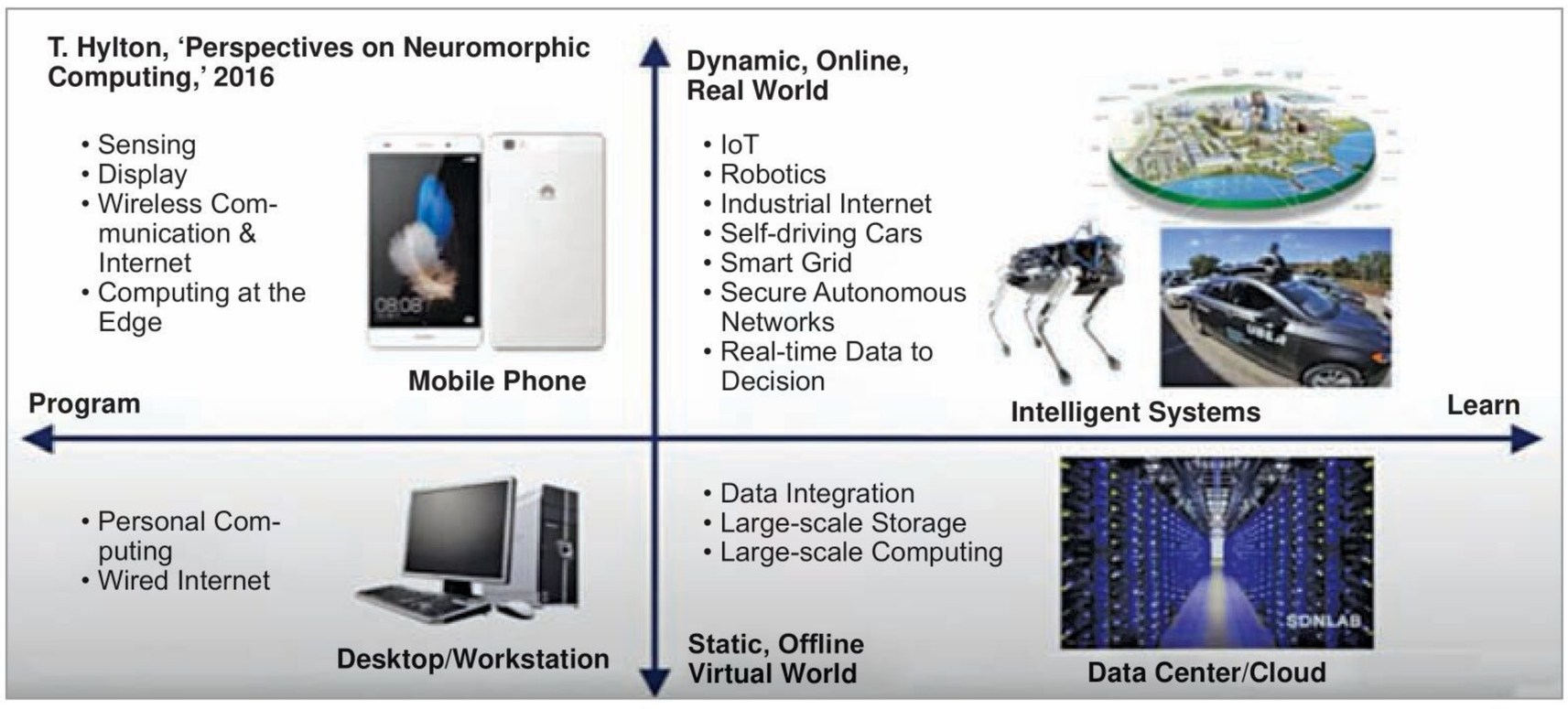


Figure 1: Technology landscape

Intelligence and artificial intelligence

The concept of intelligence has been discussed by people for many years, even as far back as the 18th century. According to Wilhelm Wundt, "Our mind is so fortunately equipped that it brings us the most important bases for our thought with our having the least knowledge of this work of elaboration." Clearly, we are curious about intelligence, but we do not know much about our brains and where does it actually come from.

People have become more curious about the intelligence itself through the years of exploration, which has led them to build intelligent systems we today call artificial intelligence (AI). The history of AI can be traced back to a summer workshop back in 1956. It is widely considered as the founding event of AI as a field.

According to Wikipedia, "It is a theory and development of computer systems able

to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.”

Evolution of computing power

The chart in Figure 2 is from Dr Raymond Kurzwell’s book that was published back in 1990, titled ‘The age of intelligent machines.’ This chart summarises the computing power measured as millions of instructions per second. The dots in the chart represent a curve.

This curve suggested that by the year 2020 the computing power could reach the intelligence level of monkeys and human beings. May be, right now this claim is too bold to go with. But the bottom line is that a sufficiently advanced computer program could exhibit human-level intelligence.

Major limitations

Some limitations have emerged from the intrinsic architecture of the computing system hardware. In the past thirty years, people building computing systems have had different objectives for improving the RAM and processors. They aimed to improve the execution speed for processors while the major goal for the memory part was to increase the storage density. The two components have a huge performance gap to this day.

Another important aspect is the limited power efficiency ceiling. The power efficiency ceiling is much lower than the efficiency provided by human brains. Human brain is indeed the most efficient computing machine there is. So, let us take a quick look at it.

The human brain, on average, contains 15 to 30 billion neurons. It essentially has very complex structures and offers complicated operations. But its power consumption is extremely low at around 20 to 35 volts.

Figure 3 shows that a human brain contains some gray matter for thinking and white matter for signal transitions.

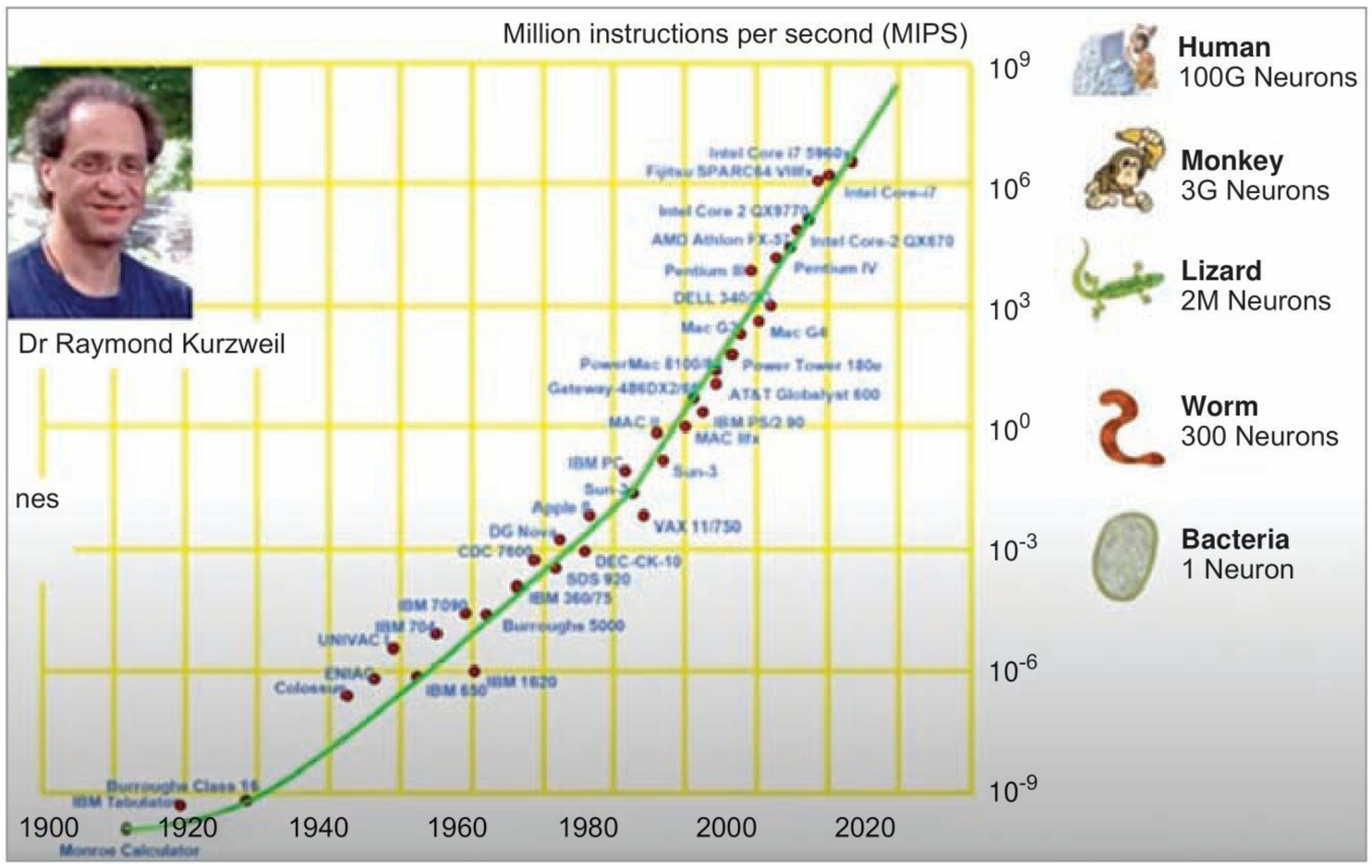


Figure 2: Evolution of computing power

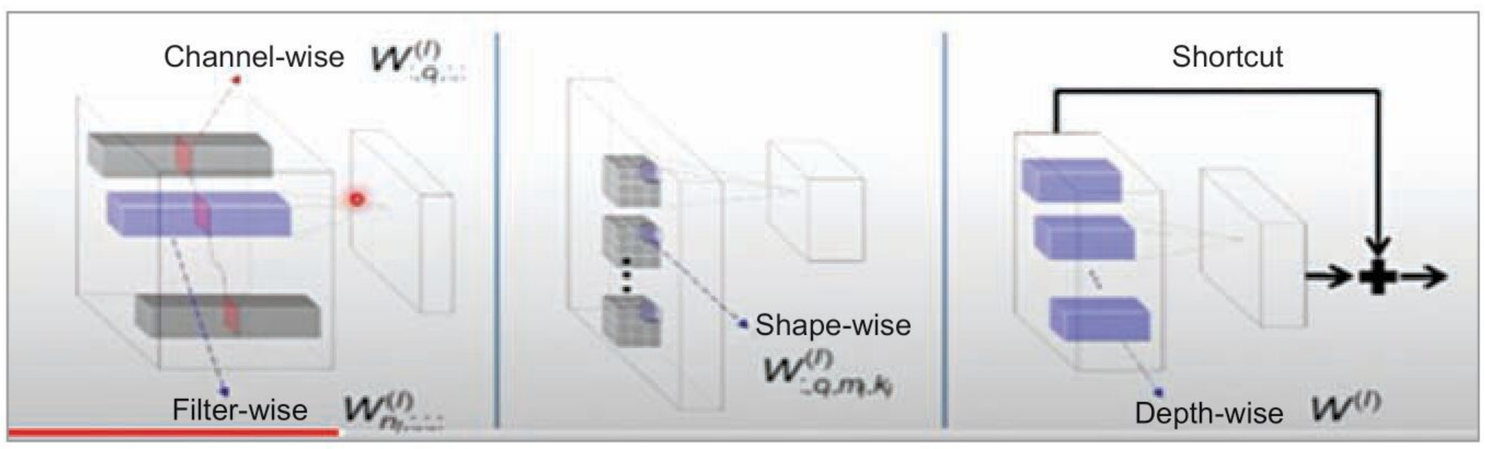


Figure 3: Structurally sparse deep neural network

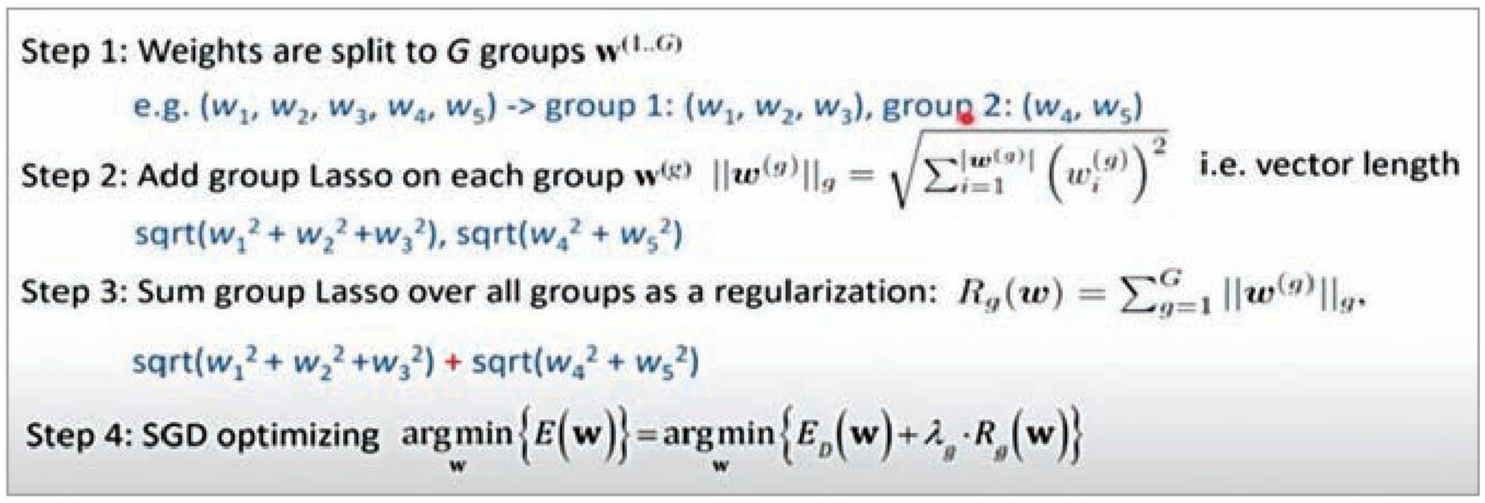


Figure 4: Group lasso regulation

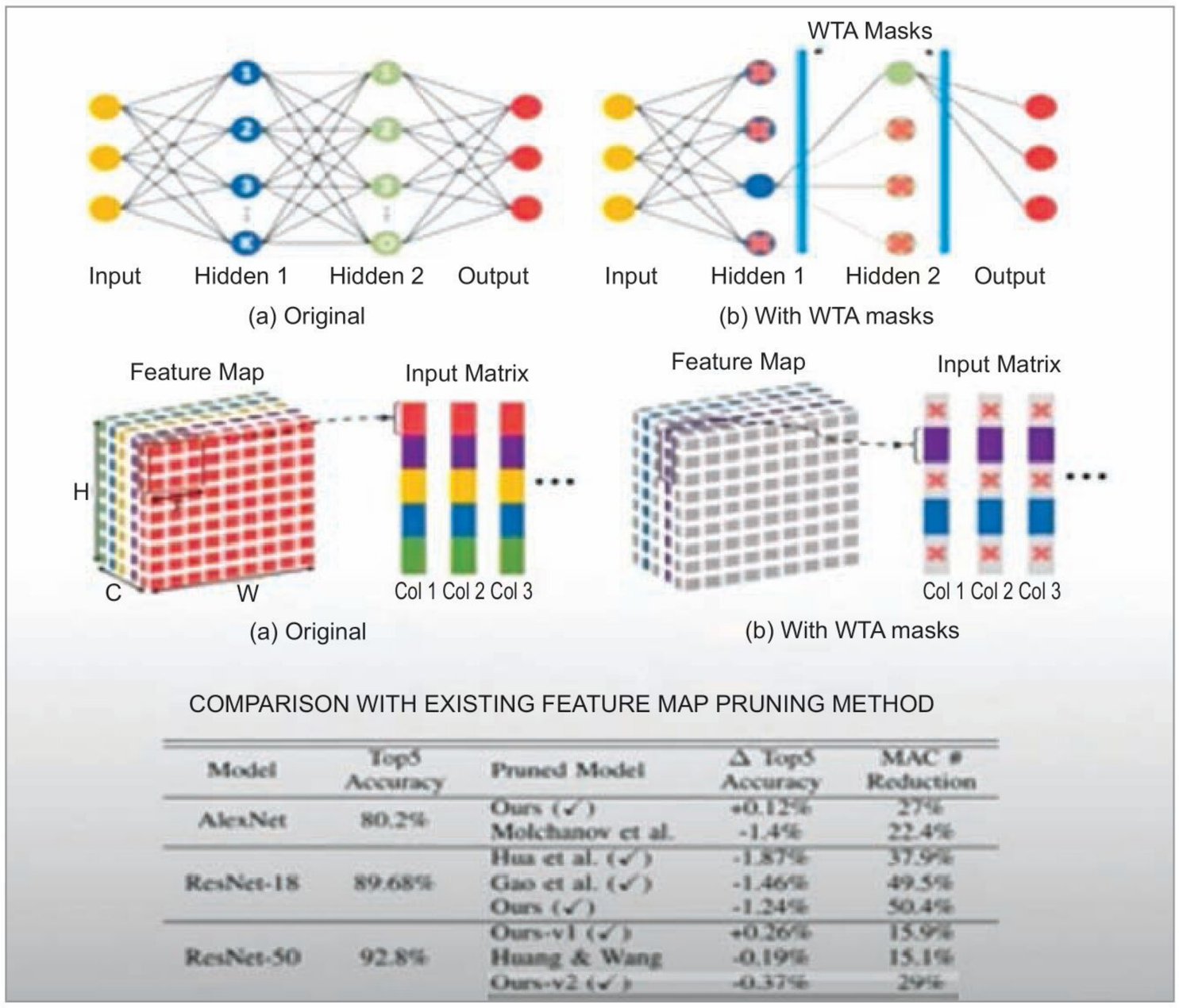
The gray matter consists of a neocortex, which contains six layers where signals travel within the layers. The fundamental elements of the neural cortex are neurons and synapses.

There are three characteristics that can be found in human brain but not in the computers:

Low power consumption: Only about 35 watts of power is consumed by the brain versus roughly 250 watts by a graphics processing unit (GPU) for rigid operations.

Fault tolerance: The cells within the human brains are replaced with new cells as the old ones die. The brain can even heal and continue its functionalities. A silicon based computer system cannot heal itself at all.

Lack of need for the programs: The human brain does not require a program in order to operate as it can learn and unlearn things. This is not the case with a computer.



Model	Top5 Accuracy	Pruned Model	Δ Top5 Accuracy	MAC # Reduction
AlexNet	80.2%	Ours (✓)	+0.12%	27%
		Molchanov et al.	-1.4%	22.4%
ResNet-18	89.68%	Hua et al. (✓)	-1.87%	37.9%
		Gao et al. (✓)	-1.46%	49.5%
		Ours (✓)	-1.24%	50.4%
ResNet-50	92.8%	Ours-v1 (✓)	+0.26%	15.9%
		Huang & Wang	-0.19%	15.1%
		Ours-v2 (✓)	-0.37%	29%

Figure 5: Comparison with existing feature map pruning method

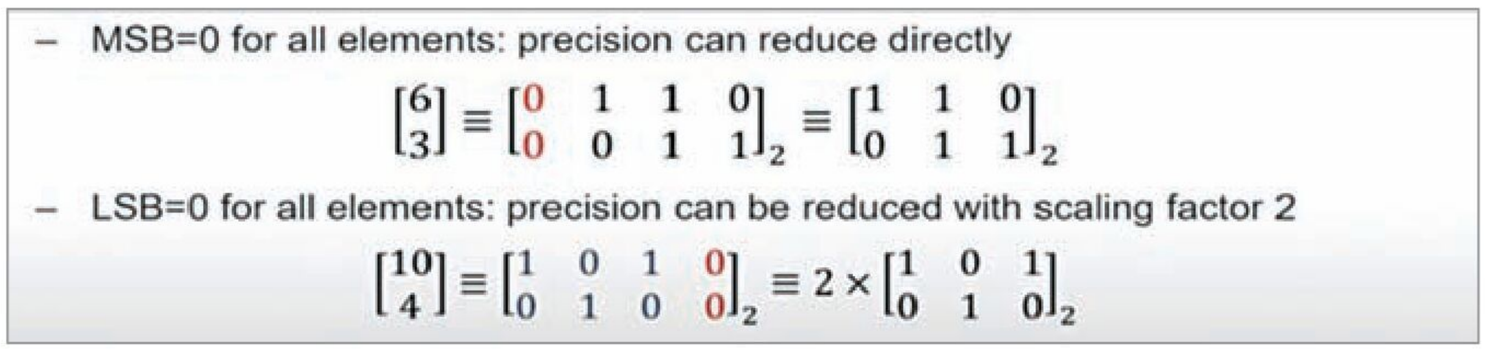


Figure 6: Mixing precision with bit-level sparsity

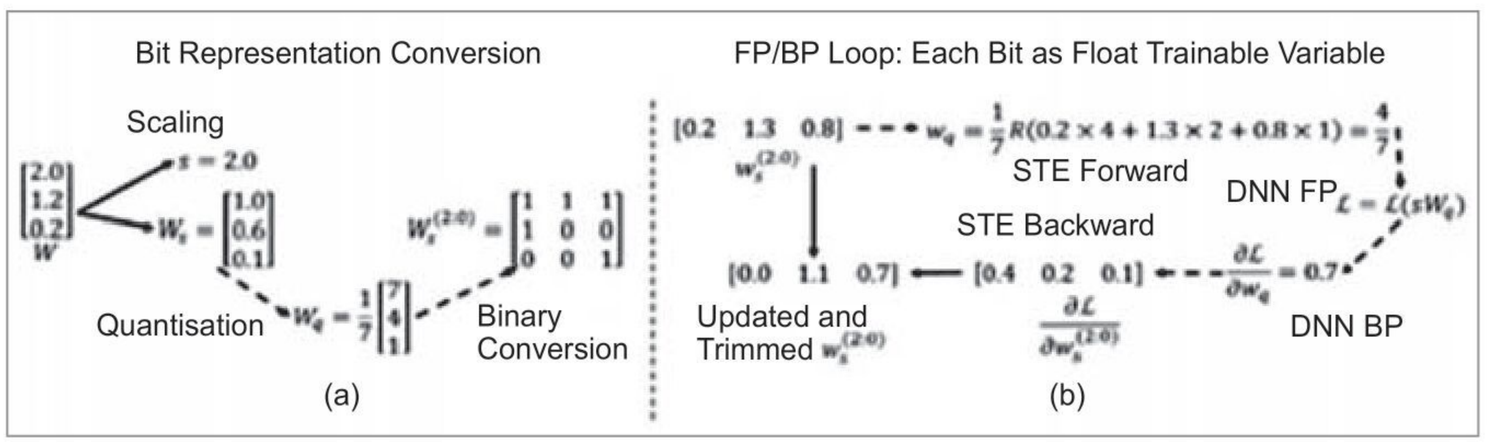


Figure 7: Bit representation conversion

Let us now look at pruning and sparsification for deep neural networks (DNN) acceleration. Why do we need to sparse DNN models? Some answers are:

- Because the ever-growing size brought challenges to the deployment of the DNN models.
- A lot of studies have shown that the parameters of DNNs are redundant. Many of them can be pruned and reduced so that we can reduce the computation and bandwidth requirement by eliminating that redundancy. This raises the question as to what solution do we have in order to achieve the goal.

When we talk about sparsification and DNNs, there are multiple ways to achieve the goal. We can sparse weight parameters through unstructured prunings or structured prunings. We can also consider activation sparsity where ReLU has been widely adopted.

Our lab has proposed bit-level sparsity, which basically decomposes the weight parameter into binary presentations that aims to remove unwanted zeros from the computation. Then there is inherent sparsity that transposes the convolutions and expands the flick necklace convolutional kernels. Hence, we do not have to involve those zeroes into the computation.

There is a need for structural sparsity because non-structured sparsity may not speedup traditional platforms like GPUs. If we are able to achieve structured sparse state, then we are printing out the weight parameters by rows or columns that will be more friendly to the hardware. These structures can be squeezed into much smaller matrices for the operations. Let us see how we can do that.

Structurally sparse deep neural network

The key idea here is to form a group. A group can be any form of a weight block, depending on what sparse structure you want to learn. For example, in CNNs, a group of weights can be a channel, a 3D filter, a 2D filter, a filter shape fibre (a weight column), and even a layer (in ResNets), as shown in Figure 3.

Once the groups have been defined, we can apply lasso regulation to the groups. For example, Figure 4 shows a weight vector with five elements, which we divide into two groups—one with two and the other with three elements.

We can apply group lasso on each group by following the equations illustrated in the figure itself. Then we sum the group lasso over all groups as a regulation and add it as a part of the optimisation terms of the overall optimisation objectives. This forms the procedure of our method called the Structure Sparsity Learning.

Sparsity-inducing regulariser for DNN involes the following:

L1 regulariser (sum of absolute values)

- It has been used for sparsity since 1996.
- It is differentiable, convex, and easy to optimise
- It is proportional to the scaling, which will lead to scaling down of all the elements with the same speed, which is undesired

L0 regulariser (number of nonzero values)

- It directly reflects the sparsity by definition
- It is scale invariant. However, our zero regulariser provides no useful gradients
- It needs additional tricks for applying on DNN pruning (stochastic approximation, ADMM), which makes the problem complicated. This leads to certain complexity in the design

Moving beyond lasso

We propose to move beyond L1 and L0 and aim to find the sparsity-inducing regulariser that is both differentiable and scale-invariant for pruning. At this stage the hoyer-square regulariser draws our attention. It is represented by the ratio between the L1 and L2, which is both differentiable and scale-invariant.

For element-wise DNN pruning, we can apply the square version of hoyer regulariser, which has the same range and similar minima structure as L0. We further extend the hoyer-square regulariser to the structure pruning settings.

We can apply hoyer-square regulariser over the L2 norm of each group of weight. The weight within the group can be induced to become zero simultaneously. Let us now look at the effect of the hoyer-square regulariser.

Regulating the activation sparsity: In operations, the activation sparsity is dynamic and the sparsity level is not identical for different inputs. Our group has proposed what is called the Dynamic Winner Takes All (WTA) dropout technique to prune neuron activations layer wise. Hence, the design introduces low overhead for the real-time interface (see Figure 5).

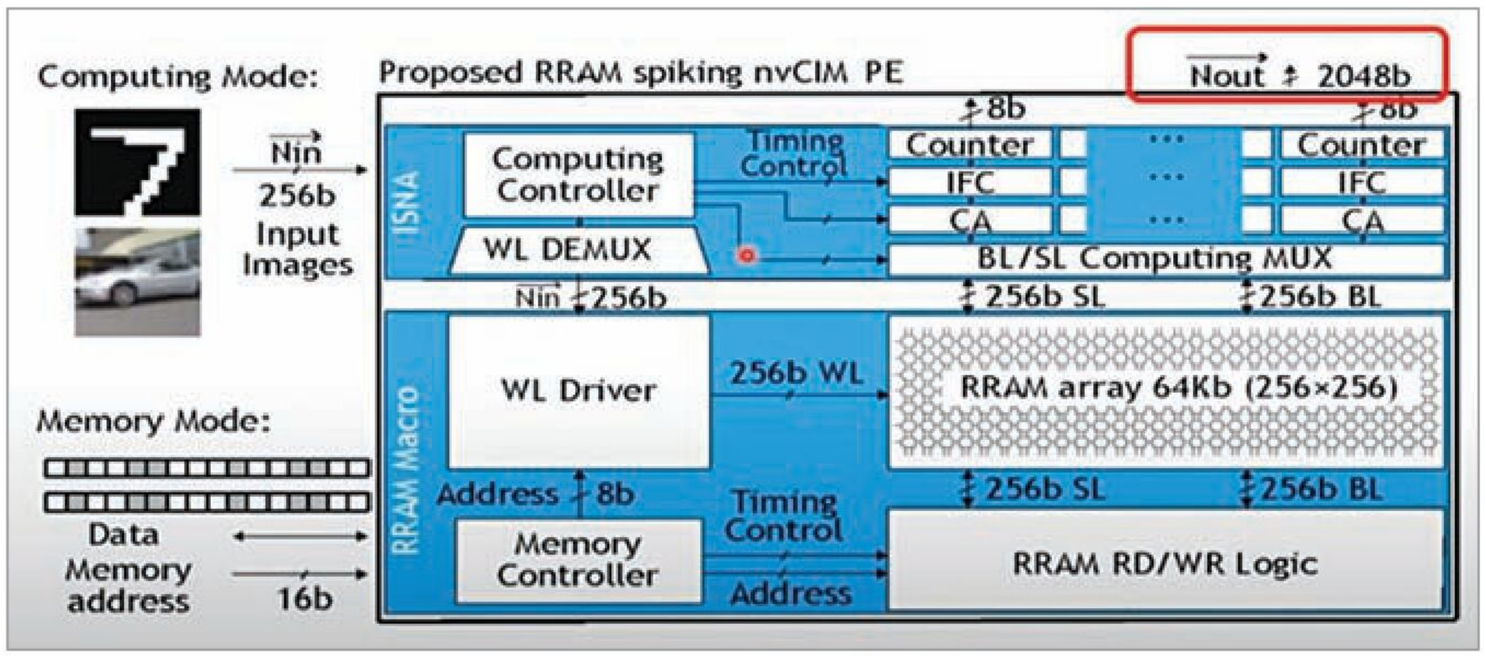


Figure 8: Processing engine with in-situ neuron activation

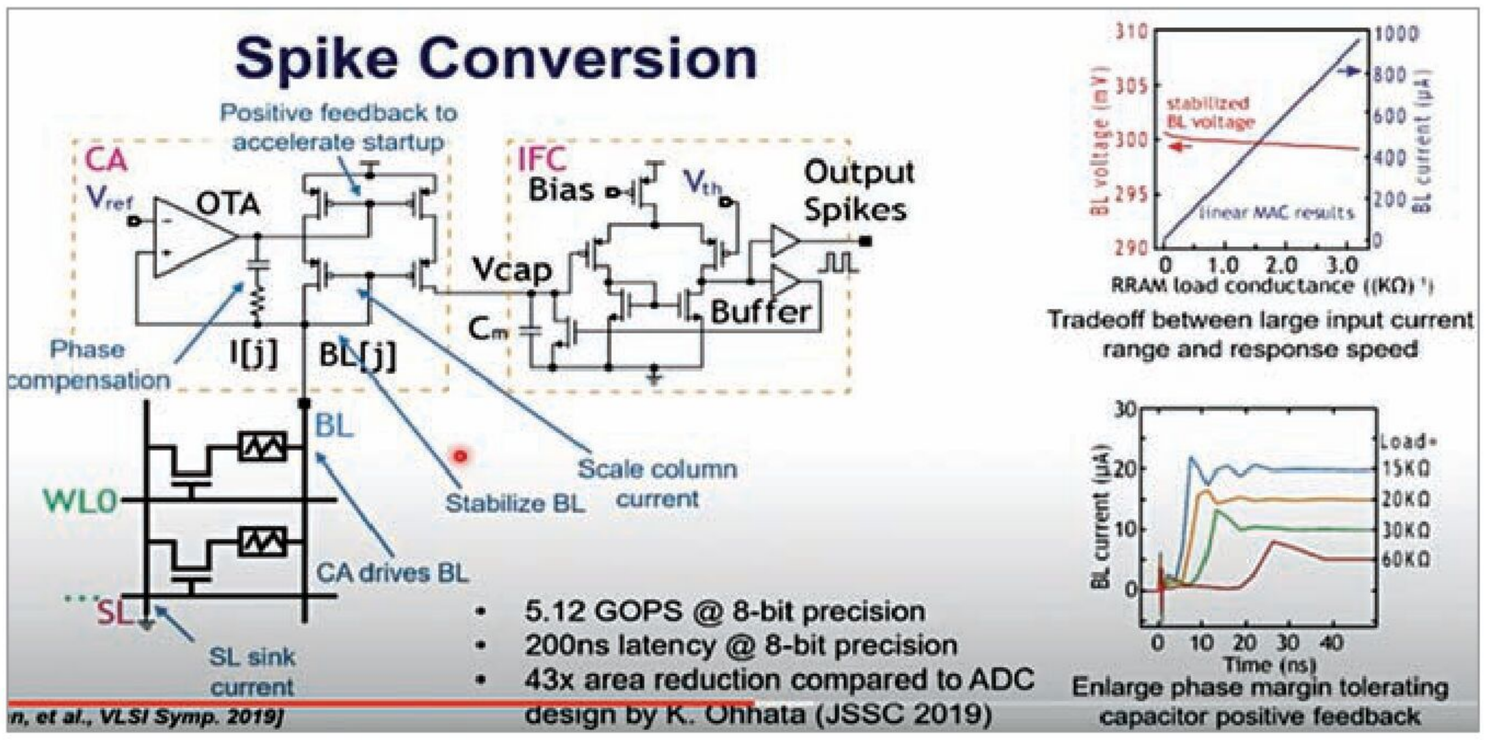


Figure 9: Spike conversion

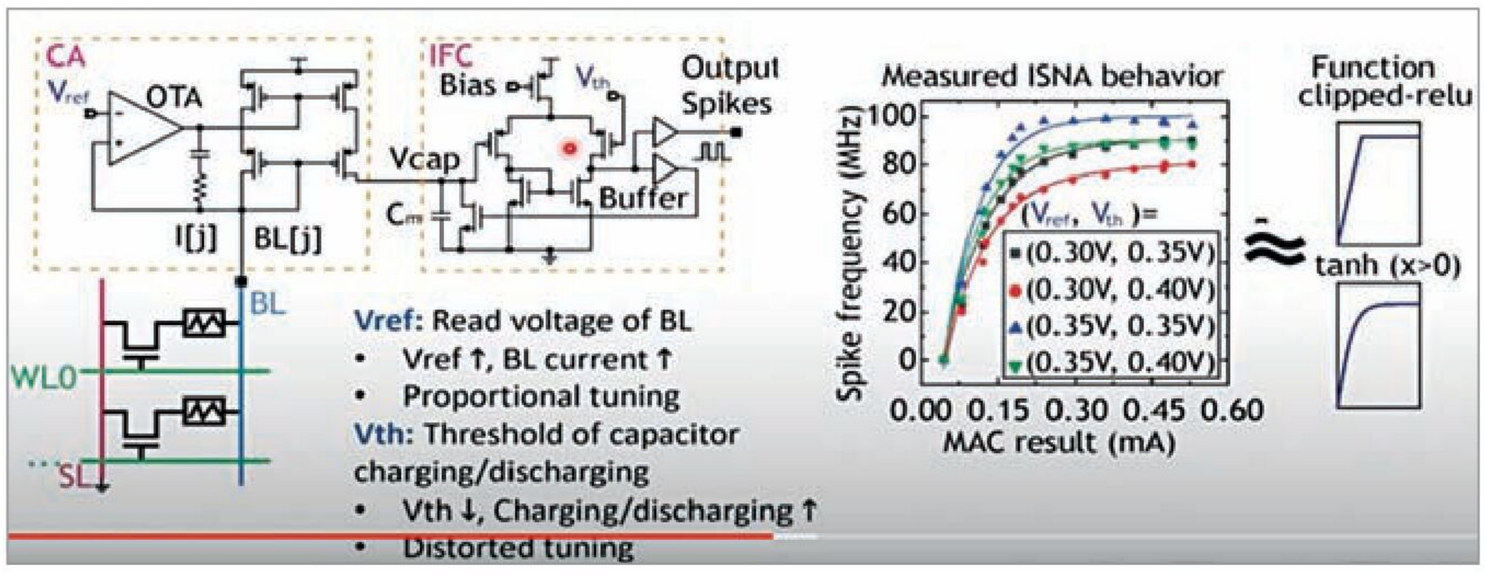


Figure 10: In-situ nonlinear activation function

Mixing precision with bit-level sparsity: We know that fixed point quantisation is an important model compression technique. In the previous research it was found that some layers can retain higher precision while others can be quantised to less precision. This is the whole idea of mixed precision quantisation.

For a quantised matrix, when the most significant bits of all the elements are zero simultaneously, we can remove the MSB directly (see Figure 6).

For bit representation we follow the dynamic scaling quantisation procedure. Here (see Figure 7) we use the element with the largest absolute value in each layer that is the scaling factor and uniformly quantise all the elements to embed the value in binary form.

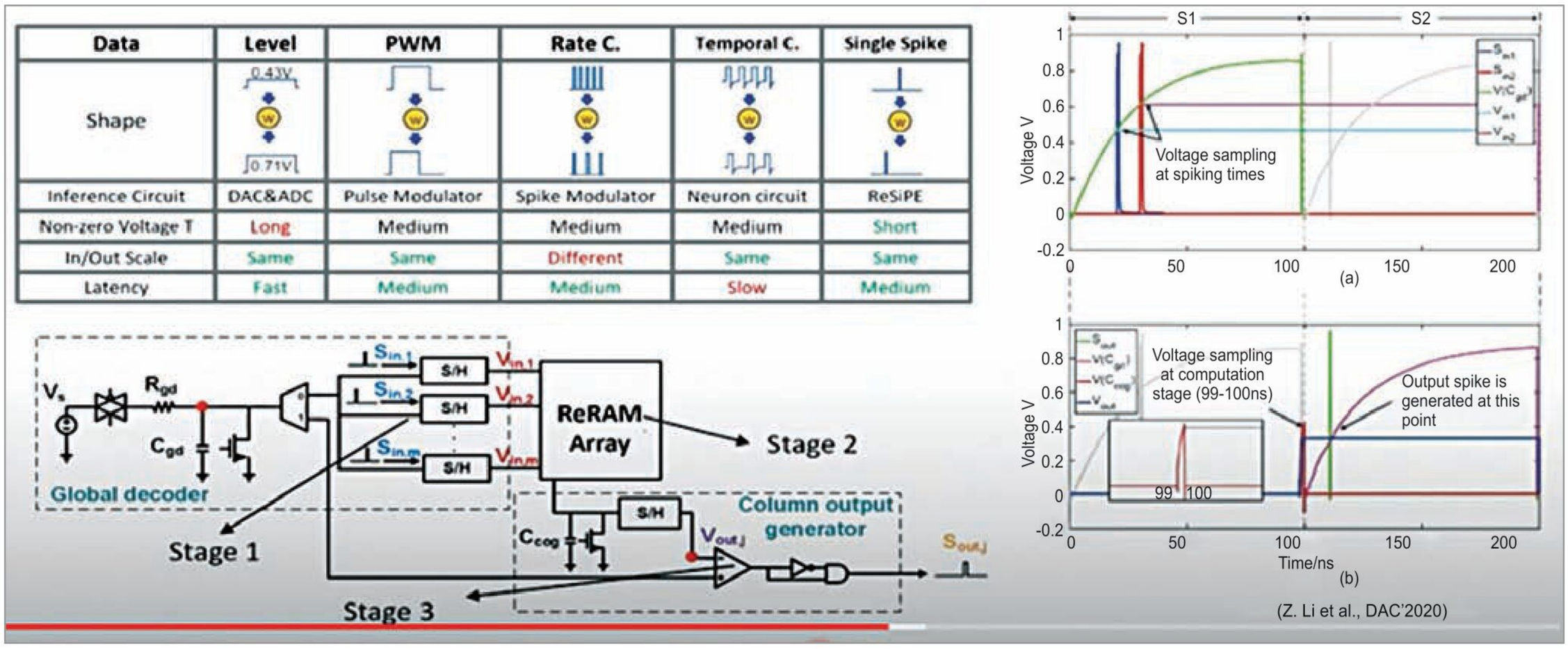


Figure 11: ReRAM based processing engine: ReSiPE

We consider the scaling factor 's' in Ws as an independent trainable variable. This way we are able to enable gradient based training of bit representation. During the training we allow each bit to take a continuous value between zero and two.

Memristor—rebirth of neuromorphic circuits: ReRam, memristor, or metal-oxide resistive random access memory is essentially for simple two-terminal devices. When we apply voltage or pass current through the devices, their resistance level could change. It is often also called the Programmable Resistance with analogue states.

The spike based neuromorphic design: We have moved to spike based neuromorphic design and minimisation of power consumption. It is very similar to the previous structure, but here we use the frequency of the spike to represent the input data amplitude. So, the information is encoded in the spikes instead of analogue voltage. Accordingly, fire circuits are used to produce output spikes along the bit line levels.

We measure the output as impulse number versus the input as the summation of the weighted inputs. The initial expectation was that these will follow a straight line like proportional relations, but the measurements show that it could get saturated. This is mainly because of the operation of the IFC (integrate-and-fire case), which have an upper bound.

Processing engine with in-situ neuron activation (ISNA): We integrate 64-kilobit RRAM arrays in this engine. The bottom part of this device is so much like an irregular memory array. On the top part we add our neural processing components (see Figure 8).

Spike conversion: The design still uses capacitors to generate spikes. Over here (see Figure 9), we use the output of the op-amp to form a positive feedback to accelerate the responses from the current amplifier. We use the current mirror to scale down the bit-line current, because in the real network applications, the current can either be several microamperes or several milliamperes—especially at the early stage when the device is not very stable and could result in a larger range of resistance.

In-situ nonlinear activation function: In the following design (see Figure 10) we have two knobs. The reference voltage can control the read voltage of the bit lines while the threshold voltage is used to control the capacitors along with charging and discharging speed. With these distant knobs we are able to slightly tune the shape of neural activation functionalities.

Configurable activation precision: We measured our chips on the application level and explored trade-offs between high precision weight and activation versus improved performance. Such exploration is useful for system-level, fine-grained optimisation and here are the results.

As mentioned earlier, the designs were fabricated by 150-nanometer complementary metal-oxide semiconductor (CMOS) and HfO RRAM devices. They contain 64-kilobit RRAM devices. The energy efficiency is also high at 0.257 pico joules (pJ/MAC).

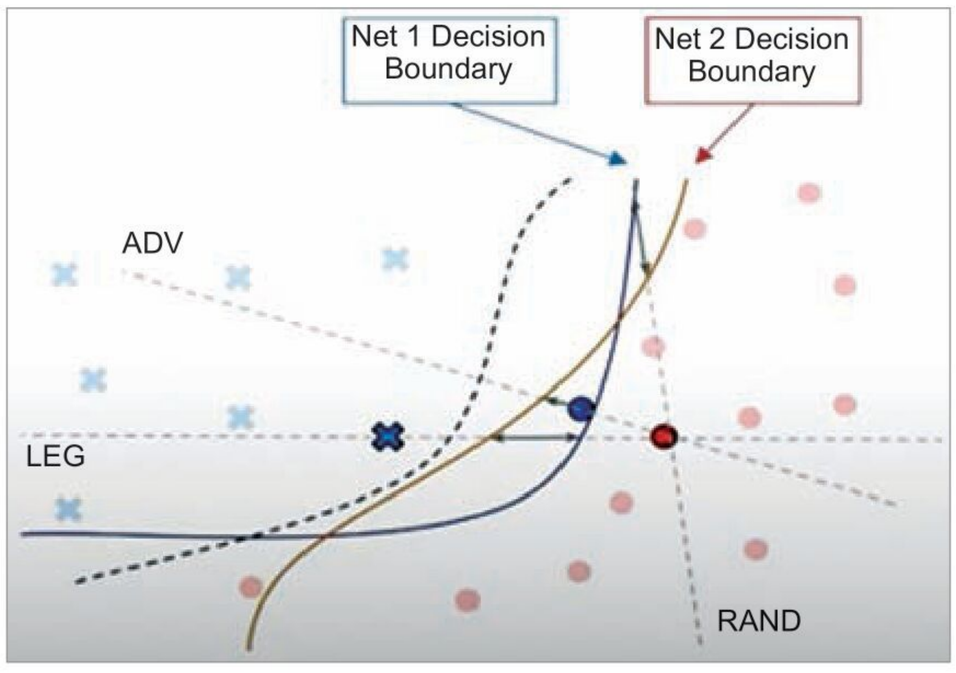


Figure 12: Transfer-based attacks

ReRAM based processing engine: ReSiPE

In the last two years we have moved to single spiking designs. The goal is to leverage the timing of the spike for data representation and the matrix operation. For that we divide the design into three stages.

In the first stage we translate the timing of the input spikes to an analogue voltage. Hence, we use a charging process of a capacitor Cgd for reference. When the input spikes 'S' then the voltage will be sampled at exactly the spiking time.

In stage two, the hinge signals are fed into the ReRAM crossbar arrays to perform matrix operation and during the computation the capacitor Ccog is charged continuously. The charging speed is determined by the bit-line current.

The last stage is the output stage where Vout is translated to upper spike timing. The voltage of Cgd is again used as the reference and it generates the Vout here (see Figure 11).

There are three important components that are needed while building machine learning techniques such as data, model, and application. All these elements are connected and operate together. Although we can say that a system is smart, it can still be attacked, hence, further creating a lot of security problems.

Many researchers have shown how well-trained deep learning algorithms can be fooled with adversary tags intentionally. Most of the work focuses on introducing minimal perturbation of the data, which results in a maximally devastating effect on the model. In the case of image based data, it works to disrupt the network's ability to classify with imperceptible perturbations to the images.

These attacks are of great concern because we do not have a firm understanding of the occurrences taking place inside these interpretable deep models. However, attacks may also provide a way to study the inner workings of such models. There are two types of attacks—the transfer based attacks and activation attacks.

Transfer based attack: Figure 12 shows two different models that are trained with the data from the same distribution. It implies the property of adversarial transferability. This is particularly relevant for a targeted attack. The region for a targeted class in feature space must have the same orientation with respect to the source image for the transferred target to be attacked successfully. In our set-up we train a white box model or source model on the same task as the black box or the target model.

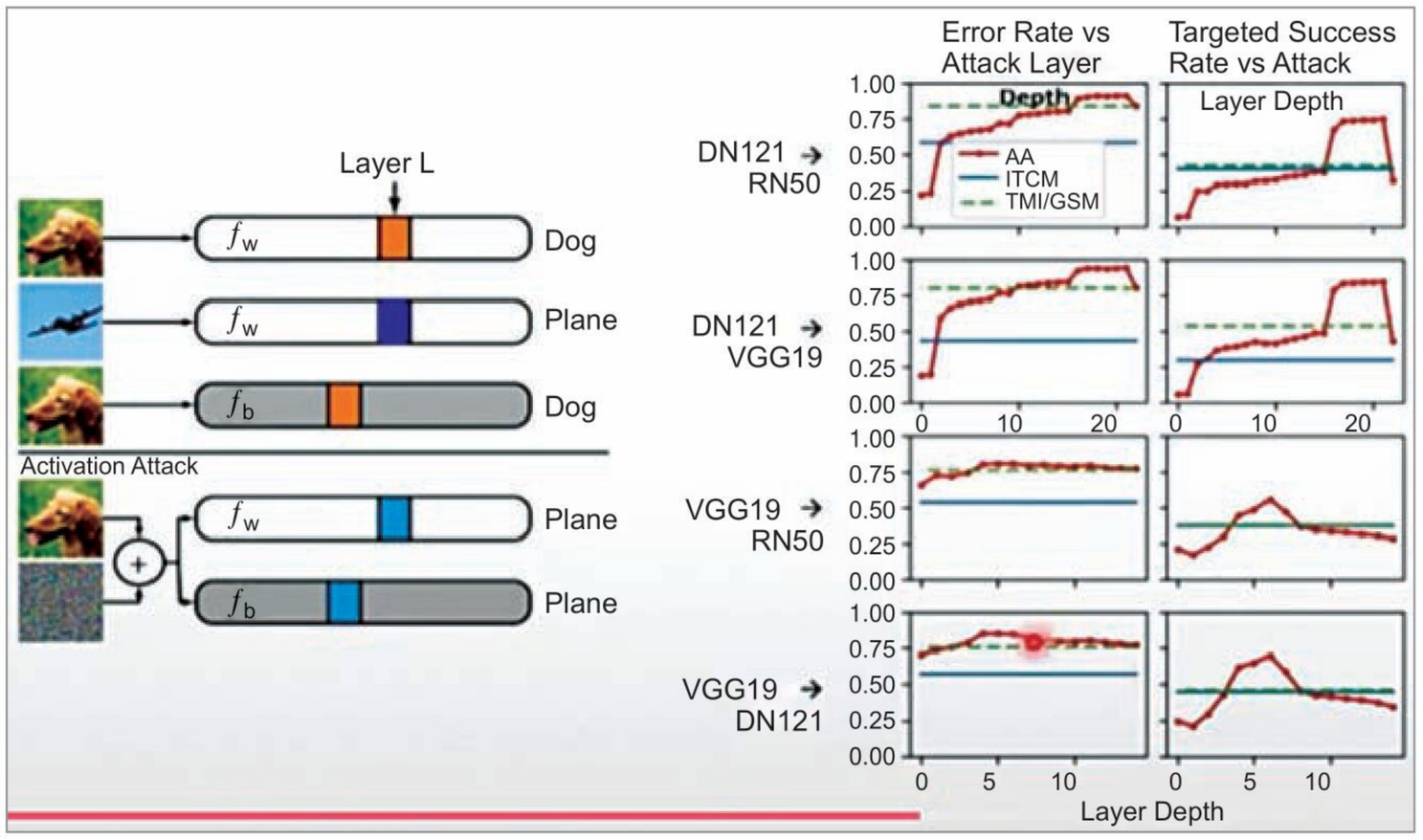


Figure 13: Activation attack

Activation attack: Here we have designed an activation attack by using the properties of adversarial transferability. As shown in Figure 13, the white box model (Fw) and a black box model (Fb) are initially correct. We perturb the source image, which drives the layer L activation of the dog image and perturbs the plane image. The results can be seen on the right hand side.

Adversarial vulnerability isolation

To minimise the attack transferability, we studied why attacks even transfer. Inspired by Eli's paper we found that models entrained on the same data set capture a similar set of non-robust features. These are the features that are highly correlated to the labels yet sensitive to noise. Such features open the door for vulnerabilities and adversary attacks.

We found that a large overlap in the non-robust features leads to overlapping vulnerabilities across different models which, in turn, leads to transferability of the attacks. For every source image and tested set, we randomly sampled another target image and non-robust images that look like the source while having the same hidden features as the target in the hidden layer of the model.

To sum it all up, it is quite understandable that the future computing systems will evolve making them more user-friendly, adaptive, and highly cost-efficient. It is a holistic scheme that integrates the efforts on device, circuit, architecture, system as well as algorithm. To achieve all of this there is a lot of work that needs to be done and a lot of challenges to be met. END

By: Prof. Hai Li

The author is a professor at Duke University.This article is based on the keynote address he gave at the VLSI Design Conference 2022.

Transcribed and curated by Laveesh Kocher

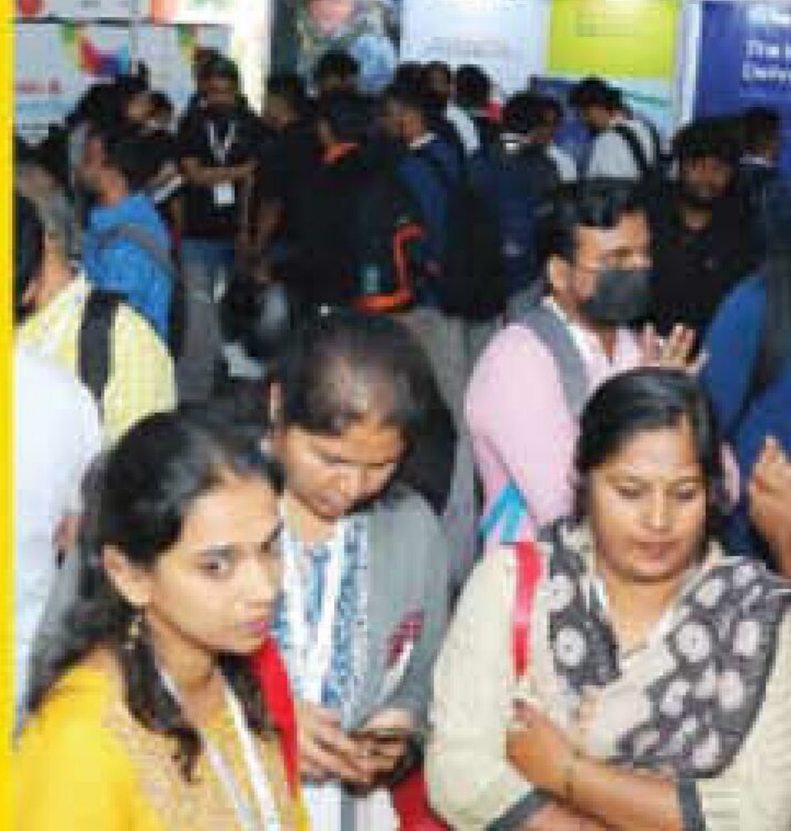
The article was originally published in the September 2022 issue of Electronics For You.

Asia's #1 Open Source Conference

Total registrations
10,340

Total attendees
3732

Speakers
90+



www.OpenSourceIndia.in

An EFYGROUP EVENT

19th Edition

OPEN SOURCE INDIA

Thank You

VISITORS, for making the event a huge success

SPEAKERS, for contributing your valuable thoughts

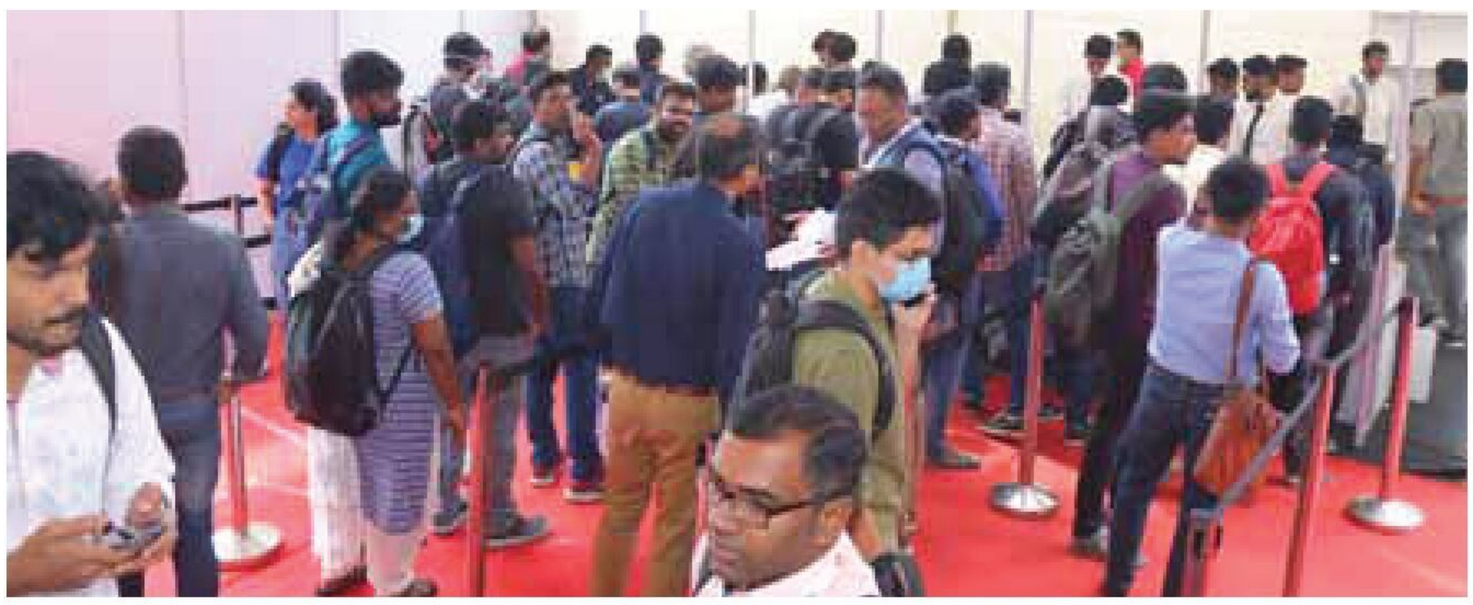
PARTNERS, for your support

Microsoft Azure · Shell · soda foundation · Google Cloud · aws · GitHub · PURESTORAGE · ThriveDX SaaS · harness · DELL Technologies · MOSIP · YML · mt-b · APPLIED SINGULARITY · JAVASCRIPT MEETUP (BANGALORE, INDIA) · OPEN SOURCE BANGALORE Meetup · iotncr · compossible umwelten · APACHE · Apache DevLake · openinventionnetwork · SODA FOUNDATION INDIA MEETUP

For more details, call on +91-98111-55335 or email us at info@opensourceindia.in

Open Source India 2022: WHERE THE INDUSTRY MET THE COMMUNITY

Here's a pictorial overview of all that took place at this mega event for open source enthusiasts, held in Bengaluru in September this year.



Powered By

Platinum Partners

Microsoft Azure | Shell | soda foundation

Gold Partners

Google Cloud | aws

Silver Partner

GitHub

Track Partner – DevOps Track

PURESTORAGE®

Associate Partners

ThriveDX SaaS | harness | DELL Technologies

Exhibiting Partners

composible unveiled | MOSIP | YML

Community Partners

mt-b | APPLIED SINGULARITY | JAVASCRIPT MEETUP (BANGALORE, INDIA) | OPEN SOURCE BANGALORE Meetup | iotner

THE APACHE SOFTWARE FOUNDATION | Apache DevLake™ (incubating) | open invention network | SODA FOUNDATION INDIA MEETUP

Media Partners

electronics FOR YOU | kube CAREERS | kube EVENTS | OpenSource ForYou

electronicsforu.com If it's electronics, it's here | electronicsB2B.com All About Electronics Industry | OpenSource for U | electronicsForYou EXPRESS

September 2022 witnessed a mega open source event – the 19th edition of the annual, much-awaited Open Source India (OSI) which was a physical (on site) event after a gap of three years. The event was held on September 29 and 30, 2022, at NIMHANS Convention Center, Bengaluru, and focused on cutting-edge technologies in open source. It featured packed sessions by over 90 speakers in seven different conference tracks, 11 workshops, and a co-hosted SODA Forum, all of which ran in parallel over two days.

Apart from the talks by experts, the event was also an occasion for exhibitors to showcase their various community-centric offerings to thousands of attendees.

The keynote sessions were delivered by experts from Microsoft, Shell, Google Cloud, AWS and SODA Foundation. The panel discussion 'OSS is not just Code, it's a Community' generated a lot of interest among visitors. The other panel discussions were also attended well.

The hallmark of this year's edition of Open Source India was the presence of community projects that were conceptualised and started in India and then went global.

Open Source India 2022 was attended by a large number of open source supporters from India and around. The audience ranged from CIOs and tech heads to evangelists and academicians. Here is a pictorial round-up for those who missed the event.

1

2

Date: September 29-30, 2022

Venue: NIMHANS Convention Centre, Bengaluru

Total registrations:

10,340

Total attendees:

3,732

Total number of sessions:

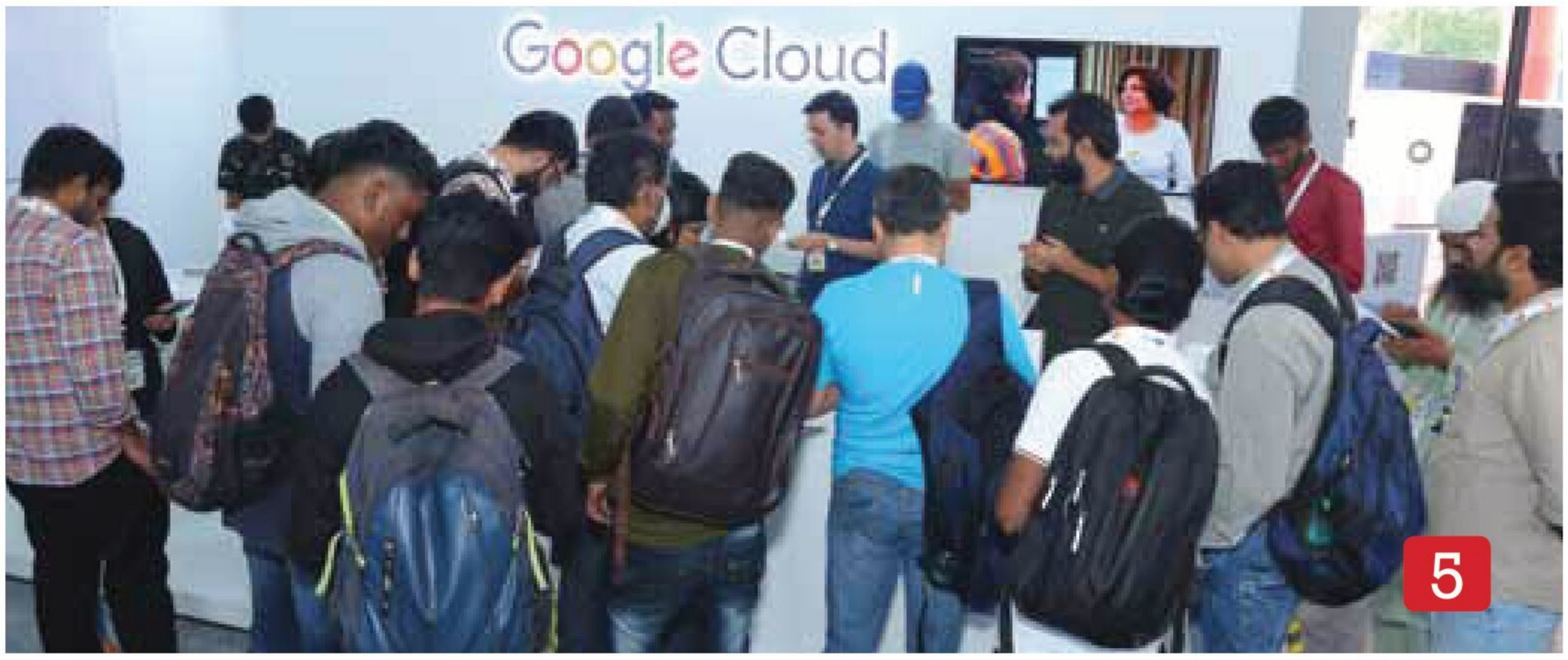
50+

Total number of speakers:

90+

3

4



5

Pictures of

01: Attendees visiting the Microsoft booth

02: Attendees visiting the Shell booth

03: Attendees visiting the SODA booth

04: Attendees at registration counter

05: Attendees at the Google booth


6

7


8


9


10

The keynote sessions

06: Sandeep Alur, Director – Microsoft Technology Center, Microsoft Corporation India, talking about 'Technology Innovations leading to Digital Acceleration'

07: The keynote session on 'Digitalization and Disruptive Innovation in the Energy Industry' by Michiel Stelten-pool, Head of Software Engineering, Shell

08: Suman Debnath, Principal Developer Advocate, Data Engineering and Analytics, Amazon Web Services, presenting his insight on open source at AWS

09: Prashanth Subrahmanyam, Lead Developer Advocate, Google Cloud, talking about open source, open data, and open cloud

10: Keynote session on 'Building Container and Multicloud ODF for Tomorrow!' by Sanil Kumar D., Huawei/SODA Foundation, Chief Architect & Head Data Management India, SODA Foundation TOC CoChair

11

The panel discussions

11: 'OSS is not just Code, it's a Community'. Left to right: Janardan Revuru, JavaScript Evangelist; Vanshika Srivastava, BoxyHQ; Frederick Noronha, Independent Journalist (Chair); Vishal Biyani, InfraCloud; Karan MV, GitHub; Sanil Kumar D., SODA Foundation; Joshua Poddoku, Apache DevLake

12: 'The Business of OSS'. Left to right: Prashant Ghildiyal, CEO & Co-founder, Devtron Inc.; Divyanshu Verma, Co-Founder and CEO, Redinent Innovations; Vinay Yadav, CEO, Webkul Software; Dibya Prakash, CTO & Principal Consultant, Neural Hub (Chair)

13: 'Open Source Careers: Skills, Expertise and Hiring Trends'. Left to right: Srivathsa NS, Senior Engineering Director, Unisys (Chair); Bharat Joshi, Co-founder, VuNet Systems; Sherin Mirza, Competency Center Lead - Full Stack Development, Shell;
Suman Debnath, Principal Developer Advocate, Amazon Web Services;
Deepak B.V., Founder & Principal Consultant, Cadence Connections

14: 'From Container to Multi-cloud: Trends and Opportunities'. Left to right: Dr Keshava Munegowda, Goldman Sachs; Kamesh Sampath, Harness; Veena Jayan, Siemens Healthineers; Sabari Girish S., Cisco; Sanil Kumar D. (Standing), SODA Foundation


12

13

Presenting popular open source projects

15: Manoj Kumar, VP Developer Relations, LambdaTest
16: Prashant Ghildiyal, CEO & Co-founder, Devtron Inc.
17: SasiKumar Ganesan, Vice President of Engineering, MOSIP
18: Vishal Biyani, CTO & Founder, InfraCloud
19: Saurav Pathak, FOSS Evangelist, Co-Founder Bagisto
20: Ankit Nayan, CTO, SigNoz
21: Navendu Pottekkat, Developer Advocate, API7.ai
22: Ruturaj Mohite, Maintainer, CircuitVerse
23: Tejaswini Chile, Product Engineer, Chatwoot
24: Nikhil Bhaskaran, Founder-Director, Brainy Pi

Sessions and workshopsprojects

25: Shubhang Vishwamitra, Senior Director, Y Media Labs

26: A session on 'Enterprise Class Observability' by Rajesh K. Jeyapaul, Engineering Lead, IBM Software Labs

27: Workshop on 'What is New on OWASP Top 10 – 2021' by Lesley Tai, Senior Solution Engineer, ThriveDX

28: Shankar Ramanathan, Site Reliability Engineering, Vonage, speaking about 'Running a Managed Service with K8s'

29: An introductory session on Cloud Native Java, 'java -jar myapp.jar = CI + GitOps' by Kamesh Sampath, Director of Developer Advocacy, Harness

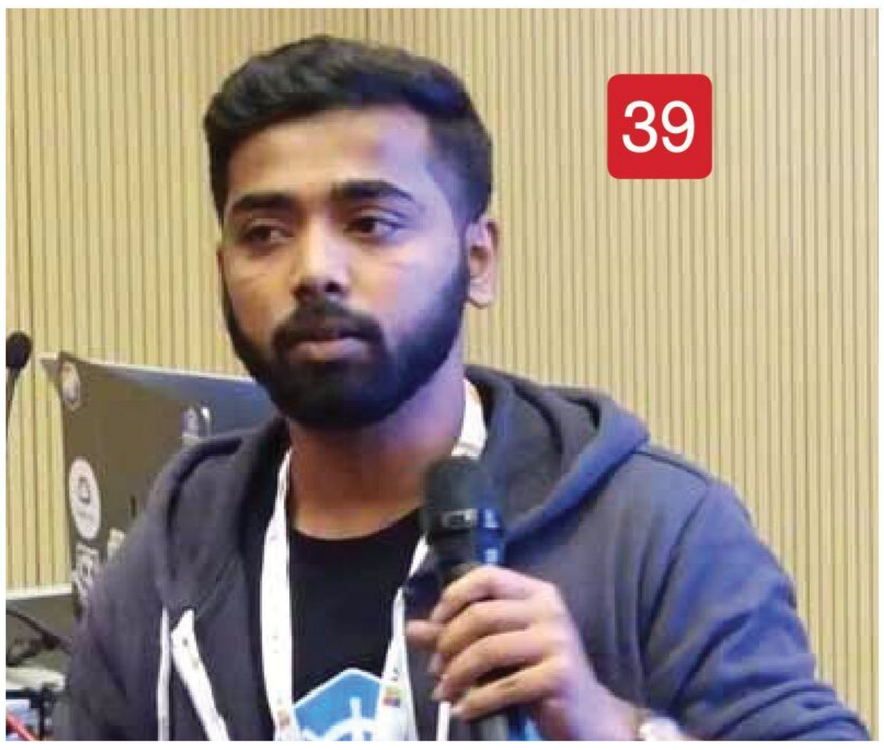
30: Sherin Mirza, Competency Center Lead, Shell, talking about 'Using GitHub Actions to Accelerate DevOps Adoption'

31: Kshithij Iyer, Member of Technical Staff, Pure Storage, discussing 'Enterprise DevOps Strategies for Portworx and K8s'

32: 'DevOps: Weaving the Maze in the Era of Digital Transformation' was the topic that Dr Biswajit Mohapatra, Head of Customer Solutions Management, AWS, spoke on

33: Vivek Sridhar, Senior Developer Advocate, Microsoft, spoke about 'Decoupling Observability from the Application using Dapr!'

34: Workshop on 'Open Source Workloads on Azure Cosmos DB' being conducted by Microsoft



35: A session on 'Adopting MLSecOps' by Dibya Prakash, CTO & Principal Consultant, Neural Hub
36: Binitha MT, Software Principal Engineer, Dell Technologies spoke about 'Time Series Analysis: Anticipating Future with Darts'
37: Subhankar Adak, Senior Software Engineer, Dell EMC
38: Shivay Lamba, Contributor and Meshmate, Layer5
39: Rohit Ghumare, Developer Advocate, Solo.io
40: Karan MV, Senior Manager, International Developer Relations GitHub, speaking on 'Let's build from here, in India'
41: A session on 'Open Source Projects and Legal Issues' by Biju K. Nair, Lawyer, LegaliTech.in
42: Nandakumar Edamana, Software Architect, Epixel Solutions Pvt Ltd, explaining 'How the Lazy Become the Prolific using Code Generators'
43: The SODA Forum team
44: Mohammad Asif Siddiqui, Staff Engineer, VMware, speaking on 'Data Management in the Container and Multicloud Era'

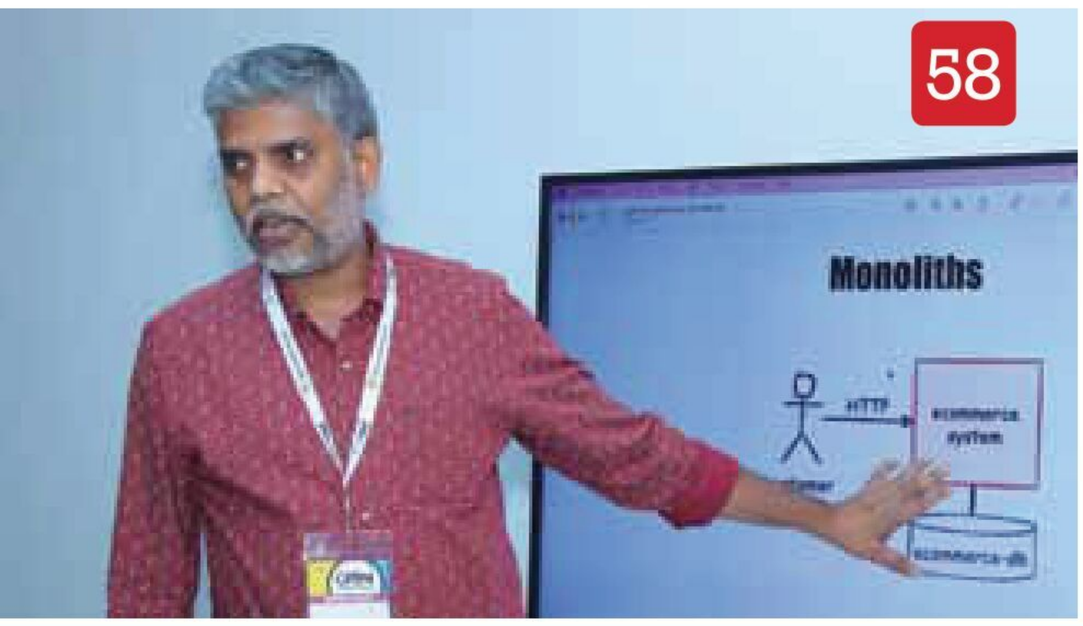
45: A session on 'Data Mesh – An Architecture Paradigm. Is it a Must for all Enterprises?' by Raghvendra Singh Dikhit, Senior Solutions Architect, Impetus Technologies Pvt Ltd

46: Attendees at a workshop by Jeeva S. Chelladhurai, CEO, Comorin Consulting Services

47: Workshop on 'Kubernetes 101: Getting Started' by Neependra Khare, Founder and Principal Consultant, CloudYuga Technologies

48: Workshop on 'Setting up a Production-ready k8s Ecosystem in 2 Hours from Scratch' by the Devtron team

49: Manish Kumar, Senior Software Engineer, Shell (Left), and Rohit Srivastava, Senior Software Engineer, Shell (Right), speaking on 'Green Software@Shell powered by Open Source'



50: A session on 'OCaml 5: Language, Platform and Ecosystem' by Shakthi Kannan, Senior Software Engineer, Tarides India Pvt Ltd

51: Kundan Kumar, Senior Member of Technical Staff, VMware (Left), and Prasad Paple, Operating System Developer, IBM (Right), speaking on 'Linux Synchronization Journey to Making Lockless Software'

52: Closing remarks for SODA Forum by P Sainath Mathur, Practice Head Compute, Storage And Peripherals, ER&D, Wipro Technologies

53: 'Getting Started with Bottlerocket' was the topic for discussion by Mani Chandrasekaran, Lead Architect, Amazon Internet Services Pvt Ltd

54: Vinay Rajagopal, ISV Technology Lead, Red Hat, speaking on 'Multicloud Communication for Kubernetes'

55: Ashay Krishna, Director of Engineering, Microsoft (Left), and Madhava Gaikwad, Senior Software Engineer, Microsoft (Right), introduced attendees to 'SONIC: An Operating System to Manage your Networking Hardware'

56: Sachin Apagundi, Software Principal Engineer, Dell Technologies, speaking on 'Using Ansible and Terraform in System Management'

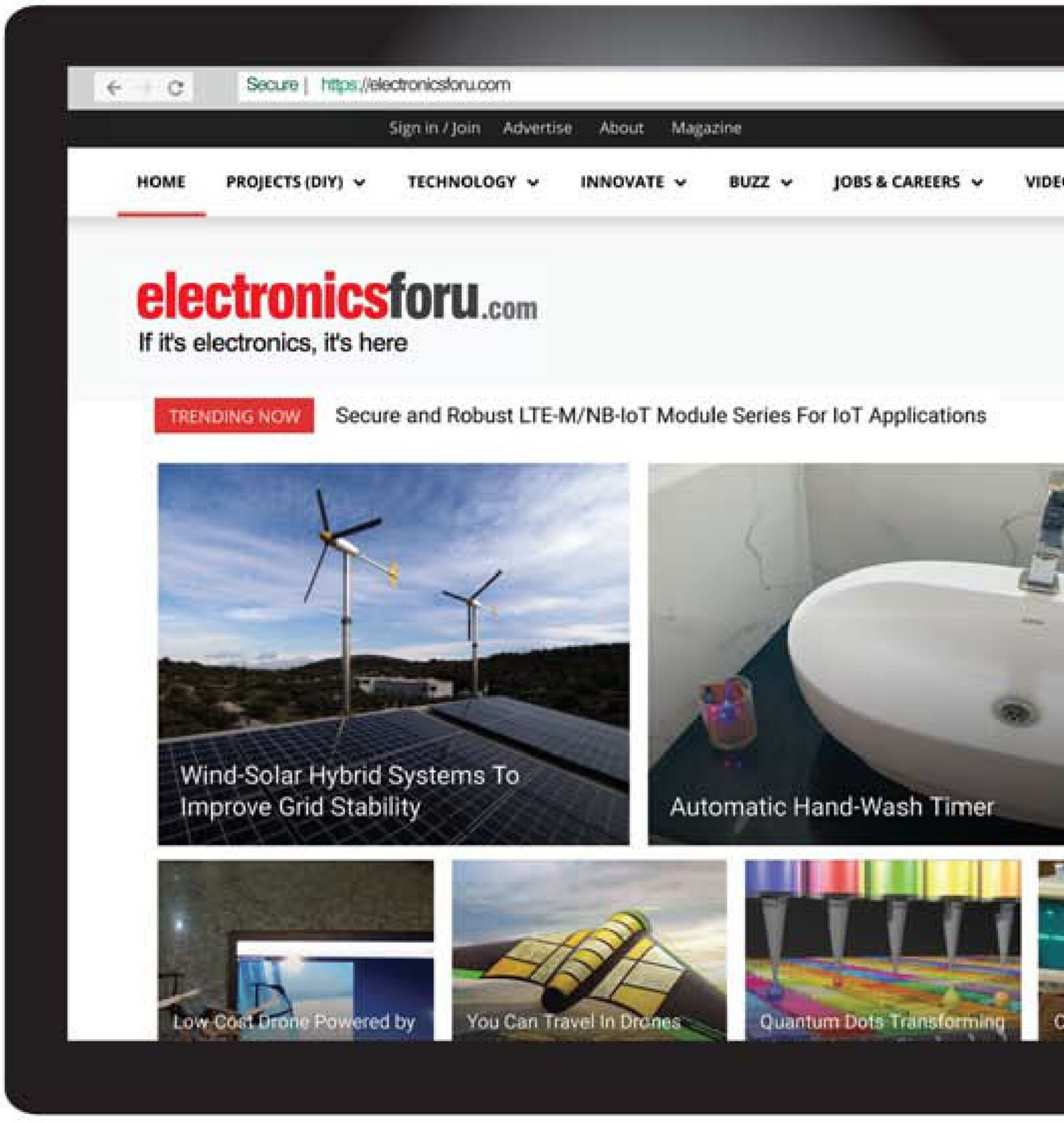
57: A session on 'Centralized Authentication System in a HPC Cluster' by Deepika K., Principal Software Engineer, Dell R&D

58: A workshop on 'Event-Driven Microservices: Challenges and Strategies' by Krishna Mohan Koyya, Proprietor & Principal Consultant, Glarimy Technology Services

59: Balaji Seetharaman, Senior Software Engineer, Extreme Networks

60: Prashant Mishra, Senior Developer Programs Engineer, Google India, speaking on 'Faster SELECT Queries Using Open Source java-bigquery'

Your favourite website has



electronicsforu.com

THANKS TO YOU—OUR ONLINE NETWORK IS

FACTS & FIGURES

- 4 websites (two more coming soon)
- Five major Facebook communities
- Seven major LinkedIN groups & pages
- Million-plus active users (monthly)
- Million-plus reach through Facebook
- Fifty-thousand-plus industry connections through LinkedIN

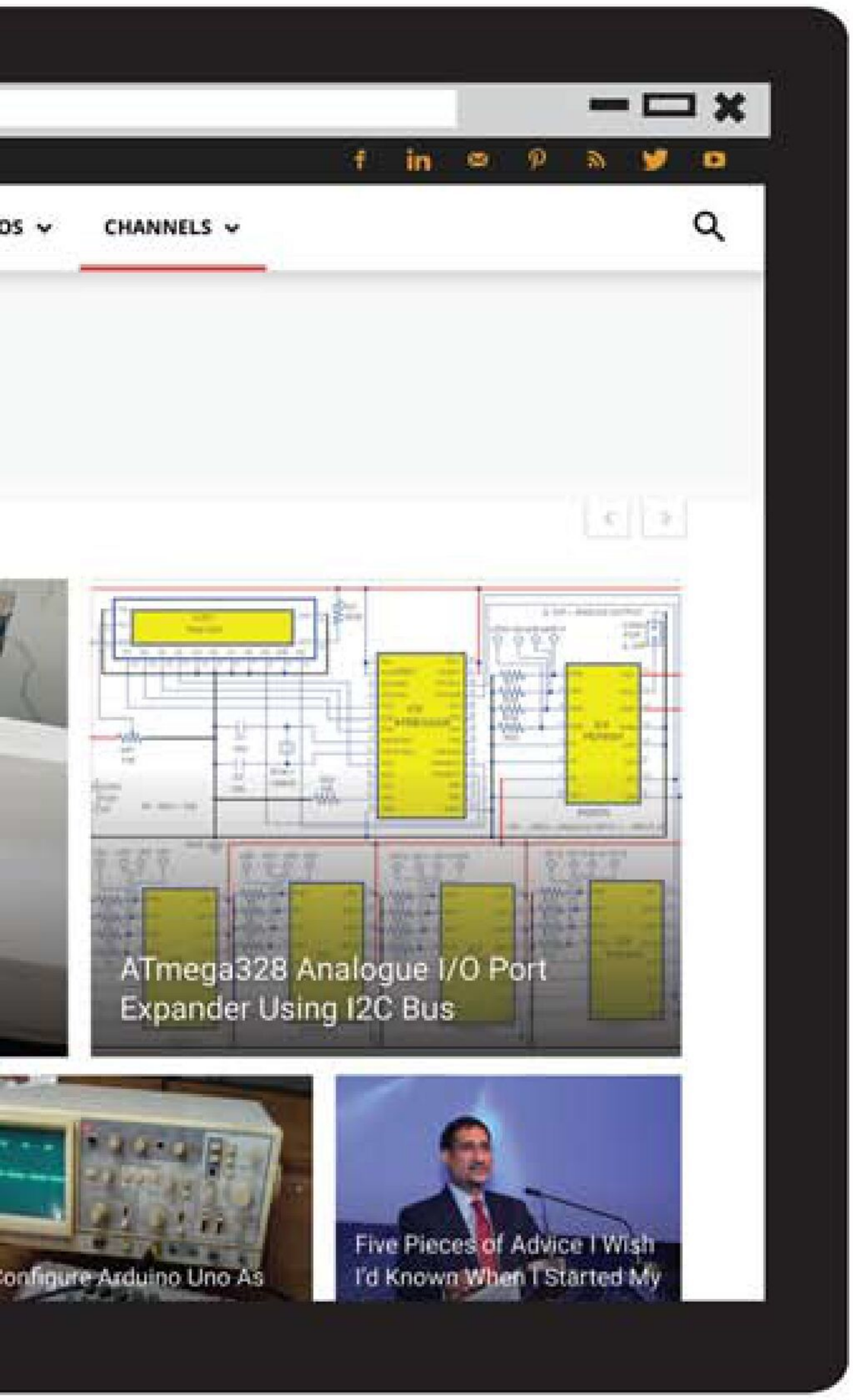
READERS

- You can access all content for FREE
- You can subscribe to newsletters for FREE –on most websites
- Register on our websites to get free invites to technical webinars and seminars

EXPERTS

- Experts who want to share their knowledge through articles, DIY Projects, etc are welcome
- We also welcome experts who want to share their knowledge through webinars or seminars
- You can contact us at editop@efy.in

fast growing peers now...



The Latest in IOT.
A platform for enablers, creators and providers of IOT solutions.

India. Electronics. Directory.
Enabling commerce between buyers & sellers of electronics in India.

Business. Electronics. India.
Everything you want to know about India's electronics industry.

Amazing DIY Projects. Latest Tech trends.

The hang-out for electronics enthusiasts.

AMONGST THE WORLD'S TOP 5 AND GROWING!

INDUSTRY

- You can advertise for as little as US$ 100 per month
- Special combo offers for advertisers in our print publications
- We've now enabled flexible CPM-based advertising
- You can advertise on the platform of your choice (based on your target audience)
- We invite press releases at efy-edit-team@efy.in
- Press releases are published free of cost, subject to discretion of the editorial team

RESPONSE GUARANTEED SOLUTIONS

We now also act as marketing partners for our clients and drive entire marketing for them, where we charge them on basis of results and not efforts!

CONTACT US: Shrikant Rao • growmybiz@efy.in • +91-98111 55335

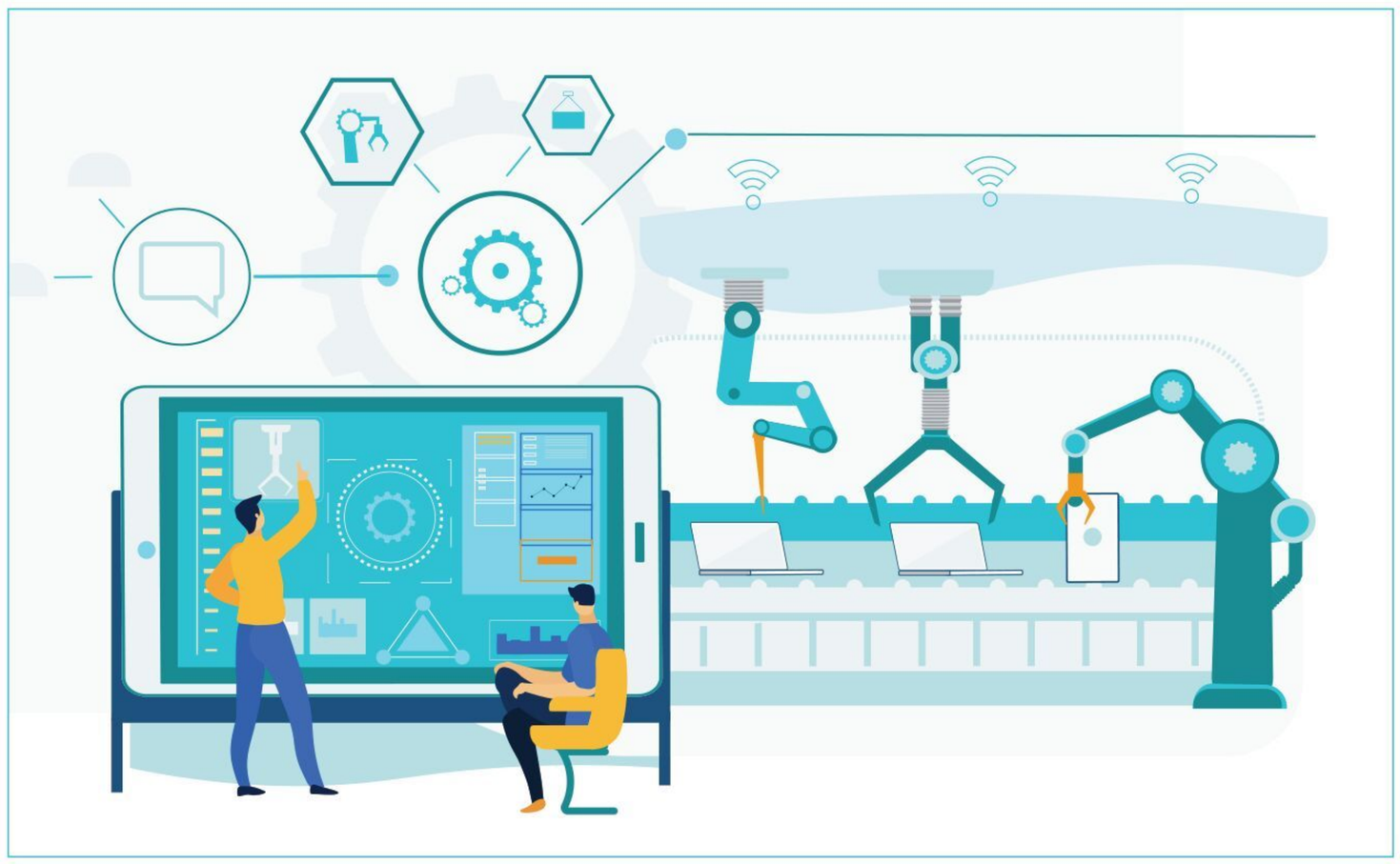
FOCUS

Internet Of Thing

Industry 4.0 and Embedded Internet of Things: An Introduction

The fourth industrial revolution, commonly referred to as Industry 4.0, is all about making businesses smarter. It is focused on deepening the impact of digital technologies by making machines more autonomous and enabling them to communicate with one another. These machines can now manage enormous amounts of data in ways that humans simply cannot handle. This article introduces you to Embedded Internet of Things (EIoT) and Industry 4.0. *Read more on page.....39*

How IoT can make the Workplace Intelligent



It's well-known that an intelligent workplace enhances employee productivity. The Internet of Things plays a big role in making a workplace smart and intelligent. Let's see how. *Read more on page.....45*

IoT Analytics: Making Sense of Big Data

This article covers the characteristics of IoT analytics and the challenges faced when adopting it in the industry. It also presents real world use cases of IoT analytics.

Read more on page.....48

A Complete Guide to Getting Started with Arduino

Building DIY projects with Arduino is super easy and a lot of fun. However, it may not always be as straightforward for beginners with absolutely no background knowledge in hardware or programming. In this article, we cover everything you need to know for getting started with your first Arduino project. That includes an overview of the Arduino board, the IDE, the programming language and a simple project to wet your beak.

Read more on page.....53

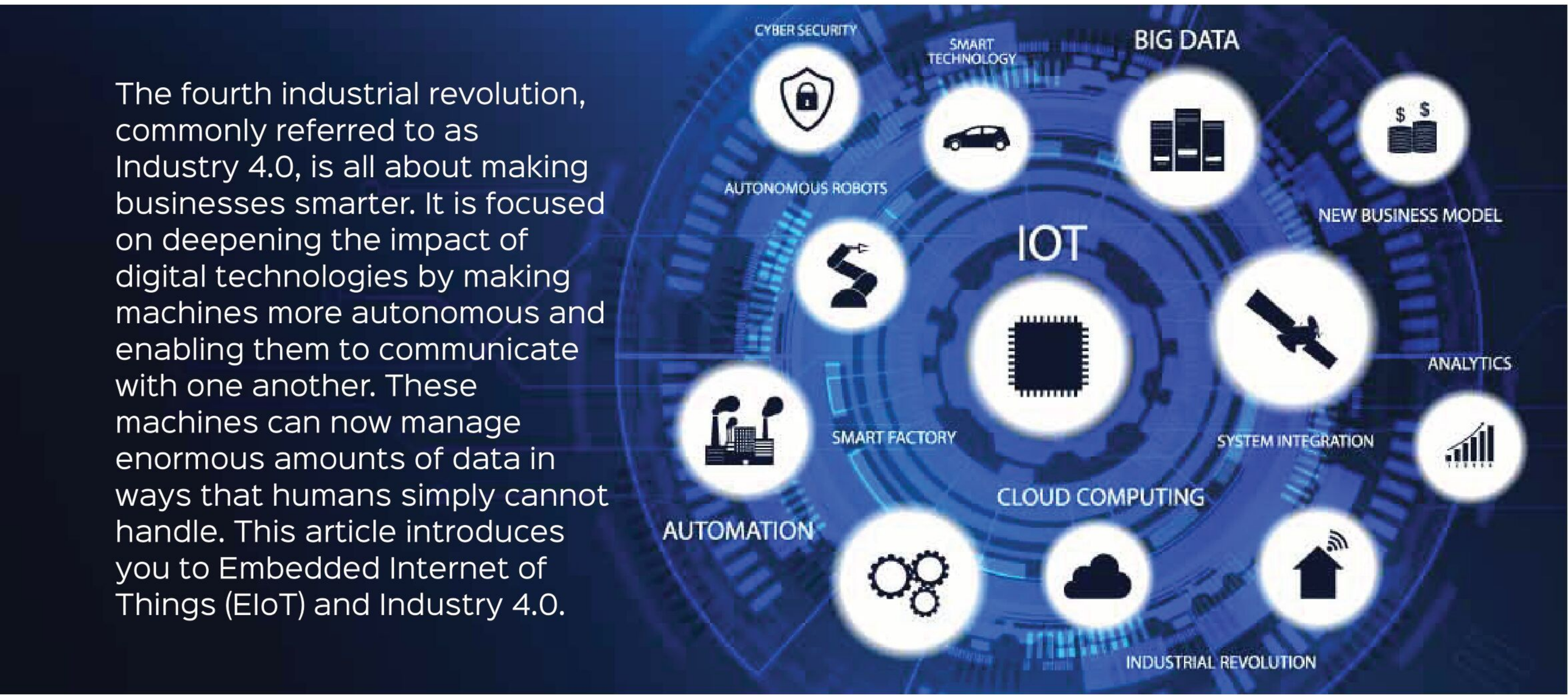
Connecting Internet of Things Devices on Thinger.io

This article is about connecting various IoT devices on Thinger.io, an IoT platform. This step-by-step tutorial will guide you on how to connect a development board to the platform and start sending data from the device.

Read more on page.....56

Industry 4.0 and Embedded Internet of Things: An Introduction

The fourth industrial revolution, commonly referred to as Industry 4.0, is all about making businesses smarter. It is focused on deepening the impact of digital technologies by making machines more autonomous and enabling them to communicate with one another. These machines can now manage enormous amounts of data in ways that humans simply cannot handle. This article introduces you to Embedded Internet of Things (EIoT) and Industry 4.0.



The third industrial revolution was focused on converting mechanical and analogue processes to digital ones to be ready for advanced automation. Similar to the transition from steam to electricity in the second industrial revolution, the advent of Industry 4.0 technology marks a major transformation in how companies conduct their operations.

The effects of Industry 4.0 on production are extensive. It is being used to increase operational effectiveness, improve demand forecasting, eliminate data silos, engage in predictive maintenance, provide virtual training to employees, and more. Industry 4.0 encompasses manufacturing from planning to delivery as part of a larger concept known as digital transformation. It includes solutions for deep analytics, shop floor data sensors, smart warehouses, simulated modifications, as well as product and asset tracking.

Industry 4.0 technologies assist manufacturers in bridging the gap between previously distinct processes and a more transparent, viewable perspective across the entire business with a wealth of useful insights. The emergence of digital solutions and cutting-edge technology, which are frequently linked to Industry 4.0, have made this convergence possible.

Some of the key technologies associated with Industry 4.0 are:

- Industrial Internet of Things
- Smart wearable gadgets
- Smart cities and advanced monitoring
- Advanced robotics
- Digital twins
- Big Data analytics
- E-governance
- Additive manufacturing (AM)
- Cloud computing
- Augmented reality (AR) and virtual reality (VR)
- Quantum computing based systems
- Simulation platforms and suites

By integrating formerly disjointed systems and processes with

interconnected computer systems across the value and supply chain, these technologies are assisting in the digital transformation of manufacturing.

Adopting Industry 4.0, digital manufacturing, and the associated interconnectivity enables businesses to reap a variety of benefits, including improved operational performance and increased agility.

Embedded Internet of Things (EIoT) and Industry 4.0

Embedded Internet of Things (EIoT) refers to the integration of embedded devices (including sensors and open source hardware) with IoT based platforms and communication channels to enable real-time communication for varied applications. EIoT with Industry 4.0 is a technology shift of traditional engineering and governance to smart deployments. The three technical themes (connectivity, intelligence, and flexible automation) driving this transformation are collectively referred to as Industry 4.0, or the fourth industrial revolution with embedded devices.

The fourth industrial revolution represents breakthroughs in technologies, such as robots, intelligent systems, nanotechnology, quantum computing, microbiology, the Internet of Things, 3D printing, and autonomous cars.

In order to build a cyber-physical environment, Industry 4.0 combines OT (operational technology) and IT. A number of software platforms and libraries can be used for the simulation of EIoT and Industry 4.0 applications (Table 1).

Some use cases

In Industry 4.0 and EIoT simulations, there is a need to integrate lightweight protocols so that low-powered and smart gadgets can communicate effectively. MQTT is a key protocol that is used in Industry 4.0 and IoT based deployments.

Figure 1: Key segments of Industry 4.0

Table 1: Platforms and software suites for simulation of Industry 4.0 applications

Platform/Software suite	URL
Cloud MQTT	*cloudmqtt.com*
Mosquitto MQTT Broker	*mosquitto.org*
FlowHub	*flowhub.io*
Sumo	*eclipse.org/sumo*
NoFloJS	*noflowjs.org*
PraxisLive	*praxislive.org*
Netron	*github.com/lutzroeder/netron*
Ryven	*gentle-sand-0114c7110.azurestaticapps.net/*
PyFlow	*wonderworks-software.github.io/PyFlow/*
Wyliodrin	*wyliodrin.com*
Flutter Wireless	*flutterwireless.com*
SiteWhere	*sitewhere.io*
DSA	*iot-dsa.org*
Zetta	*zettajs.org*
Node Red	*nodered.org*
IoTivity	*iotivity.org*
Open IoT	*openiot.eu*

There are primarily two types of objects in IoT communication — publisher and subscriber. Publishing refers to the transmission of data signals — the process of moving data from a computer to an external device. The receiver system is referred to as the subscriber in IoT communication.

Publisher and subscriber are terms used to describe the source and recipient of a data signal, respectively. The MQTT broker, also known as an IoT server, serves as the focal point and regulates communication between publishers and subscribers. It manages the data that publishers broadcast for the other devices known as subscribers. For instance, information may be sent from a temperature sensor to a farmer's mobile phone. In this instance, the temperature sensor (publisher) is the source of the data signals, and the farmer's gadget is the receiver (subscriber).

These MQTT broker applications can be installed on a variety of specialised hardware, including Raspberry Pi, Arduino, laptops, servers, and many others.

Cloud based MQTT

URL: cloudmqtt.com

CloudMQTT, one of the most widely used MQTT brokers, can be used for device communication and implementation of Industry 4.0 based scenarios. It can be used to implement IoT scenarios for a variety of applications including smart toll plaza, smart cities, smart homes, and smart offices.

For researchers and academicians, CloudMQTT offers a free package called 'Cute Cat' that allows five users or connections for IoT devices. Using an active Google account, free instances can be built on the CloudMQTT network.

The features integrated in Cloud MQTT include:

- Open source message broker that implements the MQTT protocol
- 24x7 availability of MQTT broker
- Cloud compatibility
- Lightweight and high performance; integrates with low power single boards like Raspberry Pi, Arduino, and ESP8266 as well as computers and servers
- Integration of lightweight methods for messaging using a publish and subscribe message queueing model
- Fully managed Mosquitto servers in the cloud

To establish a real-time connection between CloudMQTT and a device, IoT hardware or smartphones can use authentication information from CloudMQTT.

Traffic simulation using Eclipse SUMO

URL: eclipse.org/sumo

Smart vehicular communication in smart cities and e-governance are key application areas of Industry 4.0. Eclipse SUMO (Simulation of Urban MObility) is very effective in simulating a traffic scenario. It integrates a range of features to simulate smart vehicular environment, and can work with multiple vehicles of different types in different environmental conditions.

SUMO is free and open source. Since its introduction in 2001, it has enabled the modelling of multi-mode traffic networks, which include pedestrians, public transportation, and private vehicles. Numerous auxiliary tools, such as network import, route calculations, visualisation, and emission

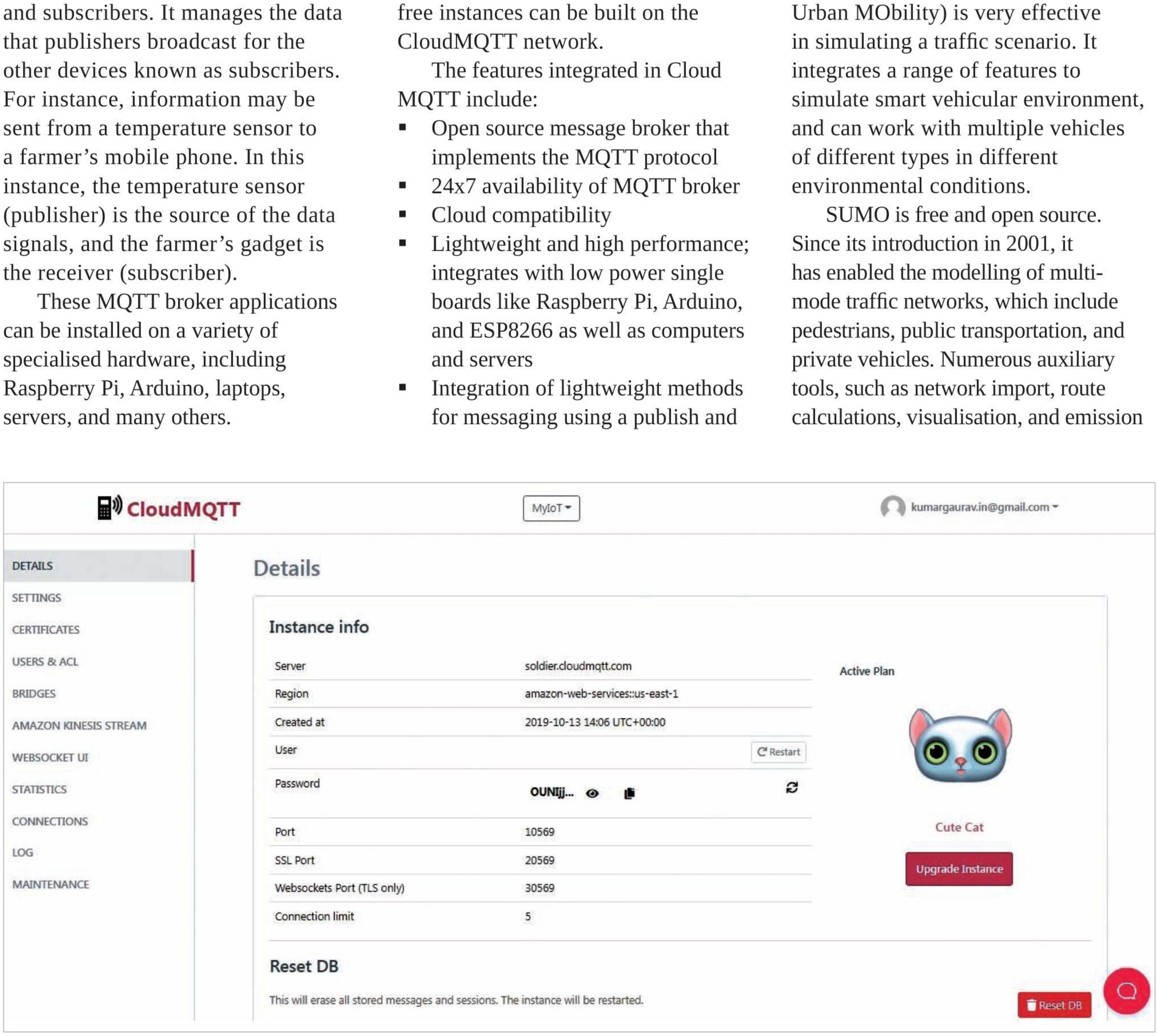


Figure 2: Cloud based MQTT for simulating an IoT scenario

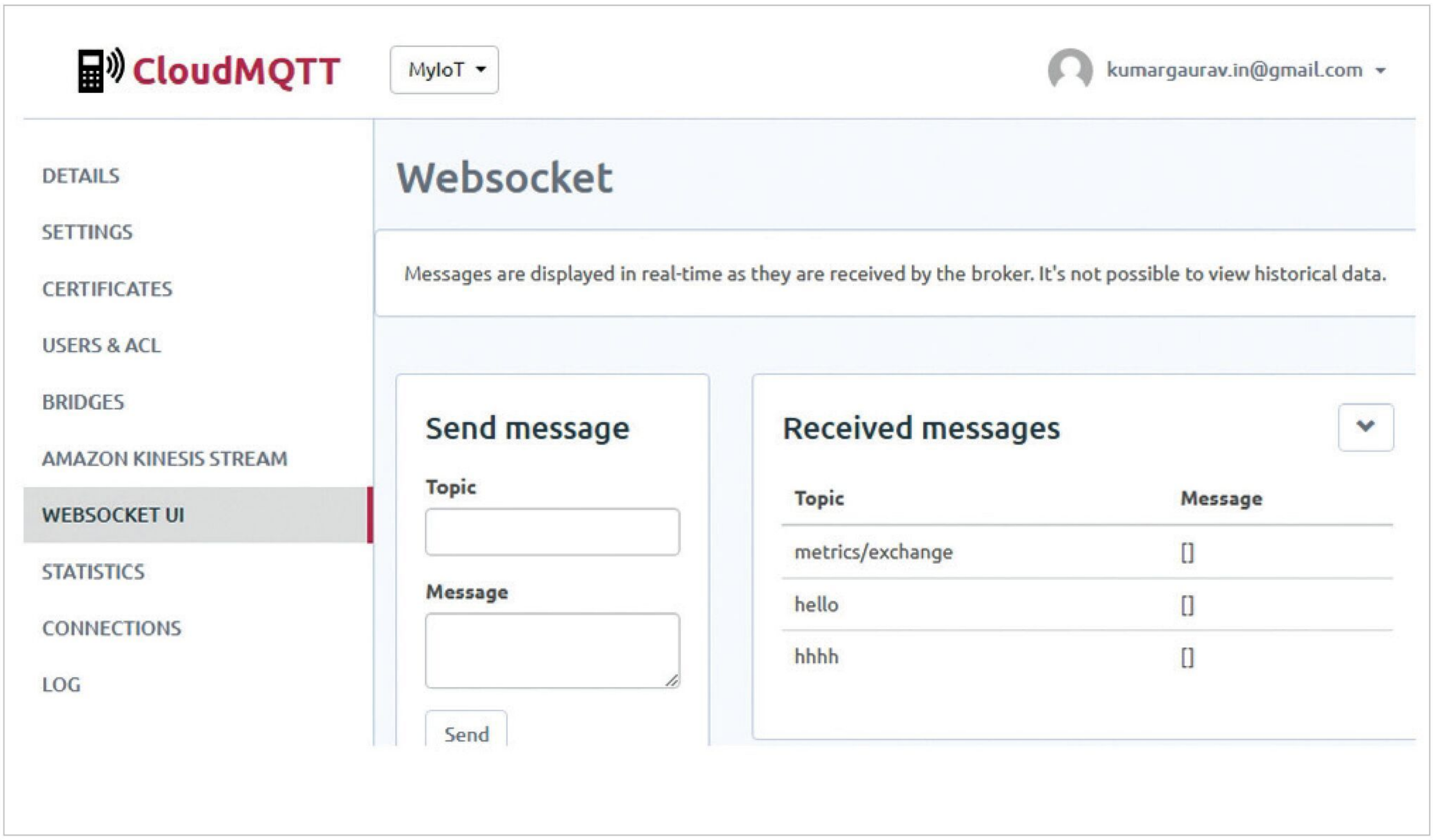


Figure 3: Interfacing with CloudMQTT and messaging panel

Figure 4: SUMO simulator for smart traffic scenarios

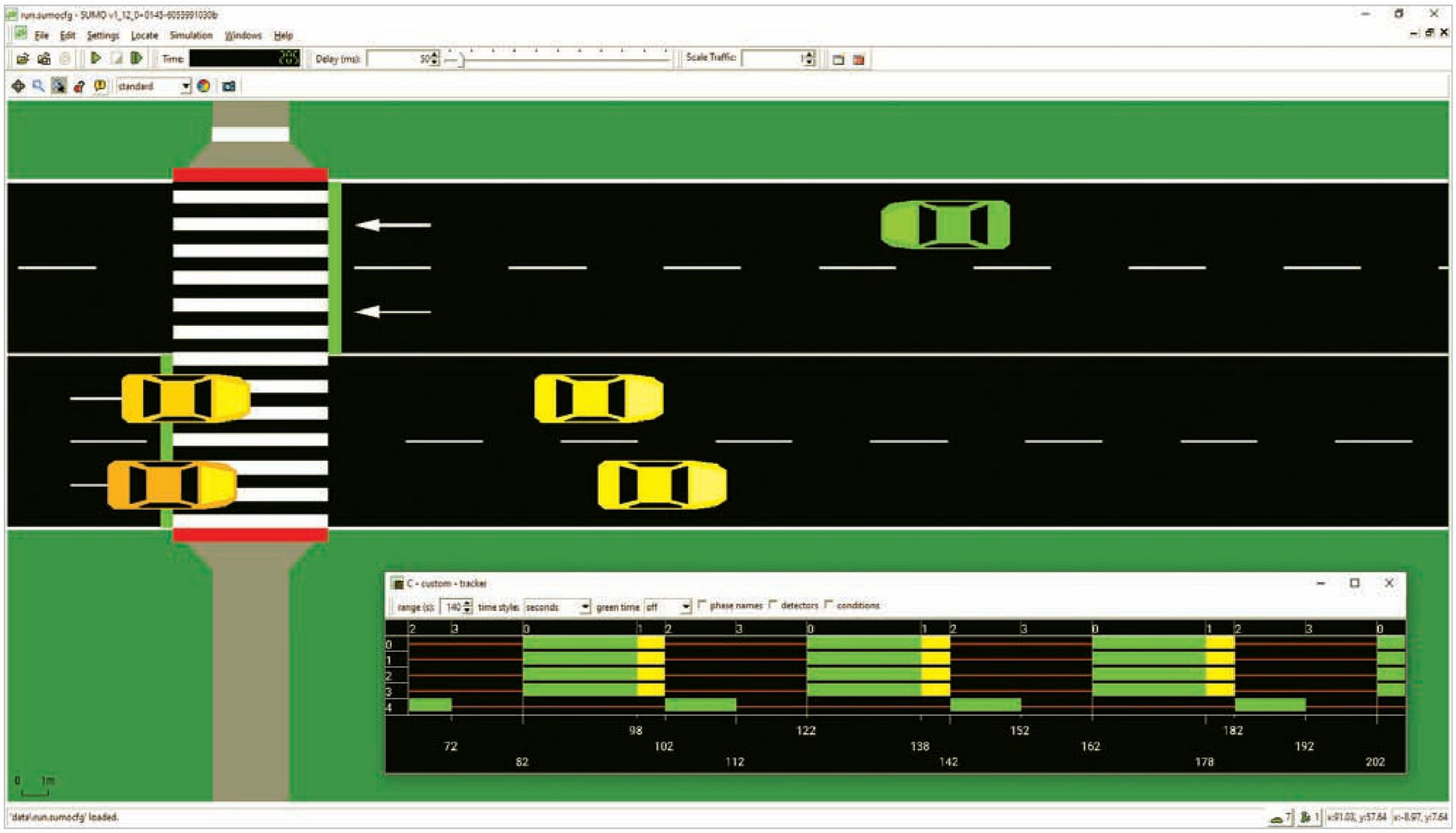


Figure 5: Smart vehicular simulation in SUMO

calculation, are included with SUMO and automate fundamental processes for the development, execution, and evaluation of traffic simulations. Custom models can be added to SUMO, and it offers a number of APIs for controlling the simulation from a distance.

The key features of SUMO are:

- Free and open source
- Automated driving
- Vehicle communication
- Traffic management
- Multimodal traffic
- Microscopic simulation
- Online interactions
- Traffic lights
- Network import
- Demand generation
- Performance
- Portability

The manufacturing sector is experiencing exciting times right now as new digital technologies emerge. Industry 4.0 is also making it easier for humans and robots to collaborate, enabling firms to gain more knowledge, lower the possibility of error, and make wiser decisions. The only way for manufacturers to stay ahead of the curve in this new digital era is by embracing Industry 4.0.

Researchers and practitioners of Industry 4.0 can use free and open source platforms to simulate and experiment with its real world applications. Such simulation sessions will help analyse the outcome in real-time before actual deployment. END

Dr Gaurav Kumar

The author is associated with various academic and research institutes for delivering expert lectures and conducting technical workshops on the latest technologies and tools.

How IoT can make the Workplace Intelligent

It's well-known that an intelligent workplace enhances employee productivity. The Internet of Things plays a big role in making a workplace smart and intelligent. Let's see how.

Enterprises demand a flexible, intelligent, and connected workplace as it enables its occupants to work smarter. It also helps in enhancing employee productivity and experience, leading to better employee retention and helping attract a new talent pool. Internet of Things (IoT) enables an intelligent office to utilise physical devices for collecting and sharing data. It helps enterprises to design flexible, activity-centric workspaces to better meet employees' evolving needs by analysing IoT sensors and connected system data on how space is being utilised. It enriches employee journey maps (such as seamless access, automatic room and desk booking, and colleague or way finding) from the moment they enter the office, and creates a conducive work environment by using smart temperature and light controls.

IoT for an intelligent workplace

An intelligent workplace uses automated processes to control an office's operation including heating, ventilation, air conditioning, lighting, security and other systems. A smart office uses sensors, actuators and microchips to collect data and manage it according to business functions and services. This infrastructure helps real estate and facility managers improve asset reliability and performance, which reduces energy use, optimises how space is used and minimises the environmental impact of buildings.

Heating, ventilation, and air conditioning (HVAC): Enterprises can digitally integrate an office HVAC system with sensors and software. They can provide mobile applications and smart technologies that allow employees to customise the temperature and lighting in their workspaces. This helps in:

- Air quality and temperature management based on occupant needs
- Automatic switch off of air conditioning and heating to avoid unnecessary energy use
- Control of humidity levels

Networked lighting: Digital integration of lighting fixtures with sensors and software can create an environment that maximises occupants' comfort and saves energy. Such lighting helps to:

- Adjust light to occupancy levels
- Control lighting as per the personal choice of occupants
- Save energy
- Adjust light based on ambient conditions, and control window shades

Occupancy sensors: These identify the utilisation of a workplace position through movement and/or thermal measurement. A sensor can be positioned at any location occupied by a person. It helps in monitoring:

- Office utilisation
- Desk utilisation
- Meeting room utilisation
- Heat maps to illustrate occupancy of office
- People counter – counts real-time occupancy
- Room sensor – detects if a room is being used or not

Access and security: This involves use of wearables (smartwatch, smart badge, etc) to authenticate employees for access to buildings, meeting rooms, equipment, devices and data. Smart wristbands could become ID badge replacements (or smart ID badges could be created) in the office space. For example, the building will recognise your car when you arrive, authenticate it, and then direct you to a parking

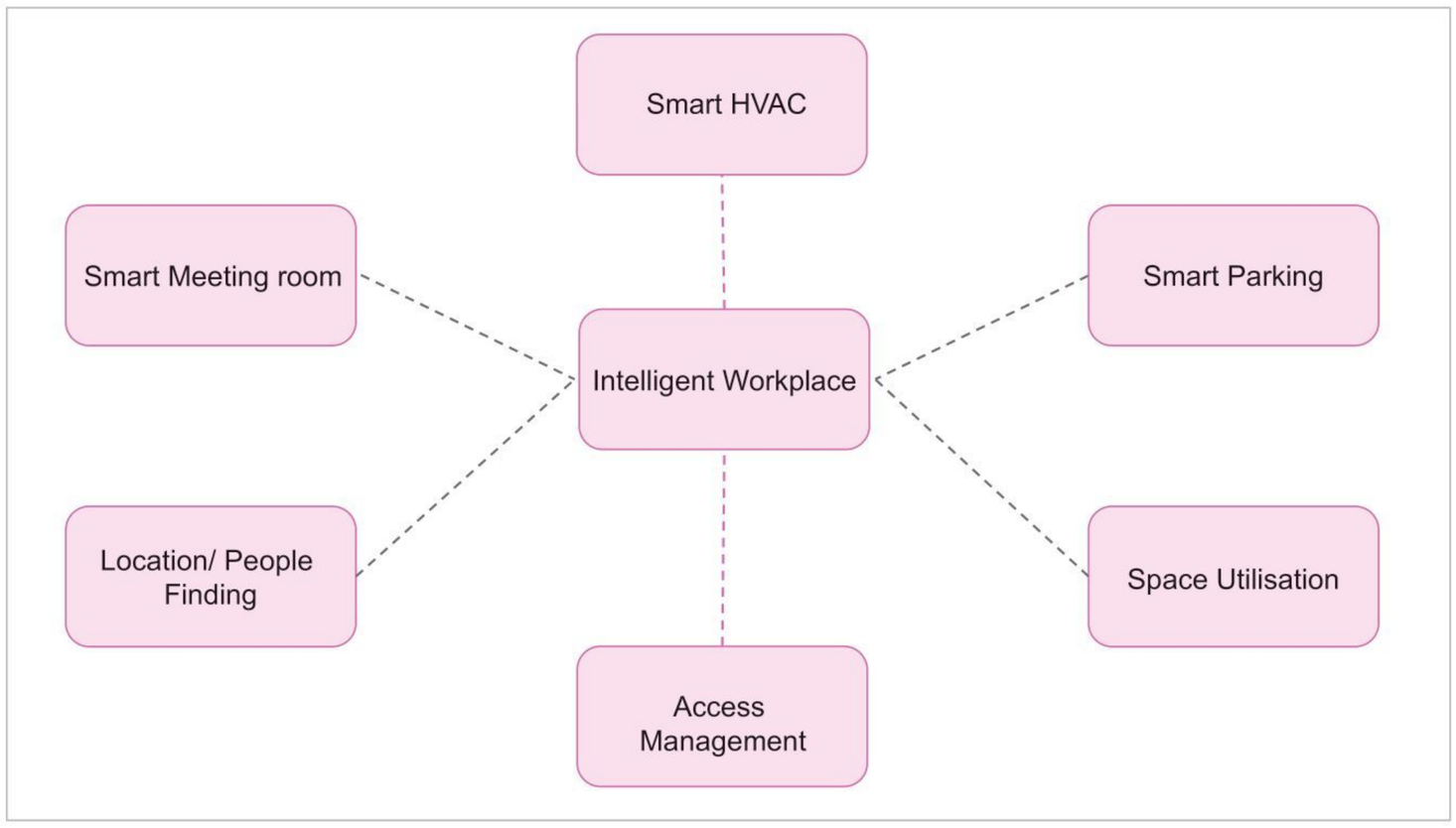


Figure 1: IoT use cases for the intelligent workspace

spot, thus providing a seamless user experience. Networked cameras can stream live video straight to central control, computer and mobile devices. They help to:

- Monitor remote locations anywhere
- Give office access to staff remotely in real-time
- Monitor real-time occupancy data

Meeting room/workplace reservation: This enables occupants to simplify workplace meeting room reservations. Instead of their wasting time looking for places to work and conduct meetings, a smart workspace can proactively carry out such routine work. Employees can use a digital workspace mobile app to book rooms and seats. Integration with employees' calendars enables the system to send smart prompts to devices. This helps to:

- Book a room anywhere and in real time
- Sense when a room is occupied
- Know when a room is available
- Eliminate ghost meetings
- Find the right room to fit the needs of every meeting

With sensor technology, an individual doesn't have to manually sign in or out with their card, thus providing a seamless experience. Sensors will recognise when that person enters or exits a room, which can assist in knowing when meeting rooms are in use. In case people don't show up for a meeting or forget to cancel it (which is common with recurring meetings), these sensors can automatically set the meeting rooms free to be booked by other employees.

Wellness: Implementing wearable programs to increase employee health, wellness and productivity helps to:

- Promote healthy habits
- Gamify wellness for employees (making participation a fun and engaging 'game')
- Give employees benefits for participation

Eighty per cent of employees using healthy wellness programs report being more engaged at work.

Wayfinding and people finding: Knowing the location of a person or thing unlocks many contextual digital workplace opportunities; examples include asset tracking, improving worker safety, understanding employee behaviour, predicting needs, and providing new services in the smart office.

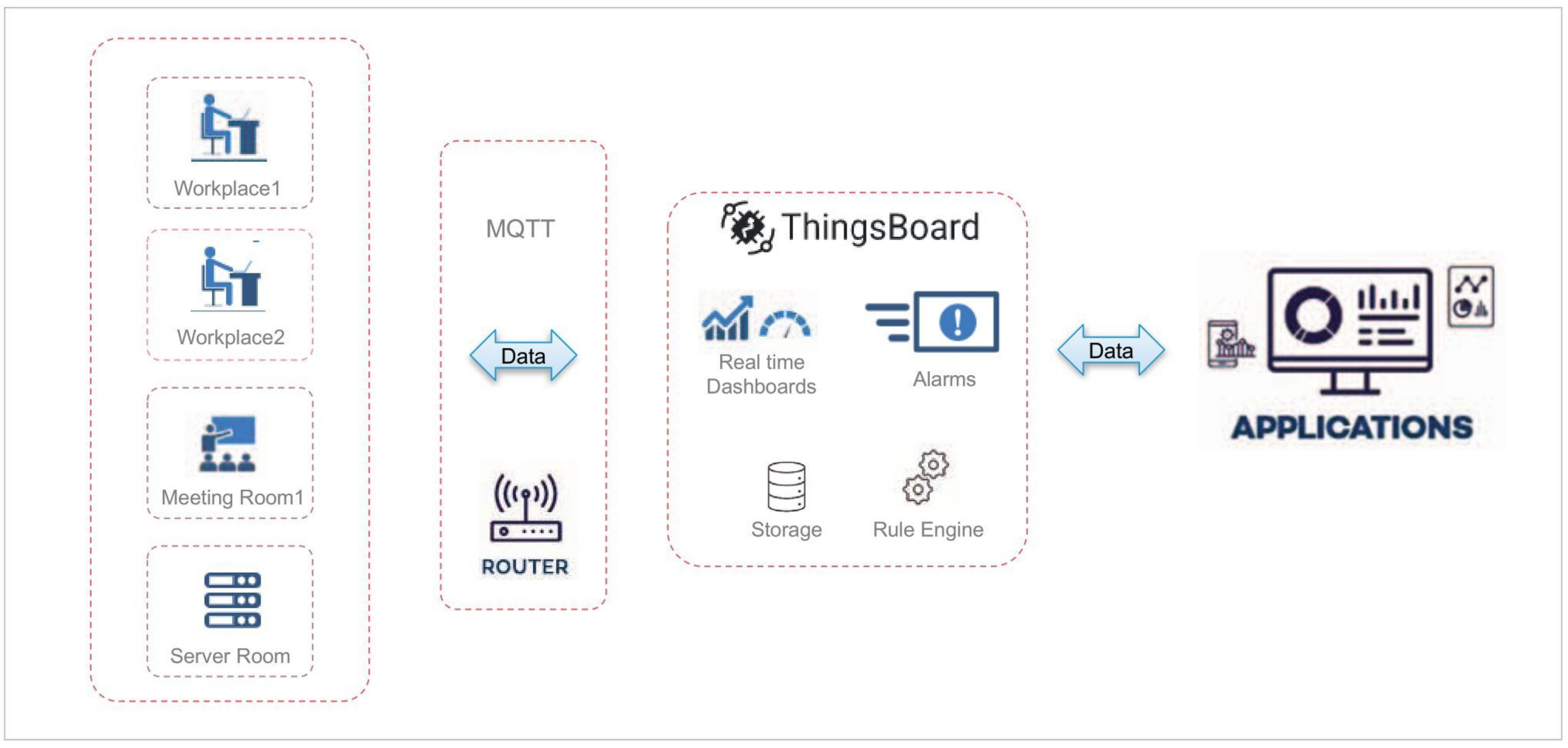


Figure 2: ThingsBoard smart office solution

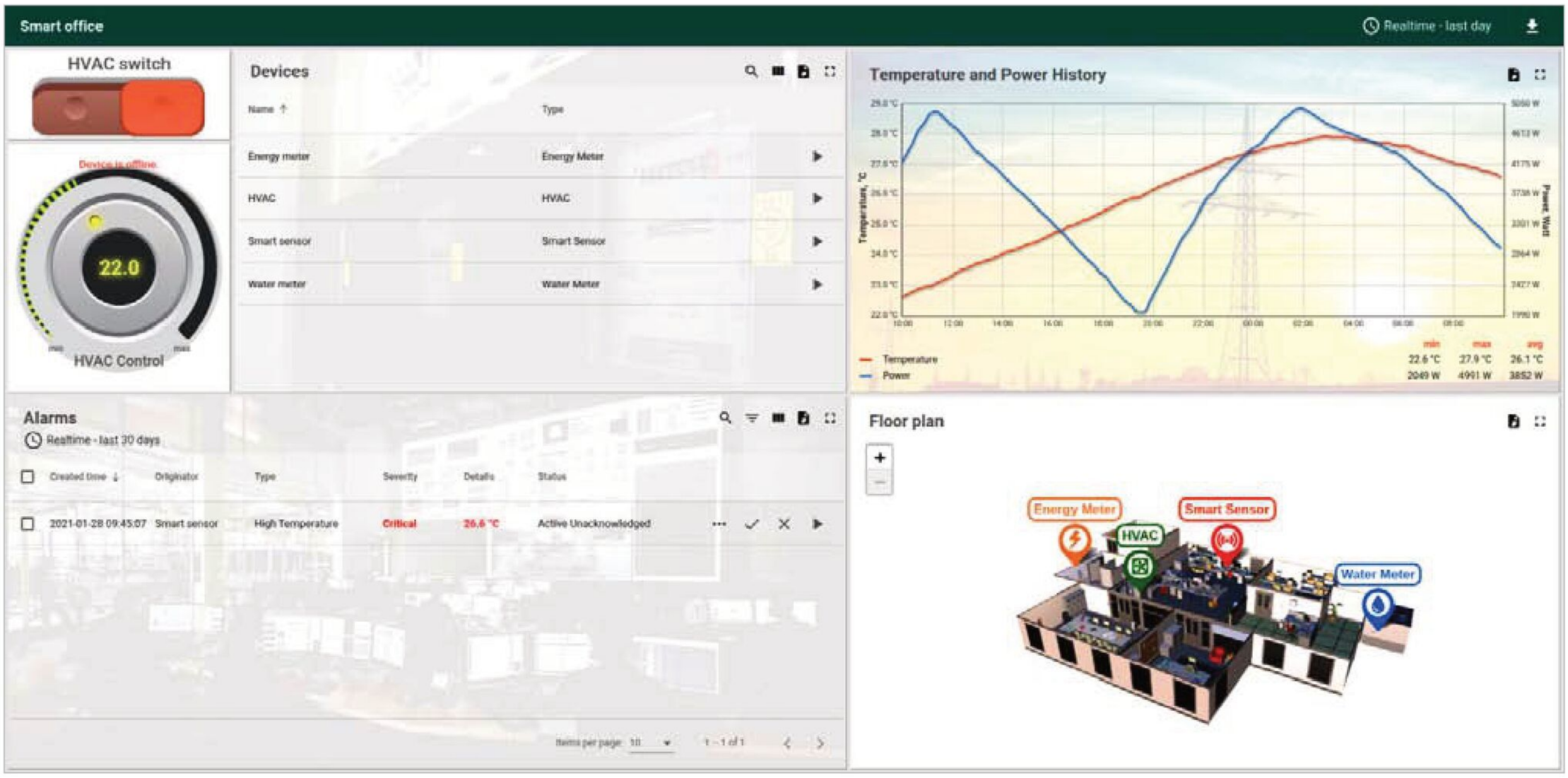


Figure 3: Smart office dashboard

Open source intelligent office solutions

There are many open source platforms available that can be used to build intelligent office solutions. Mainflux, ThingsBoard, DeviceHive, Thinger, Zetta, OpenRemote, Kaa, Fiware, ThingsConnect, WS02Rap are popular open source IoT platforms.

ThingsBoard: This platform dramatically reduces the effort needed to create smart office solutions. It is widely used for:

- Smart energy management.
- Access logging and control
- Indoor tracking of employees
- Occupancy of meeting and wash rooms
- Indoor air quality and HVAC monitoring

By leveraging this IoT platform, we can save up to 90 per cent of the development time needed to build smart office solutions. The benefits are:

- Reliable and fault-tolerant data collection from IoT devices and sensors
- Powerful rule engine to process collected data for valuable insights like setting of alarms
- Advanced and flexible visualisation of real-time and historical data
- Customisable end user dashboards to share the monitoring results
- On-premises and cloud deployment options
- Remote control and OTA (over the air) updates for IoT devices
- Customisable mobile applications with minimum coding efforts

ThingsBoard provides production-ready server infrastructure to connect IoT devices, as well as store, analyse and share the collected data.

The ThingsBoard smart office template represents a basic workspace monitoring and management solution. With this layout (Figure 2), we get an interactive dashboard with the ability to control the HVAC system, and have a high-level report of the key metrics necessary for effective and proactive office management. As part of this solution, there are smart office dashboards that display data from multiple devices. These dashboards may be used to:

- Observe office sensors and their location
- Browse indoor temperature and power consumption history
- Monitor temperature alarms
- Control HVAC (requires connected device)
- Observe specific details for each sensor

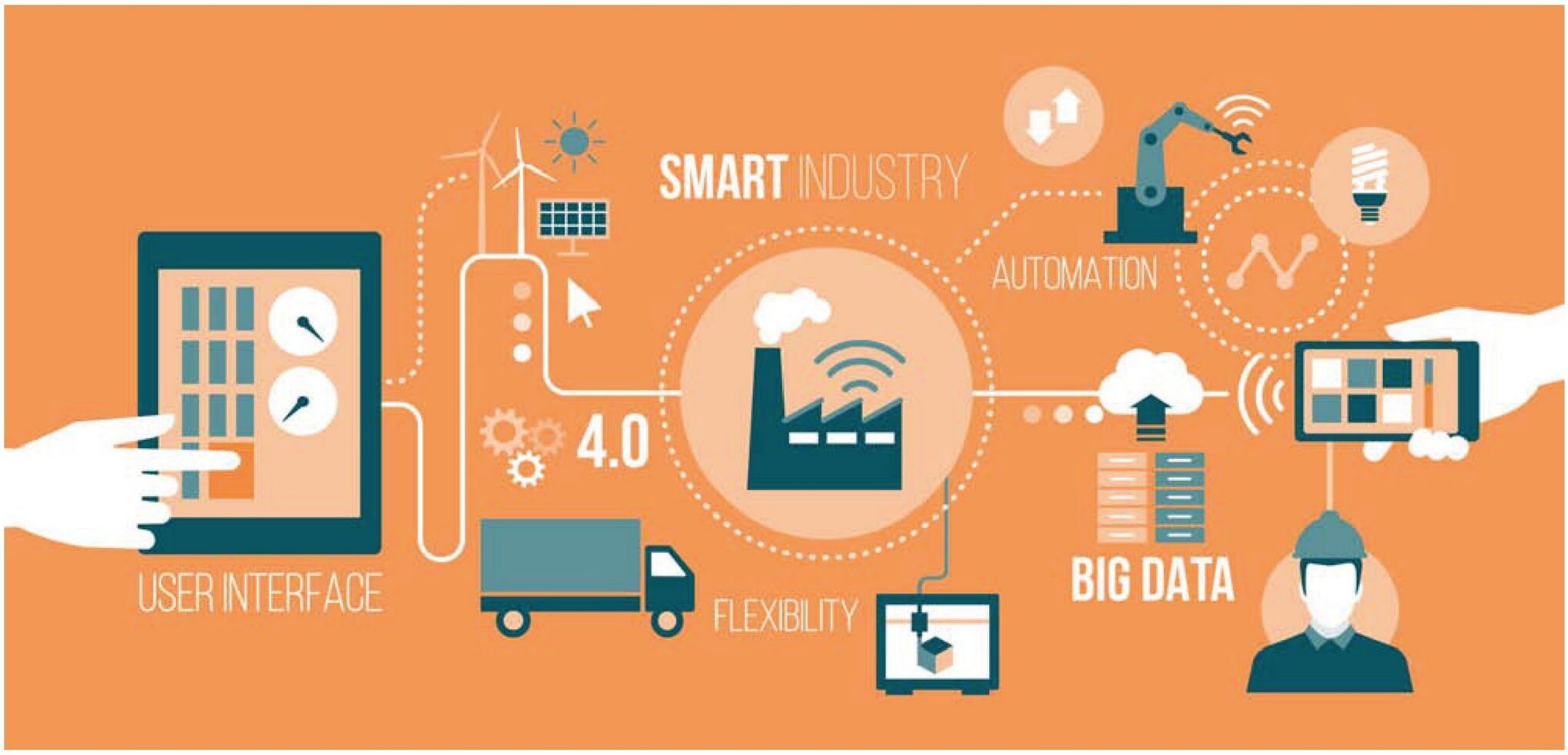
IoT devices also accept commands from the dashboard. For example, HVAC sensors accept commands to enable/disable air conditioning as well as set target temperature. The commands are sent using the platform API. END

By: Narasimha Sekhar Kakaraparthi

The author has more than 27 years of software industry experience in product development for hybrid cloud and telecommunications. He is currently working as principal architect for virtual desktops and end user services at Wipro Technologies. He is a senior member of Wipro's eminent architects group called Distinguished Member of Technical Staff (DMTS).

IoT Analytics: Making Sense of Big Data

This article covers the characteristics of IoT analytics and the challenges faced when adopting it in the industry. It also presents real world use cases of IoT analytics.



The standard definitions of IoT are:

- "A network of physical objects or things accessed through the internet, as defined by technology analysts and visionaries. These things contain embedded technology to interact with internal states or the external environment. In other words, when things can sense and communicate, it changes how and where the decisions are being made and who makes them." – *Cisco*
- "The IoT can be viewed as a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies." – *ITU*
- "An IoT system is a network of networks where, typically, a massive number of objects, things, sensors or devices are connected through communications and information infrastructure to provide value-added services via intelligent data processing and management for different applications (e.g., smart cities, smart health, smart grid, smart home, smart transportation, and smart shopping)." — *IEEE Internet of Things Journal*

Overall, the Internet of Things (IoT) is a network of devices which can sense, accumulate, and transfer data over the internet without any human intervention. IoT and data remain intrinsically linked together. Data consumed and produced keeps growing at an ever-expanding rate.

Internet of Things (IoT) analytics assesses the wide range of data collected from IoT devices. It analyses vast quantities of data and produces useful information called Big Data. It addresses large, diverse, complex, longitudinal, and/or distributed data sets generated from instruments, sensors, internet transactions, emails, videos, click streams, and/or all other digital sources available. The processing and integration of this heterogeneously generated data is done by IoT analytics.

IoT analytics helps gain insights into the operations of an enterprise, boosts productivity, increases the quality and flexibility of operations, and minimises material consumption.

Industry adoption of IoT

According to Gartner, more than 30.9 billion IoT devices will be in use worldwide by 2025. A forecast by International Data Corporation (IDC) estimates that there will be 41.6 billion IoT devices in 2025, capable of generating 79.4 zettabytes (ZB) of data.

The global Internet of Things market is estimated to rise from US$ 478.36 billion in 2022 to US$ 2465.26 billion by 2029 at a CAGR of 26.4 per cent during the forecast period, as per *Global Wire.*

According to McKinsey, by 2030, the IoT market will be valued between US$ 5.5 trillion and US$ 12.6 trillion globally, including the value captured by consumers of IoT products and services.

Key characteristics of IoT analytics

IoT analytics integrates business intelligence and data analytics, including conventional and Big Data. Its characteristics are explained briefly below.

Volume: The volume of data processed by IoT data solutions is substantial and ever-growing. High data volumes demand massive data storage, data processing, data preparation, data curation, and data management processes. Big Data requires processing high volumes of data, that is, data of unknown value — for example, network traffic, sensor-enabled equipment capturing data, and many more. Some of the data sources that are responsible for generating high data volumes are:

- Scientific and research experiments
- Sensors, such as temperature, pressure and GPS sensors
- RFIDs
- Smart meters and telematics

Velocity: In IoT data analytics environments, massive data gets accumulated in a short time, as sensors collect data pretty quickly. The velocity of data arrival within an enterprise's perimeter is directly proportional to the time taken to process the data. It demands highly elastic, highly available data processing and massive data storage solutions for the enterprise.

Variety: IoT data is primarily unstructured or semi-structured. Such data types require additional processing to derive meaning and the supporting metadata. Further complexity arises when data from a known source changes without notice.

Veracity: Veracity refers to the quality or fidelity of data. Assessment of data that enters IoT data environments for quality can lead to data processing activities to resolve invalid data and remove noise. Data acquired in a controlled manner usually contains less noise than data acquired via uncontrolled sources. Data with a high signal-to-noise ratio has more veracity than data with a lower ratio.

Value: There is a range of quantitative and investigative techniques to derive value from data. The cost of data storage and computing has fallen exponentially, thus providing an abundance of data for statistical sampling and other techniques.

Response time and reliability: The essential requirement of IoT data analytics is better response time and reliability. The solution is built on three fundamental factors: deterministic analytics, data computation, and quick response time.

Bandwidth: The massive amount of data generated by IoT devices is gathered from control systems. Handling of this data requires better bandwidth. The increased network and infrastructure required to transfer data from one domain to another is compensated by creating valuable insights.

Security: Transferring raw data outside of controlled areas comes with security concerns and the associated costs. It may be more efficient to perform data analytics locally and share the necessary summaries with other domains.

Analytics maturity: Analytics involves the processing of raw data (measurements) into descriptions (information), and then contextualizing the results (knowledge) and benefiting from historical experience (wisdom). The maturity of the processed raw data depends on the design of the analytics solution.

Temporal correlation: Applying analytics closer to where the data is generated reduces the burden of correlation. It helps IoT systems to correlate the data between multiple sensors and process control system states.

Provenance: Performing analytics at a lower architecture tier maintains the genuine sources of the data as it progresses through the IoT layers.

Principles of IoT analytics

Architecture principles provide a basis for decision-making when developing IoT data analytics solutions and design. They form a structured set of ideas that collectively define and guide solution architecture development, harmonising decision-making across an organisation.

The principles that guide enterprises in the way they approach IoT analytics data are:

- *Data is an asset:* Data is an asset that has a specific and measurable value for the enterprise.
- *Data is shared:* Data must be shared across the enterprise and its business units. Users have access to the data that is necessary to perform their activities.
- *Data trustees:* Each data element has trustees accountable for data quality.
- *Common vocabulary and data definitions:* Data definition is consistent, and the taxonomy is understandable throughout the enterprise.
- *Data security:* Data must be protected from unauthorised users and disclosure.
- *Data privacy:* Privacy and data protection is considered throughout the life cycle of a Big Data project. All data sharing conforms to the relevant regulatory and business requirements.

Data integrity and the transparency of processes: Each party to a Big Data analytics project must be aware of and abide by their responsibilities regarding the provision of source data and the obligation to establish and maintain adequate controls over the use of personal or other sensitive data.

Key challenges of IoT analytics applications

Heterogeneity: IoT data streams tend to be multi-modal and heterogeneous in their formats, semantics, and velocities. To access IoT data sources, database drivers and adaptors need to be deployed in Big Data applications. These applications incorporate features like interoperability, transaction management, and common data formats to address the heterogeneity.

Data quality: Different manufacturers provide various sensor categories and types. Substandard and unreliable sensors generate noisy and incomplete IoT data streams. Usage of these data streams leads to uncertainty and inaccuracy in IoT analytics.
To improve the data quality and eliminate the noise, statistical models and probability based applications are developed to feed into IoT data analytics applications to generate an accurate output.

Real-time data sets: IoT devices provide high-speed and real-time data. In the Big Data ecosystem, data streaming platforms consume real-time data sets to generate valuable insights.

Location and time: IoT data comes with temporal and spatial metadata information, directly associated with its business value in each application context. These spatial and temporal dimensions can serve as a basis for dynamically selecting and filtering streams towards analytics applications for specific timelines and locations.

Security and privacy: IoT applications collect and process personal data and device data. IoT analytics helps in exhibiting strong IoT security requirements and managing privacy sensitivities. The data generated by IoT data analytics needs to be encrypted, secured, and stored.

Data processing: IoT data analytics systems need to follow the steps listed below for data processing:

- Collection of data aggregated from a variety of sources, in a range of formats, and at multiple frequencies
- Understanding the data and categorising it
- Preparation of data
- Processing data from different sources
- Storing the information in time series for analysis
- Analysing the results
- Building applications to fulfil business processes

IoT analytics architecture

The requirements for an IoT analytics system with respect to its architecture are as follows.

- *Context:* Capture the context at all times, 24 hours a day, 365 days a year.
- *Standards:* Leverage standard protocols of communication between IoT devices and enterprise systems.
- *Scalability:* Always respond even during increased load; increase capacity proportionally as resources are added.
- *Data management:* Efficiently manage enormous volumes of data.
- *Connectivity:* Provide high network connectivity for large data payloads and continuous streaming.
- *Security:* Move and encrypt information securely even as IoT introduces new risks and vulnerabilities.
- *Interoperability:* Network all systems together and ensure interoperability of all data.
- *Device management:* An ability to connect to devices remotely at scale to manage them such as updating security credentials and firmware updates.
- *AI-driven analytics:* Enrich and explore prescriptive and predictive analytics data to deliver actionable insights.

Figure 1 shows the logical architecture of the IoT analytics system with key components and layers. While there exist many standard logical architectures for IoT analytics, this one is based on my practical experience across various domains and technologies.

The various layers for an enterprise application are classified as:

- Data source layer
- Edge layer
- Analytics layer
- Application layer

Data source layer*:* This layer consists of IoT devices and IoT gateways. Sensors and actuators are essential components of IoT.

Sensors sense the environment to monitor and detect changes of a particular attribute and collect the data, such as a change in temperature, heat, or cold. The output of the sensor is an electrical signal.

Actuators are devices used to control the system. They receive electrical output from sensors and produce a physical change. For example, if the sensor detects a change in temperature in a room, the actuator can turn a device on or off to maintain and control it.

These sensors and actuators establish a link between the physical and digital world via an IoT gateway. This layer basically deals with the gathering of specific data from data sources. This data is then passed to the edge layer so that it can be securely transmitted to the analytics layer.

Edge layer*:* The edge layer is an entry point for all things or devices to capture and process the data. The layer consists of a data store, services to analyse the data streamed from the IoT devices and execute AI-driven analytics locally in real-time, security mechanisms (including data encryption), and control applications.

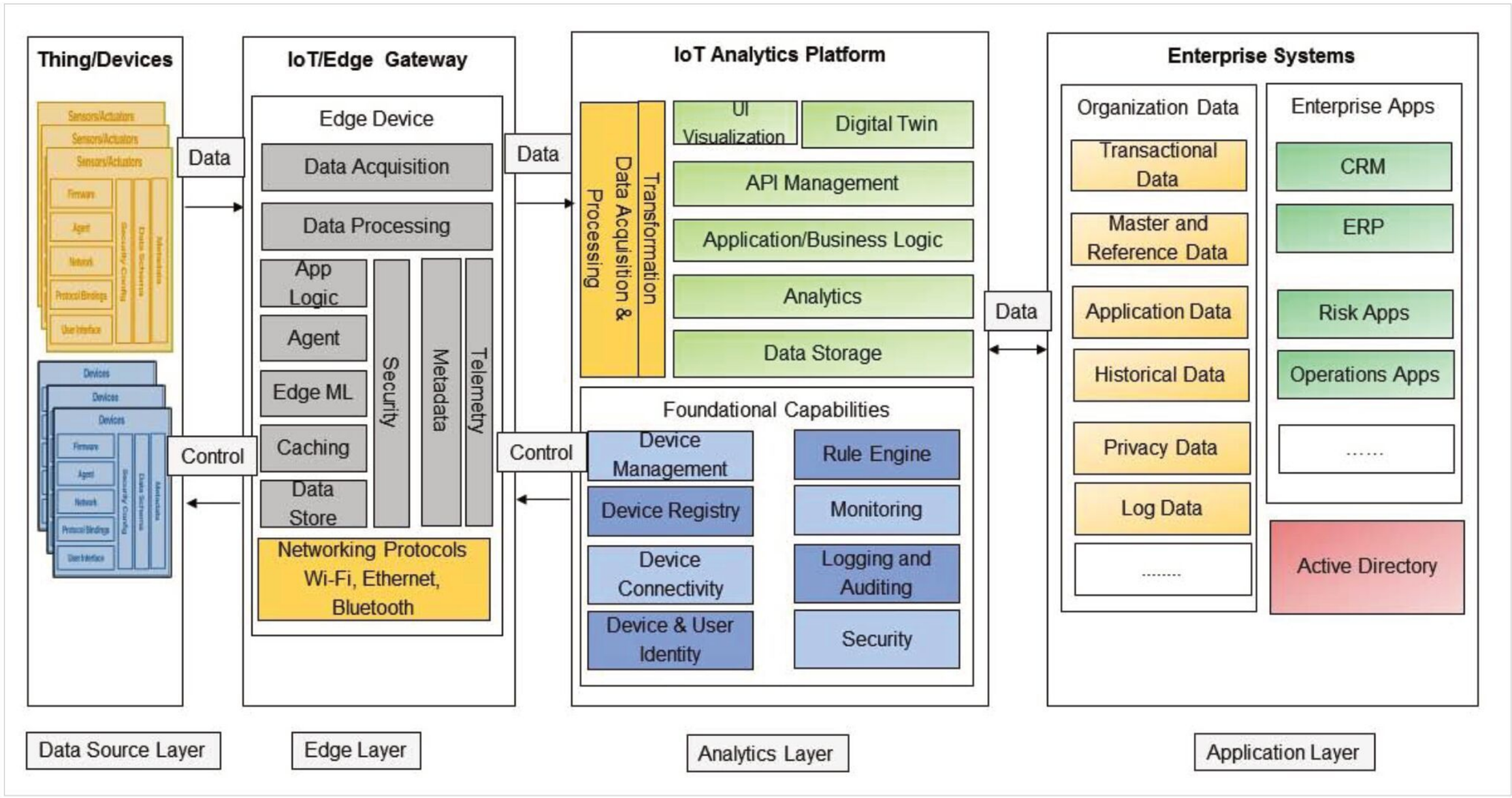


Figure 1: IoT analytics logical reference architecture

It consists of an IoT gateway that facilitates bi-directional communication with the platform and enterprise systems. It also implements edge intelligence with different processing capabilities, enabling connectivity between devices, routing, filtering, and protocol and identity translation.

Analytics layer: The primary purpose of this layer is to interface with the edge layer to receive, store, process, and analyse the data. In addition to data acquisition capabilities, it offers AI-driven analytics for intelligence and visualisation, device management, security, connectivity and network management, and operations.

The data from devices is collected and aggregated from the IoT device gateway. Real-time AI-driven analytics and visualisation combined with machine and deep learning models are applied to the data to derive intelligence.

Machine learning is also performed over the collected device telemetry data, and this data is blended with other enterprise data or secondary sources of telemetry from other systems or organisations.

Application layer*:* This layer acts as an interface to interact with applications for end users. Devices like smartphones, computers and tablets are used as a medium to make IoT applications interactive.

Real world use cases of IoT analytics

Implementation of IoT analytics based solutions helps enterprises eliminate redundant assets and products, leading to better operational efficiency and cost savings.

IoT for digital government: The vision of any government in the digital era is to develop connected and intelligent IoT based systems that contribute to the economy, improve citizen satisfaction, and lead to a safer society and environment sustainability.

Processing of data feeds from various sources like weather and environmental sensors, traffic signal cameras, parking zones, etc, leads to improved government-IoT agency coordination and better services to citizens.

IoT for manufacturing*:* IoT in manufacturing allows richer insights in real-time. It can facilitate the production flow in a manufacturing plant, as IoT devices automatically monitor development cycles and manage inventories.

IoT for supply chain management: Manufacturers use IoT to monitor their supply chain and get estimates of the available resources. This includes information regarding the status of the work on the floor, equipment collection, and the delivery date of required materials.

IoT for smart homes: IoT helps to analyse and optimise use of devices in a smart home. This can mean simple switching off or dimming of lights, and modifying multiple home settings to optimise energy use. It also helps discover problematic energy consumption by older and damaged appliances, or faulty system components.

IoT for media: IoT based applications in media and advertising lead to a customised experience in which the system analyses and responds to the needs and interests of each customer. It helps in understanding the customer behaviour

patterns, buying habits, preferences, culture and other characteristics.

IoT for healthcare: IoT based applications/systems of healthcare enhance the traditional technology used today. These devices help increase the accuracy of the medical data collected from a large set of devices connected to various applications and systems. They also help gather data to improve the precision of the medical care delivered through sophisticated integrated healthcare systems.

- IoT devices give direct 24/7 x 365 access to the patient in a less intrusive way. IoT based analytics and automation allow health providers to access patient reports before they arrive at the hospital. They also help improve responsiveness in emergency healthcare.
- IoT-driven systems are used for continuous monitoring of a patient's health status. These employ sensors to collect the physiological information analysed and stored on the cloud. Doctors access this information for further analysis and review. This way, a continuous automated flow of information is provided.
- Patients' health data is captured using various sensors, analysed and sent to the medical professional for proper remote medical assistance.

IoT for education*:* IoT customises and enhances education by optimising content and forms of delivery.

- *Student tracking:* IoT facilitates the customisation of education to give students access to what they need.
- *Instructor tracking:* Educators can use the IoT platform for customised instructional designs for their students.
- *Facility monitoring and maintenance:* IoT based applications improve the professional development of educators.
- *Data from other facilities:* IoT also enhances the knowledge base used to establish education standards and practices. It introduces large, high-quality, real-world data sets into educational design.

IoT for agriculture: IoT minimises human intervention in farming analysis and monitoring. IoT based systems detect changes in crops, the soil, etc.

- *Crop monitoring:* IoT based devices can be used to monitor crops and the health of plants using the data collected from them. These devices can also monitor pests and diseases proactively in the early stages of infestation.
- *Food safety:* IoT helps connect the entire supply chain for ensuring the food is safe when it reaches the customer.
- *Climate monitoring:* IoT based devices are used to monitor temperature, humidity, the intensity of light, and moisture of the soil. The data is ingested into the central repository to trigger alerts and automate water, air, and crop control.
- *Logistics monitoring:* Location based IoT devices can be used to track vegetables and other farm products during transport and storage.
- *Livestock monitoring:* Farm animals can be monitored using IoT devices to detect potential signs of disease. The data can be analysed from the central repository, and the farmers can share relevant information.

IoT analytics has a number of benefits that are helping transform businesses and different industries.

- Greater transparency; transaction histories are becoming more transparent
- Increased visibility and better control, resulting in quicker decision-making
- Flexible scaling of business needs and expansion in new markets
- Reduction in operational costs due to automation and better resource utilisation
- New revenue streams, resulting from the clearing of operational challenges
- Enhanced customer experience based on an analysis of the purchase history

IoT is one of the most significant sources of Big Data, which is rendered useless without analytics power. IoT analytics comes into play when voluminous amounts of data needs to be processed, transformed, and analysed at a high frequency. It helps to provide advanced services across many domains. END

Acknowledgements

I would like to thank Santosh Shinde of BTIS, enterprise architecture division of HCL Technologies Ltd, for giving the required time and support while writing this article as part of architecture practice efforts.

By: Dr Gopala Krishna Behara

The author is an enterprise architect in BTIS, the enterprise architecture division of HCL Technologies Ltd. He has 26 years of experience in the IT industry.

A Complete Guide to Getting Started with Arduino

Building DIY projects with Arduino is super easy and a lot of fun. However, it may not always be as straightforward for beginners with absolutely no background knowledge in hardware or programming. In this article, we cover everything you need to know for getting started with your first Arduino project. That includes an overview of the Arduino board, the IDE, the programming language and a simple project to wet your beak.

Arduino is an open source project developed for students, hobbyists, artists and programmers who are interested in combining hardware and software to make DIY (Do It Yourself) devices and utilities. It was created in 2005 by Massimo Banzi and his colleagues and students at the Interaction Design Institute Ivrea, Italy. The primary purpose behind the creation of the Arduino project was to provide enthusiasts and professionals with a platform consisting of a programming language, an IDE, and a printed circuit board to create simple and effective DIY projects. The project quickly gained steam and by 2011, an estimated 300,000 Arduino boards were in circulation globally. By 2013, that number was 700,000.

Today, the Arduino project is a mammoth with 100+ devices consisting of various types of boards, shields, carriers, kits and other accessories. All the information regarding the Arduino project is available at its official website *arduino.cc*

So what makes Arduino so attractive? Go to YouTube and search 'Cool Arduino Projects' and the answer will become glaringly obvious. Arduino offers developers the freedom and autonomy to get creative with their circuit design, and write innovative code to make DIY devices and utilities that not only help make life better but are also artistic and innovative in their approach.

Consider life before Arduino. If you wanted to build a light that turns on automatically, you would have to design the entire circuit diagram yourself. Solder it all together on a board. Write low level machine code yourself and create your own mechanism to power the whole thing up.

With Arduino, you can do all that in 5 minutes. Just connect a few pins here and there using an Arduino and a breadboard. Write the code in a language that is as simple as C and run the code with the click of a button. That's it! Arduino massively simplifies your life as an engineer. What's more, it is designed keeping students in mind, which means it is cheap to buy.

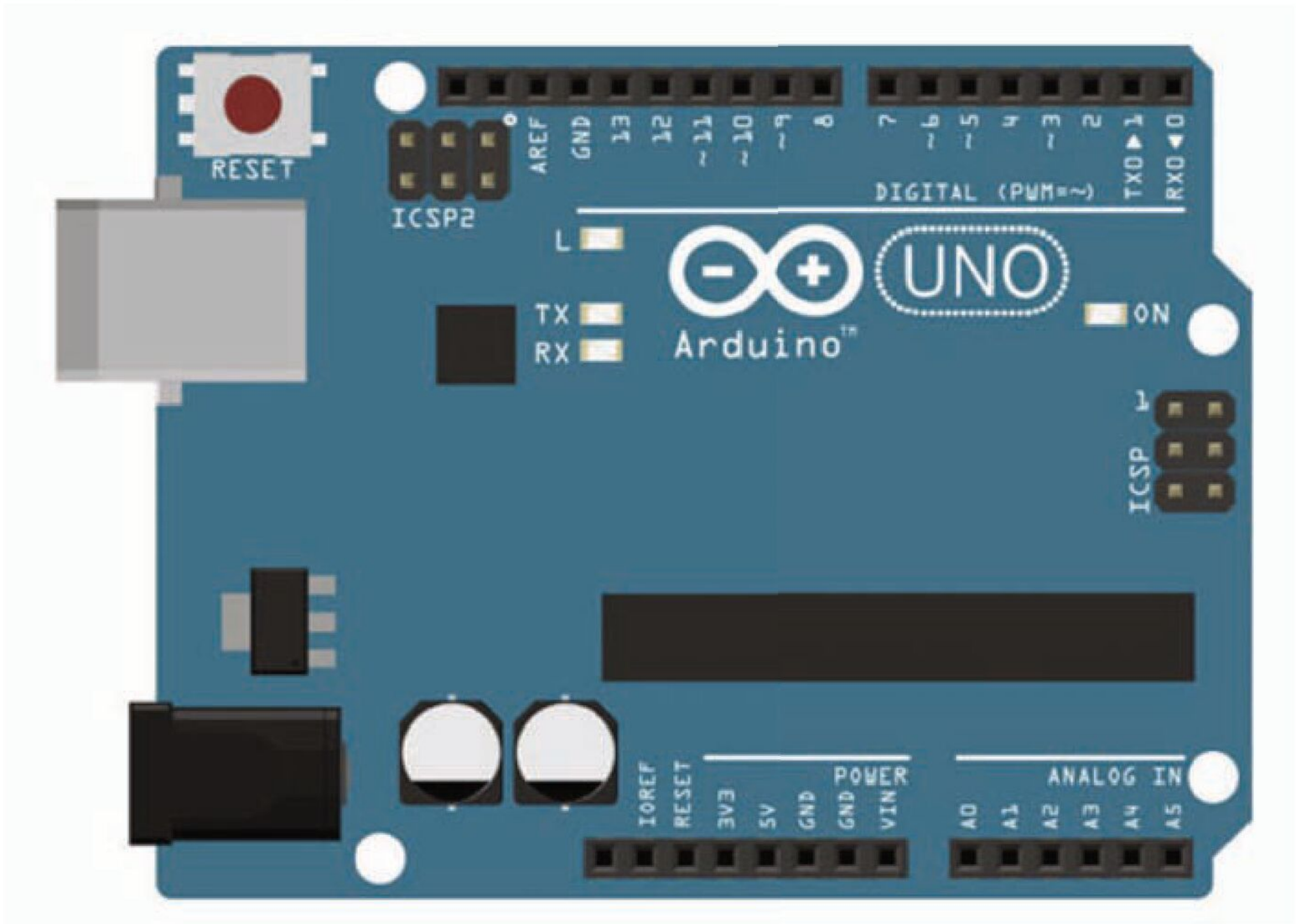


Figure 1: Arduino Uno

The anatomy of an Arduino device

Before we get to the code part, it is necessary to have a basic understanding of the Arduino board. Every Arduino board is differently constructed based on the purpose it serves. For this article, we shall be using the beginner-friendly and most popular Arduino board, i.e., the Arduino Uno. Figure 1 shows the anatomy of an Arduino Uno.

There are four major components of an Arduino Uno board — digital pins, analog pins, USB connector, and battery power.

Digital pins: As shown in Figure 1, there are 14 digital pins in the Arduino

Uno. Digital pins can be used both as INPUT and OUTPUT ports. They have two states, i.e., HIGH and LOW. A HIGH state indicates the presence of current in the pin whereas the LOW state indicates the opposite. Digital pins can be used for various purposes such as blinking of LED lights. When set to HIGH, the light will turn on and when set to LOW, it will turn off. This can be done in a loop to create a blinking effect.

Analog pins: There are six analog pins in an Arduino Uno, as shown in Figure 1. They are indexed as A0 to A5 on the board. These pins are used for reading sensor data. A lot of the work done in your Arduino projects will require sensor data. For instance, if you are creating a light that turns on when it detects movement, then that will require you to use a sensor to detect motion. The analog pins can register readings between the values of 0 and 1023.

USB connector: The USB connector helps you to connect your Arduino to the computer so you may upload your code.

Battery power: You can use the battery power connector as shown in Figure 1 to connect your Arduino Uno to a remote battery. This will make your device portable and deployable to places where you may not be able to connect it to a computer or a continuous power source.

The Arduino IDE

To write code for your Arduino project, you will have to download and install the latest Arduino IDE. You can download this from the official Arduino website *arduino.cc*

Figure 2 shows the default view of an Arduino IDE. Let us go through some of its major features. There are three major components in the IDE, i.e., Toolbar, Status window, and Serial Monitor window.

Toolbar: The Toolbar is the pane on the top as shown in Figure 2. It contains shortcuts to the most commonly used options in the IDE. We shall go through each shortcut in a left to right order.

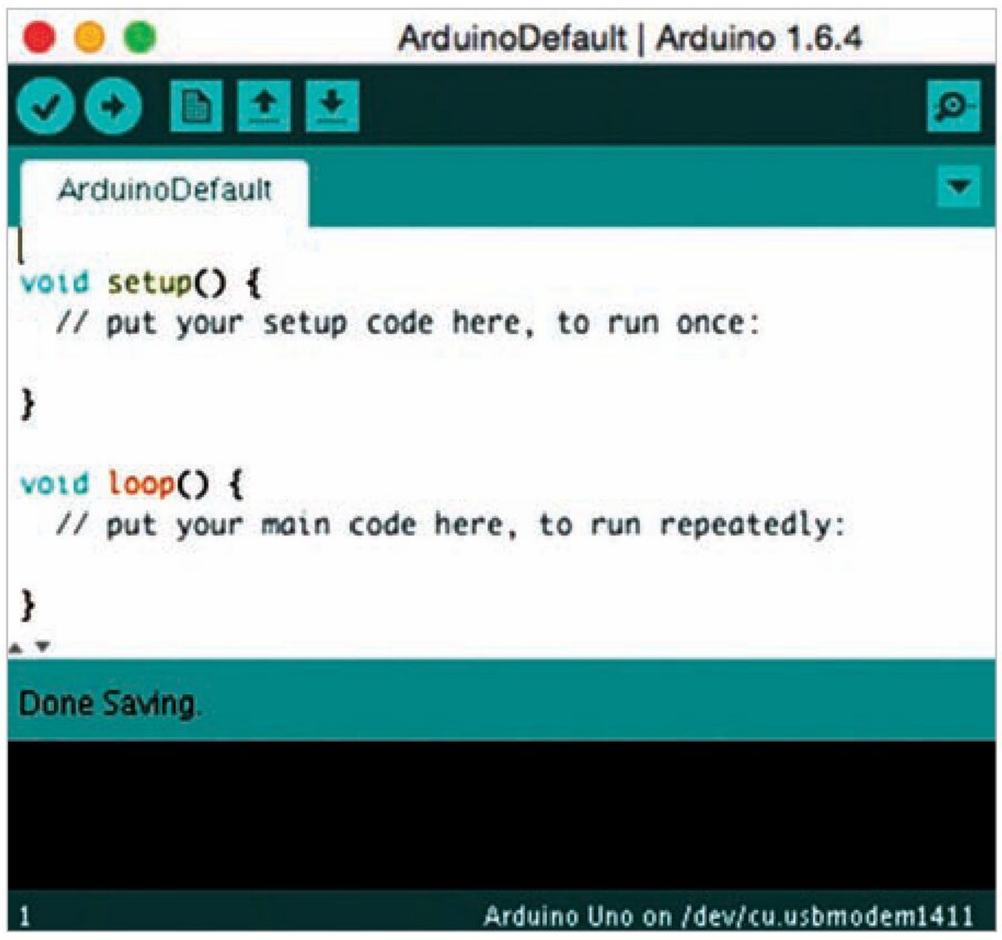


Figure 2: Default view of the Arduino IDE

1. *Compile:* This is the first button from the left used for compiling and verifying the code for any syntactic or logical errors.
2. *Upload:* The second button from the left, it is used for uploading the code onto the Arduino device. The results of each upload can be viewed in the Status window at the bottom.
3. *New:* This button is used for creating a new code document.
4. *Open:* This button is used for re-opening an already existing code document.
5. *Save:* This is used for saving the progress on a code document.
6. *Serial monitor:* This is the last button on the right that lets you access the Serial Monitor window.

Status window: Status window is the pane shown at the bottom of Figure 2. Whenever you compile or upload your code to the Arduino device, the Status window shows the status of your actions. It is also used for displaying any error messages.

Serial Monitor window: The Serial Monitor window logs all the messages generated by your code. Whenever we use commands such as *Serial.print()* or *Serial.println()* to log any details of our code, they will be displayed on the Serial Monitor window, as shown in Figure 3.

Arduino programming constructs

Before we begin to write and execute our first Arduino code, let us get familiar with the Arduino programming language. In this section we will cover all the basic constructs used in the Arduino programming language. These are more than enough to get you started with writing beginner to intermediate level code for your Arduino projects.

The basic programming constructs are listed below.

1. *Int:* Used for integer values such as 12345
2. *Float:* Used for decimal values such as 1.34
3. *char[]:* Used for string values such as ‘Car’
4. *HIGH:* Denotes digital pin with current
5. *LOW:* Denotes digital pin without current
6. *INPUT:* To assign a pin to read
7. *OUTPUT:* To assign a pin to set
8. *A0 - A7:* Constants used for each analog pin
9. *0 – 13:* Values used for each digital pin
10. *analogRead():* Returns the value of the analog pin
11. *analogWrite():* Sets the value of the analog pin
12. *digitalRead():* Returns the value of the digital pin
13. *digitalWrite():* Sets the value of the digital pin
14. *Serial.begin():* Initialises the serial monitor
15. *Serial.print():* Outputs a message on the serial monitor
16. *Serial.println():* Outputs a message

on the serial monitor in a new line
17. *delay():* To add wait time in processing in milliseconds
18. *setup():* The standard function in Arduino to be only called once
19. *loop():* The standard function in Arduino to be called repeatedly
20. *if …. Else:* Used for conditional statements
21. *//:* Used for adding single line comments
22. */* */:* Used for adding multi-line comments
23. *#define:* Used for defining a constant value
24. *#include:* Used for including external libraries

Structuring the Arduino code

Arduino code should ideally be structured in three parts, i.e., external libraries, constants and variables, and functions. Each part follows the other. We start by calling the necessary external libraries for our project. Next, we define any necessary constants or variables required. This is followed with the required functions.

There are two standard functions that must be called in every program, i.e., *setup()* and *loop()*. The *setup()* function is used for initialising the code while the *loop()* runs certain methods for a certain number of iterations depending upon your requirements.

Shared below is an example of what an ideal code structure should look like. We shall output the message 'Hello Internet of Things' on the Serial Monitor window every 3000 milliseconds.

```
// External Libraries
#include <SPI.h>
// Constants and Variables
char text = "Hello Internet of Things";
// Custom function(s)
void sayHello()
{
     Serial.println(text);
}
// Standard functions
void setup()
{
     Serial.begin(9600); // For initializing the serial port
}
void loop()
{
     sayHello();
     delay(3000);
}
```

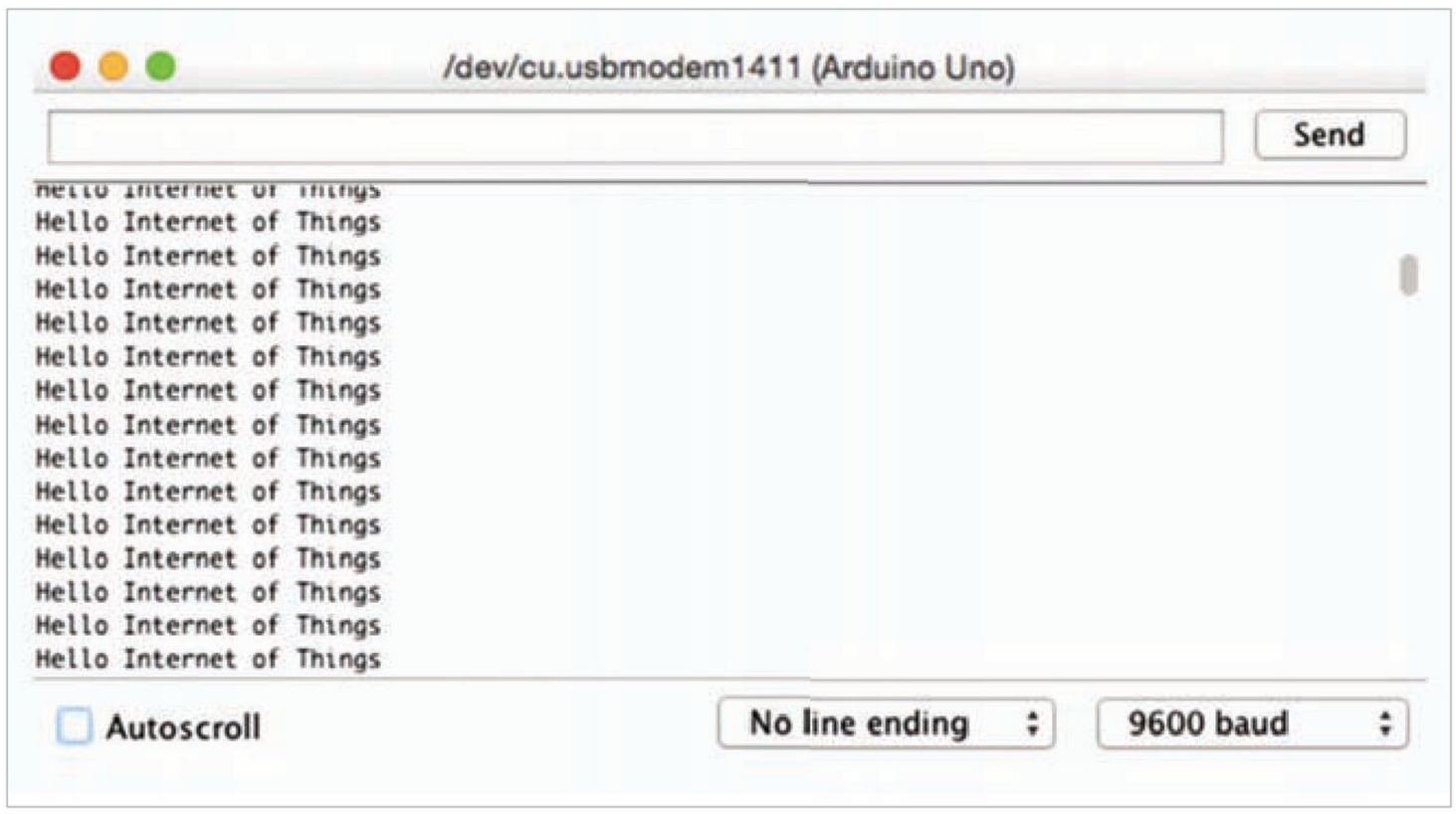


Figure 3: Output of *sayHello* program

Executing the Arduino code

Now that we have the code written and structured in the proper order, we shall start the execution process. In order to run the code, you will need to do the following.

Step 1: Connect your Arduino device to the computer using the USB cable.

Step 2: In your IDE, select the Arduino board that you will be using by clicking *Tools > Board.*

Step 3: Next, you will be selecting the port by clicking on *Tools > Port.* Select the port that corresponds to the USB modem of your Arduino device.

Step 4: Click on the *Upload* button and open the Serial Monitor window. Figure 3 shows the output on this window.

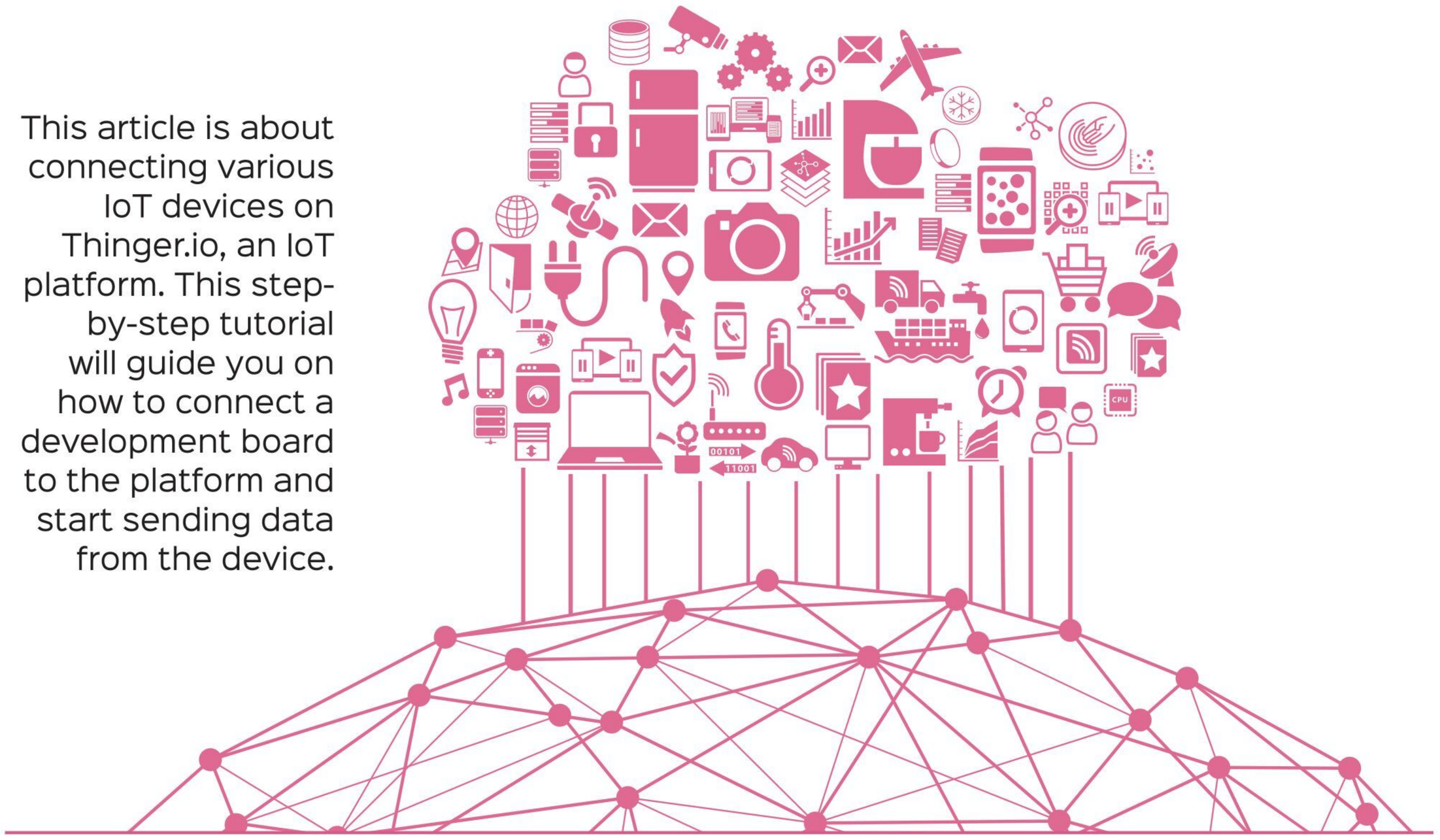
Learning to use an Arduino device can be a rewarding experience. Whether you end up liking IoT or not, just the feel of using an Arduino will give you a taste of programming hardware from the ground up. The greatest challenge that beginners face with Arduino is not the code as much as the debugging. Many times, newbie developers write code without any unit testing. My recommendation if you are just getting started with Arduino is not to get frustrated if your programs don't run properly. It can be challenging at first to find out where you may have gone wrong with your code or circuits. But over time, you will learn to create checks for your code at every level. Once you get a hang of that, the sky's the limit! Every day, people are coming up with new and interesting ways of using an Arduino device in their DIY projects. I look forward to seeing what you do with it. END

By: Mir H.S. Quadri

The author is a research analyst with a specialisation in artificial intelligence and machine learning. He is the founder of Arkinfo, which focuses on the research and development of tech products using new age technologies. He shares a deep love for the analysis of technological trends and understanding their implications. Being a FOSS enthusiast, he has contributed to several open source projects.

Connecting Internet of Things Devices on Thinger.io

This article is about connecting various IoT devices on Thinger.io, an IoT platform. This step-by-step tutorial will guide you on how to connect a development board to the platform and start sending data from the device.



Devices connected by the Internet of Things (IoT) achieve higher functionality because they interface via a distributed network with numerous physical objects. They are equipped with embedded software, sensors and connectivity choices with the objective of collecting and sharing data with each other as well as with the central point through the internet. This opens the doors to various challenges in terms of data processing, storage and security. IoT platforms bridge this gap between the world of physical objects and of actionable insights.

Every IoT ecosystem has its own unique set of input, processing and output elements. However, all of them have the typical flow setup shown in Figure 1, which consists of the components listed in Table 1.

1	Physical objects: People, things, machines, vehicles
2	Raw data: Reading of any parameter
3	Sensors: Conversion of any physical quantity to an electrical signal
4	Quick decision at the edge: Urgent decision making through limited resources
5	Storage: Physical/cloud based choices
6	Data analysis: Study and processing of data

Table 1: Components of IoT

An IoT platform is an on-premises software suite or a cloud service (IoT platform as a service [PaaS]) that monitors and may manage and control various types of endpoints. Open source IoT platforms are gaining popularity by blending hardware devices, APIs and online services for task management.

Thinger.io provides a scalable cloud base to connect devices. With the help of its admin console, it can be linked to a project through various boards like Raspberry Pi, Arduino, ESP8266, and Intel Edison. Open source libraries imported by the user assure scalability, security and speed through a couple of lines of code.

There are various components and applications offered on the Thinger.io menu bar. Our task for now is limited to adding a device — the first and preliminary step to start an ecosystem.

The first step involves signing up to the platform. Enter your personal details for the profile settings. Remember this also becomes your user name for all further processing on this

Figure 1: A typical Internet of Things setup

platform. Once you have entered your profile settings, you can move on to the statistics, as shown in Figure 2.

Go to the *Devices* tab and create a device. This will ask for more details, as shown in Figure 3.

The device type is generally a generic Thinger device (Wi-Fi, Ethernet, GSM). However, the user has the option to select HTTP devices (Sigfox, Lora, cURL)/ MQTT device/ NB-IoT. The Device ID is important and you have to remember it while interfacing with your code. Device information is optional and is limited to the user (for identification). But Device Credentials are vital. Each device has its own identifier/credential, so that a compromised device does not affect other devices. All your passwords in the server are stored securely using PBKDF2 SHA256 with a 32-byte salt generated with PRNG and a non-depreciable number of iterations. (Please remember, the password cannot be recalled later.)

If the steps have been implemented correctly till now, you should see your device disconnected when you press the *Add Device* button.

Using your new Device ID and the Device Credentials to connect the new device, you will need to install the required libraries or development environment. For the present example, we will be using the Arduino IDE along with an ESP8266 device.

The ESP8266 chip (Figure 4) from Espressif is the new generation of low-cost Wi-Fi chips after the TI CC3000/ CC3200. This small chip not only integrates the entire Wi-Fi features, but also a powerful programmable MCU. Depending on the board layout (ESP-01, ESP-03, ESP-07, ESP12, etc), it is attached to a programmable flash, ranging from 512K to 4M. This increases the available user code space, and makes possible other cool features like a small file system or OTA updates. I am using the ESP 12 module for the present illustration.

The device can be directly programmed from the Arduino IDE. The very first requirement is to install the board via the Arduino Boards Manager (Figure 5). From the *File* menu, go to *Preferences*, and enter *http://arduino.esp8266.com/stable/package_esp8266com_index.json* in the Additional Boards Manager URL.

The next step involves installing the ESP8266 board. Go to *Tools* > *Boards*

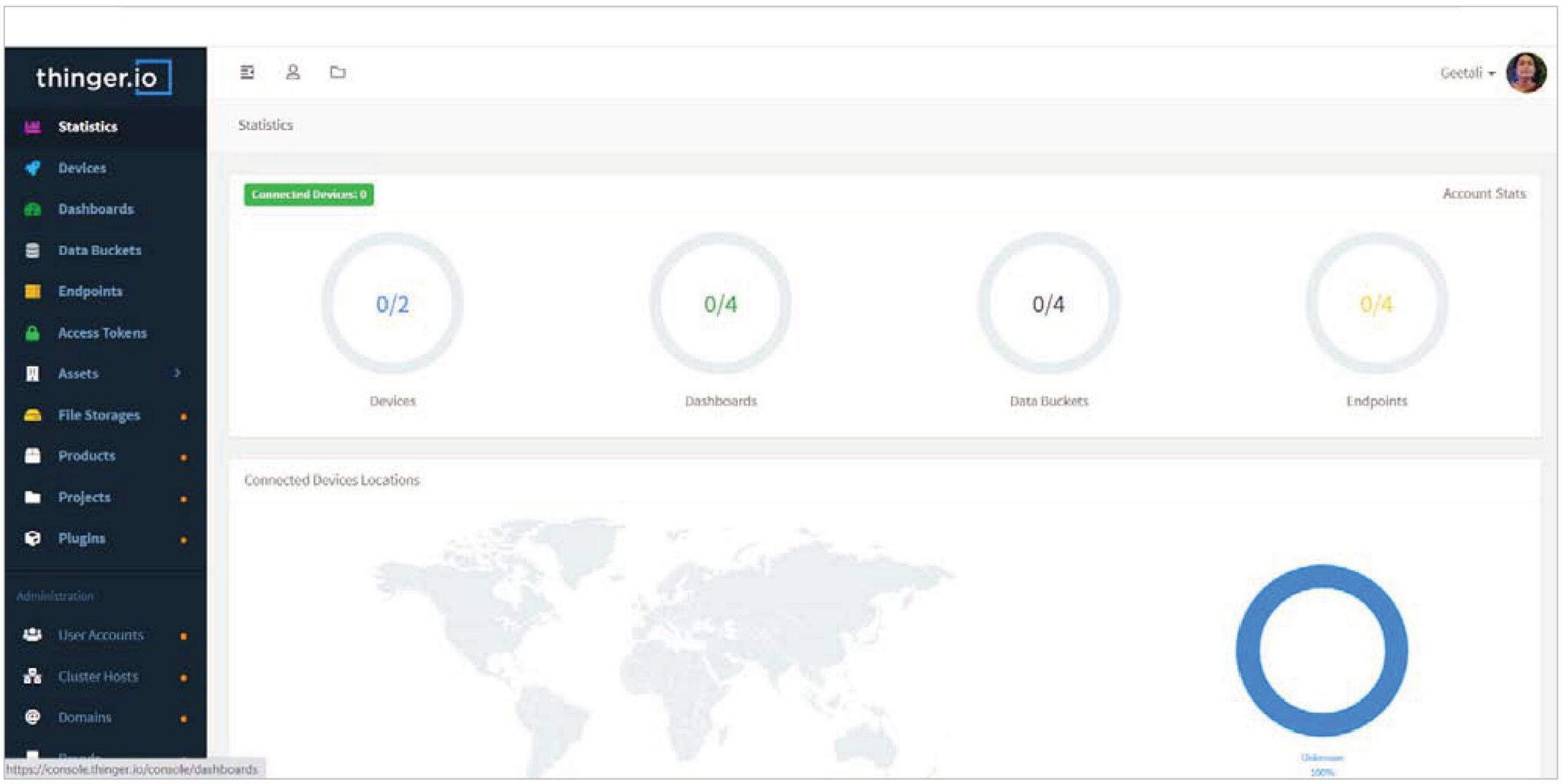


Figure 2: View of the statistics

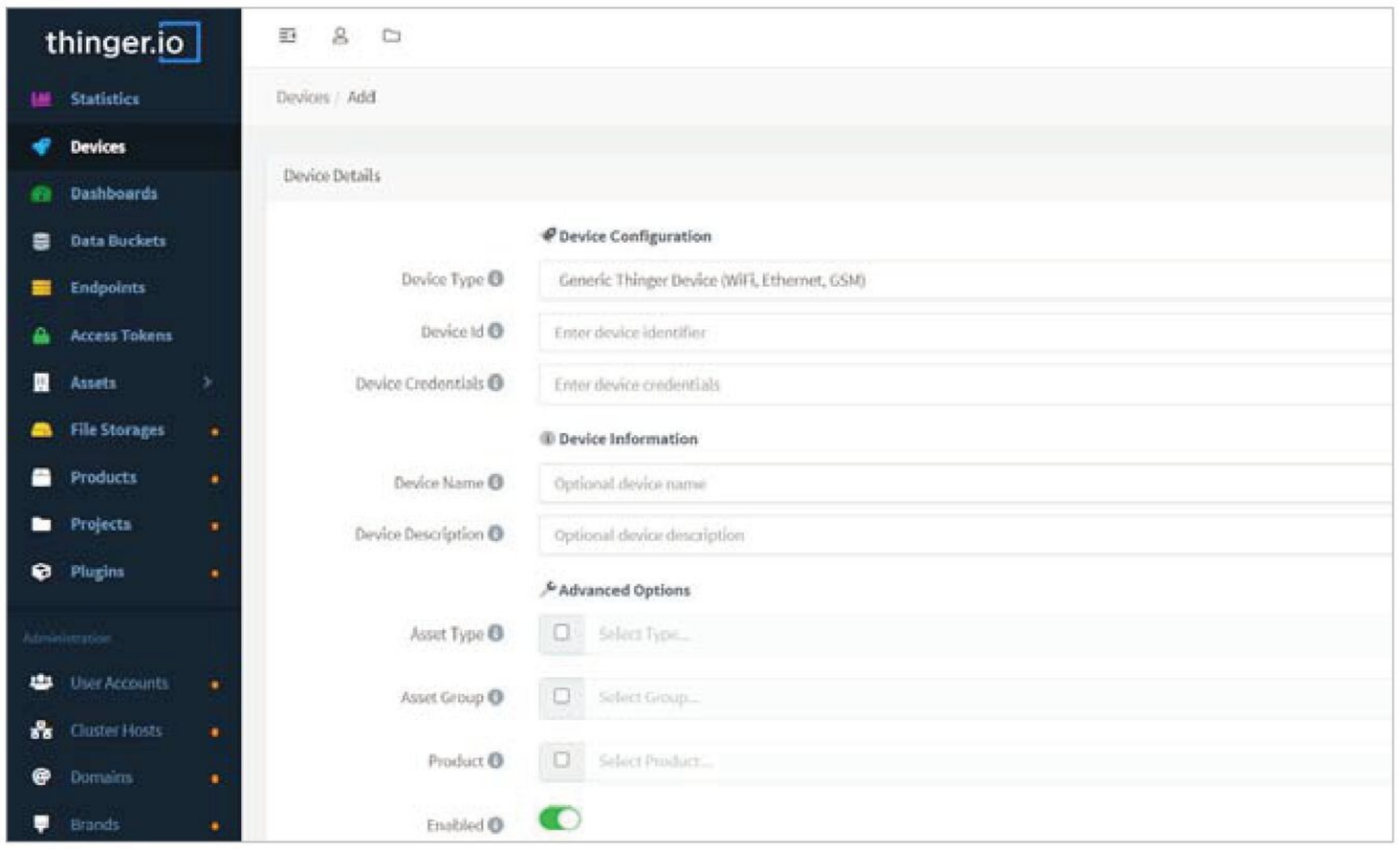


Figure 3: The *Device* tab from the menu

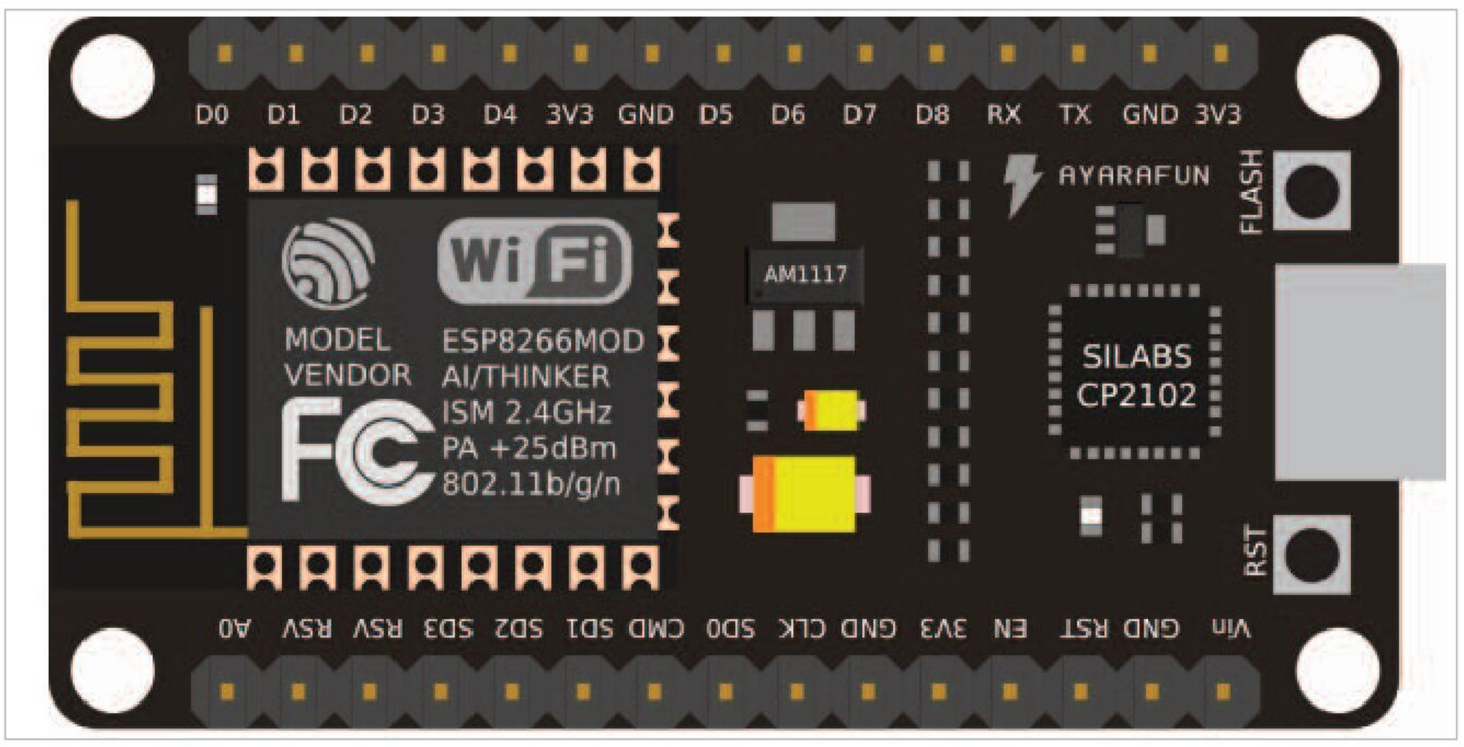


Figure 4: The ESP8266 chip

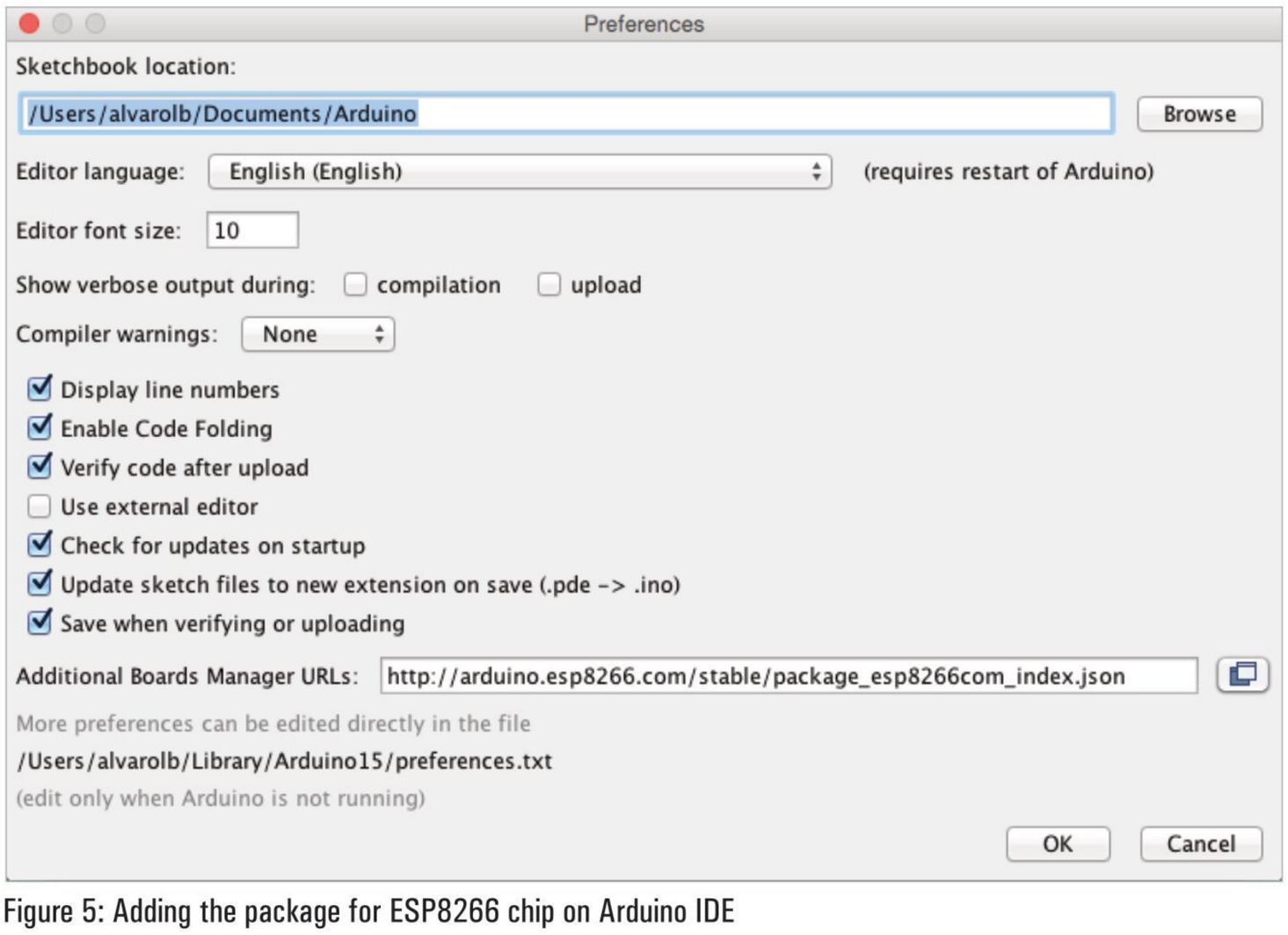


Figure 5: Adding the package for ESP8266 chip on Arduino IDE

> *Board manager*. Then search and install the esp8266 package (Figure 6).

This should now enable any program involving ESP8266 directly from the Arduino IDE. By going to *Tools* > *Boards* you should see the new ESP8266 boards installed. Select your board to be able to compile code for the ESP8266.

You can find additional information for the ESP8266 package in the ESP8266 GitHub repository. The easiest board to program is the NodeMCU, which does not require pressing *Flash + Reset* buttons for uploading the sketch. For other boards you will need to use a USB to serial converter (3v3!) and flash the sketch by setting some GPIOs to GND.

Use the code below to connect an ESP8266 device to the cloud platform in a few lines using the Wi-Fi interface. Please remember to modify the *arduino_secrets.h* file with your own information.

```
#define THINGER_SERIAL_DEBUG
#include <ThingerESP8266.h>
#include “arduino_secrets.h”
ThingerESP8266 thing(USERNAME, DEVICE_
ID, DEVICE_CREDENTIAL);

void setup() {
  // open serial for monitoring
  Serial.begin(115200);
  // set builtin led as output
  pinMode(LED_BUILTIN, OUTPUT);
  // add WiFi credentials
  thing.add_wifi(SSID, SSID_PASSWORD);
  // digital pin control example
(i.e. turning on/off a light, a relay,
configuring a parameter, etc)
  thing[“led”] << digitalPin(LED_
BUILTIN);
  // resource output example (i.e.
reading a sensor value)
  thing[“millis”] >>
outputValue(millis());
  // more details at http://docs.
thinger.io/arduino/
}
void loop() {
```

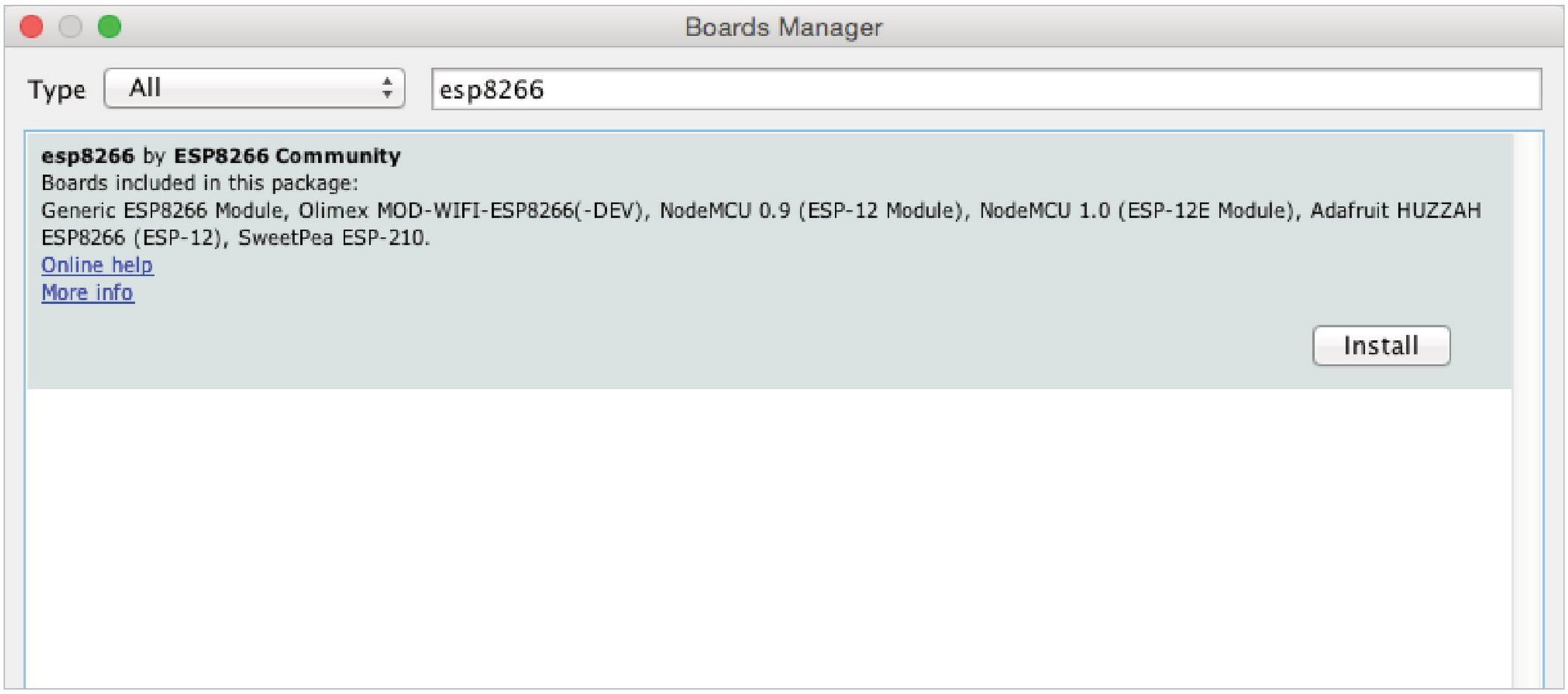


Figure 6: Installing the board for ESP8266 chip

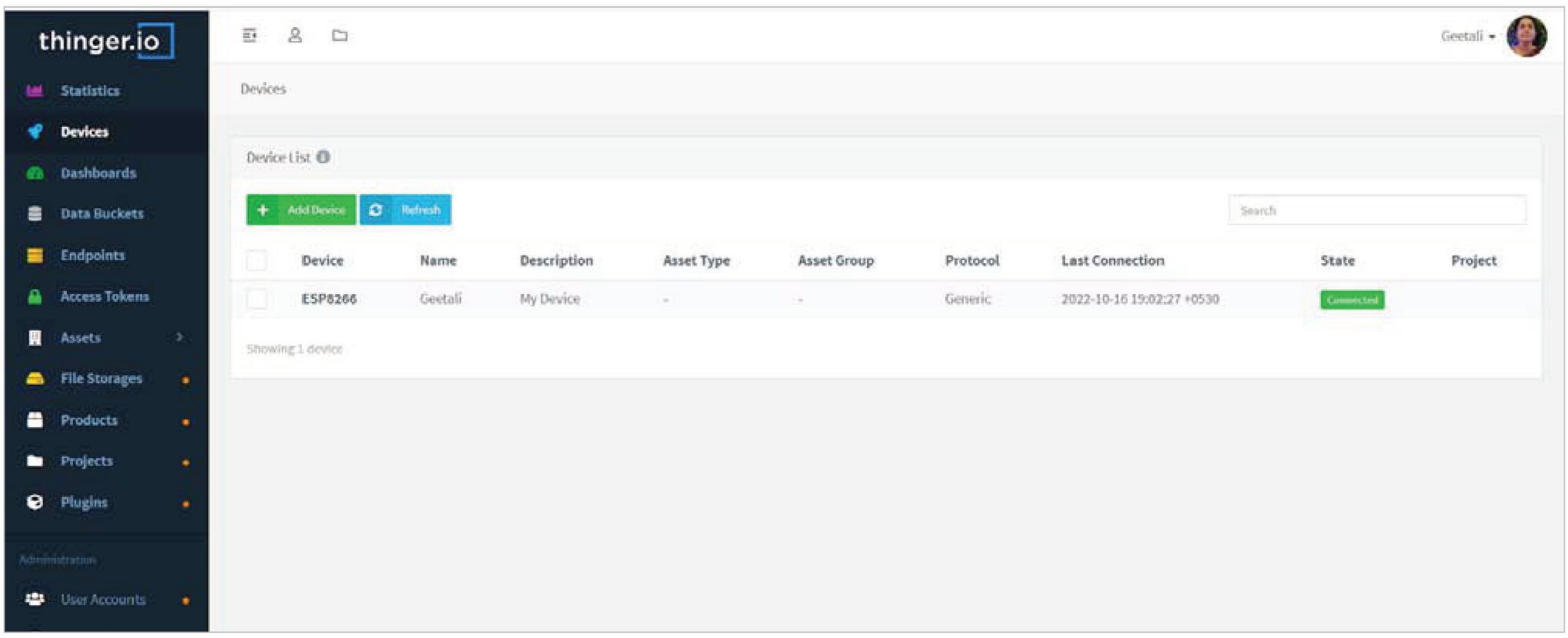


Figure 7: Device ESP8266 is now connected on the Thinger.io IoT platform

```
  thing.handle();
}
```

arduino_secrets.h

```
#define USERNAME "your_user_name"
#define DEVICE_ID "your_device_id"
#define DEVICE_CREDENTIAL "your_device_credential"
#define SSID "your_wifi_ssid"
#define SSID_PASSWORD "your_wifi_ssid_password"
```

Alternatively, install the *Thinger.io* custom library. Follow the path *File>Examples>thinger.io>ESP8266>serial debug*.

Compile and upload the code as we do for any other Arduino code. Once implemented, the inbuilt LED in the ESP8266 should glow.

If the steps are implemented correctly, Thinger.io should show that ESP8266 has been connected (Figure 7).

We just saw how to connect ESP8266 to Thinger.io. Do follow the steps to connect it or similar development boards to try out more possibilities with the platform. END

Reference

- *https://docs.thinger.io*

By: Dr Geetali Saha

This author is the university program co-ordinator for Internet of Things, Minor Degree Program, CVM University, Anand, Gujarat. She is also a faculty member at the Department of Electronics & Communication, GCET, Anand, Gujarat.

IoT Might Disrupt Lots of Technologies for Greater Good

"I believe IoT will disrupt multiple industries in the coming decade and beyond," says Manish Kothari, senior vice president of software development at Silicon Labs. In his keynote address at the VLSI Design Conference 2022, Manish talked about the challenges and future of IoT and its potential to be one of the most disruptive technologies of this century.

In today's time, safety and comfort are paramount, and automation is playing an active role in enhancing lifestyle, safety, and comfort along with improved efficiency. And to realise automation, gathering and processing data is the most crucial part, which is taken care of by the Internet of Things (IoT).

In a layman's terms, IoT is basically connecting different applications and sensors to the cloud or connecting them to the internet using a variety of different wireless protocols. The inclusion of IoT on a large scale has not just improved the quality of life but has also increased the complexity manifold.

IoT is simple yet complex

IoT solutions seem straightforward but are not as simple as they sound. For example, an embedded system, which is the backbone of an IoT system, is often a low-powered device performing a simple task, which can easily be performed by even a low-cost smartphone, or a decade-old microcontroller (MCU) combined with a simple sensor that is easily available in the market.

The real challenge here is not to complete the task or gather data but to do all of it in a space-constrained, budget-constrained, power-constrained system that can work for years without any human intervention. And not to forget about the security, which in an IoT system is paramount. Furthermore, there is a time-relevance aspect to the IoT solution, that is, the product and its components should remain relevant for at least 5 to 10 years.

"The IoT solution has at least a 5-to-10-year life," says Manish. "So, there is a time relevance aspect of the hardware and software, both of which cannot go obsolete. You must be able to upgrade the capability of your devices, capabilities of your hardware and software, till the time an end customer has the product."

Selecting the right SoC

An IoT system needs to sense, communicate, perform inference and computation, while taking the security of the system and data into consideration. These things require tight integration of hardware-software architecture. Designers need to design compact products, running on minimum power, and utilising really low bandwidth while costing extremely low.

To achieve all of this the selection of an optimum system-on-chip (SoC) is very critical. SoC is one of the main components that play a crucial role in integrating all the components in an embedded system.

In addition to the discussed problems, the SoCs must last for years and, in many cases, last over a decade on a single coin-cell battery. So, very tight integration of all the aspects is necessary for a good IoT solution. To overcome the problem, engineers are trying to simplify inferencing, and to achieve that we can observe the development of Micro Edge.

Inference at Micro Edge

The inference is turning raw data from the sensors and other IoT devices into a piece of actionable information. In the inference we have seen a development of Micro Edge. Although sounding too fancy, Micro Edge is nothing but where your sensors and devices are residing.

According to Manish, "When you do inferencing right at the sensor or near the device, you are taking care of security, you are sending that inferred solution or the inference across as opposed to the live data. So, it gives you lower latency. It gives you a more secure solution, and obviously, it is cost-effective." He adds, "This is a growing use-case where inferencing is occurring right at what we call Micro Edge by the sensors and devices out there in the field."

Need for configurable system

The IoT applications are very diverse. From a light bulb to a health monitor, to machinery in a factory to an asset tracker, we have diverse applications and various micro verticals, with very different requirements for each one of them in every vector.

Therefore, designing a one-fit solution for all situations is not possible. But being a low-cost solution, it is also not feasible for a company to do R&D from ground zero for every single product. You must do quick turn derivatives in order to stay relevant.

> SOC SENSES AND CONTROLS, IT COMPUTES, IT INFERS, AND IT ALSO SAFELY HANDLES THE COMMUNICATION PART OF IT.

Furthermore, there are industrial and commercial customers with massive quantity requirements who want to have their own ecosystems and not of Google, Amazon, etc for privacy, security, and confidentiality. So, new players in the IoT segment have a wide variety of opportunities to tap in.

"We have diverse applications and various micro verticals, with different requirements for each one of them in every vector, whether it is sense requirements, compute requirements, etc. Your hardware-software needs to be configurable," adds Manish.

Finding the right opportunity

IoT has the potential to change how we do a lot of things. We are not just connecting things, or inferring things because it's cool, but because it has the potential to change the world. Apart from enhancing convenience and improving the lifestyle, Manish also feels that it will transform a variety of industries; it will transform even those industries that are considered low-tech, such as the construction industry.

Manish gives an example of how IoT can become beneficial for the construction industry, which does not usually require automation. He claims that the maximum amount of time and resources are not efficiently managed in the construction industry, and deadlines often get extended due to multiple reasons, such as improper management of inventory, work getting delayed, or resources being wasted due to bad weather or improper working conditions.

These are the areas where IoT can streamline things. IoT can help get live data of these tools, machines, and equipment, the utilisation of the equipment and tools, and their efficiencies. All these parameters are very critical for delivering projects on time.

Manish also believes that there are other parameters, such as how and when you service and maintain your equipment, or the status of the secure access for some of these construction sites where there is a danger of operation, like in mining operations, where you have to take personal safety into account. So, how do you ensure secure access?

There are clear problems right in front of us that we need to identify and solve to improve lives, improve businesses, their efficiencies, and the money that they make. This is just one industry. Nobody's going to give us a lot of money to just look cool. And unless we start solving real problems using IoT, the technology will not take off.

What the future holds

"IoT will disrupt multiple industries in the coming decade and beyond."

Disruption occurs due to technologies that are generally cheaper and simpler to use. Initially, in terms of performance, these disruptive

	Phones →	Feature phones →	Smartphones →		
	1980-1990	1990-2000	2000-2010	2010-2020	2021+
Communicate	1G (Analog Cellular)	2G (Digital Cellular)	3G (Mobile Broadband) BT NFC Wi-Fi	4G (Native IP Networks)	5G
Communicate-Data	~1Mbps or less	~10Mbps or less	~100MBps or less	~1GBps or less	~10GBps or less
Compute	Single core ~ 1Ghz or less	Single core ~ 2Ghz or less	2 and 4 cores ~ 2Ghz+	4 and 8 cores ~ 3Ghz	8 cores+ ~ 3Ghz+
Sense			Camera Touch ALS/PS → Accelerometers	Fingerprint	
Gaming		Gaming →			
Location		GPS →			
Inference		Predictive text →		Facial and voice recognition	AR/VR
Secure			Arm TZ →		
Software		Symbian OS	Full web browsing, app store →		
Process nodes - (microns) - Moore's Law	3 or better	0.8 or better	0.1 or better	0.02 or better	0.01 or better
			Digital/RF separation		

Figure 1: Mobile phone evolution: Predictor for IoT evolution

technologies are only suitable for a small number of users. Over time, these technologies improve and become a necessity for most people.

This phenomenon has occurred time and again. One such example is of the smartphone, which is making cameras, audio players, PCs, handheld GPS, and many other things obsolete.

According to many industry stalwarts, IoT will disrupt multiple industries in the coming decades.

To predict the growth of IoT we may look at the evolution of smartphones. The timeline of phones is shown in Figure 1.

As can be seen, there was massive advancement with each successive decade. Alongside the features in the phone's hardware, the networking technology kept improving. From 1G in the 80s, which could transfer less than 10kbps, the growth in data transfer rate kept on improving by a factor of ten.

The inception of digital cellular technology in 1990s changed the scope of mobile phones. From just a device to make calls the mobile device had added functionalities, such as gaming and GPS, and in terms of inference we got a new feature of predictive text.

Then the next big development happened in the 2000s, when the data transfer rate jumped from 64kbps to over 1Mbps. The phones now had cameras, touchscreens, accelerometers, etc. This is when mobile phone technology really disrupted the market. The processor in smartphones began catching up with the basic PCs, the phone cameras affected the sale of low-level point-and-shoot cameras, and many different gadgets were affected due to these powerful smartphones.

Smartphones were no longer merely a tool for communication but became much more than that. We saw a very high level of integration of the various functions, including computing prowess, sense, and gaming—all of these started to disrupt several industries.

The advancement made in the first decade caused major adoption of smartphones, thus bringing a flood of manufacturers driving the cost lower and lower with each passing year. The rate of refinement and growth continued in the 2010s. The inception of 4G improved the internet speed and the processing power of mobiles was now comparable to baseline PCs.

The phone cameras started to outperform the basic cameras. And similar was the fate of music players, such as MP3 players and iPods. Smartphones were no more a novelty but a necessity.

We have just set foot in the 2020s, but by now smartphones pack enough power to compete with decent PCs, cameras, etc. Some of the latest models have higher RAM and processing power than most entry-level laptops. They offer much higher resolution than most cameras, and the 5G internet is capable of matching or even exceeding the data transfer rate of cabled internet available in many areas.

Smartphones are now secure enough to do banking transactions and, in many countries, are being used as the primary mode for everyday transactions. However, it is too early to discuss the advancements that are likely to be made by smartphones in this decade!

The future of IoT

IoT technology is in its second generation in terms of development. Compared to mobile phones, IoT is where the phones were in 1990s as feature phones. The IoT technology is being adopted by a few, but it is yet to become a necessity for most.

From a process node standpoint, in terms of our inferencing capability, in terms of our compute capability, we are nowhere close to gigahertz, right? We have the sub-gigahertz capability today. So, these are very early days for IoT. Very likely, as the Moore's Law progresses, the sense of integration will also progress, and there will be dislocations that vendors and players will benefit from.

We normally think of Moore's Law as digital integration, and we can say that the IoT is at the beginning of Moore's Law journey. It could possibly have a high level of function integration driven by Moore's Law. When we talk about sensors, there is a variety of analogue front-ends on the sensors that can get integrated. So, there's going to be a high degree of integration.

It is again similar to the case of smartphones, with a lot of constraints. Compared to the technologies smartphones were battling - which

included PCs, servers, camera, etc - the smartphone had constraints of power, space, memory, and budget. So, it is safe to say that the constraints of the IoT are similar.

We can expect a high level of integration that will come in next few decades. Some other expectations are:

Cloudless architecture to overcome design constraints: Cloudless technology will enhance various domains of the IoT. It will provide greater choice and agility while reducing cost for applications that incorporate state-of-the-art hardware, software, and data access. You can think of a variety of cloudless architectures, like mesh networks that offer better latency and privacy. In fact, there are some companies already thinking about piloting peer-to-peer cloudless architectures. Therefore, privacy and reduced latency will be the biggest reason for the adoption of cloudless technology.

Self-organising network: A self-organising network (SON) uses artificial intelligence (AI), predictive analytics, and pre-optimised software algorithms to offer automated functions, such as self-configuration, self-optimisation, self-healing, and self-protection. SONs simplify network administration by enabling the creation of a plug-and-play environment for a network task. There are over fifty different wireless protocols that serve a variety of different needs. Bluetooth has a different role than Wi-Fi and so does the sub-gigahertz protocol that gives you a range of a few kilometres.

The benefit of implementing SON is that it will allow us to change the protocols on the fly. For example, the transmitter radios can automatically adjust parameters such as transmission power to avoid interference and maximise both coverage and capacity. It also provides better security as the self-organising network automatically defends itself from penetration by unauthorised users.

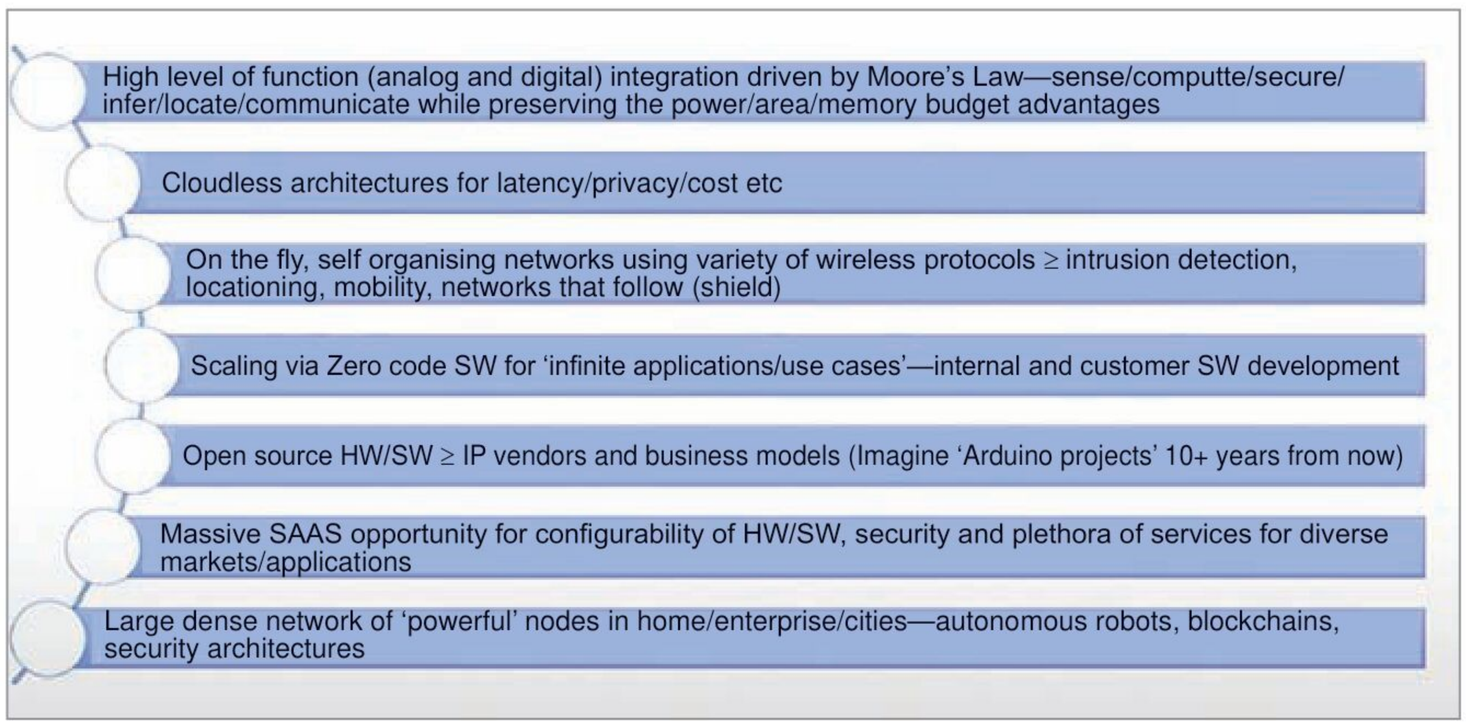


Figure 2: IoT future (Decade+) possibilities: Infinite and open to anyone's imagination

Open source software: The other very interesting thing that is bound to happen in software is scaling. In a world with infinite applications and use-cases, it is not possible to serve each individual customer's requirements. This is where a zero-code solution holds the potential to change the world. The zero-code solution will allow the customers to take the code, drag and drop connectors in a visual platform—and the code is outputted and the product is ready to go.

Growth with open source hardware: Open source hardware (OSH) is free-to-use physical artefacts of technology that can be made and used by anyone without any copyright infringement. There is a big open source community supported by electronics makers and DIYers all around the globe. One such example is of Arduino projects for microcontrollers and NodeMCU, which is an open source IoT platform. OSH is bound to grow and result in a variety of different business models that will disrupt the industries. Furthermore, it will empower individuals to create IoT solutions, thus democratising IoT.

Massive SAAS opportunity: The need for configurability of hardware and software creates a huge opportunity for software as a service (SaaS). Since there is a diverse market with specific requirements, IoT needs to meet a plethora of different demands, such as security and time relevance. So, there is an opportunity for a large dense network of powerful nodes. For example, every lighting pole present in the smart grid will have some sort of SoC that can self-organise and help you with location, sensing, etc. You can also think about running blockchain on it.

Although it seems far-fetched, we are talking about these in the future because we will have powerful computing power in these nodes. We are thinking about the 5G equivalent of IoT, which will have 8, 10, 12 cores offering significant compute power. Memory is only going to grow in all of these massive networks that would be available in the future.

These are all predictions, but what is certain is that the IoT holds the ability to disrupt a lot of industries and change the world for greater good. END

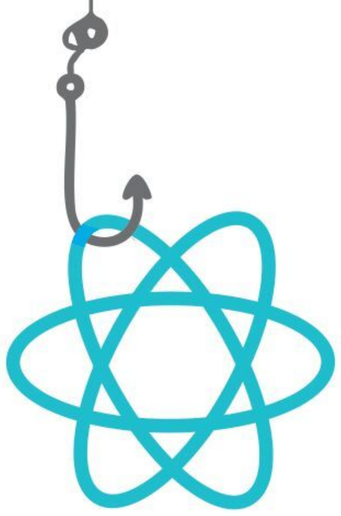
By: Manish Kothari

The author is senior VP of software development at Silicon Labs. This article is based on the keynote address he gave at the VLSI Design Conference 2022.

Transcribed and curated by Sharad Bhowmick

The article was originally published in the September 2022 issue of Electronics For You.

Developers Let's Try



Hooked on React Hooks!

In this final part of the two-part in-depth series on React Hooks, we continue to discuss a few more hooks and custom hooks.

In the first article in the series, which was carried in the October 2022 issue of *OSFY*, we learnt what hooks and custom hooks are, and discussed a few hooks briefly. We will look at a few more hooks and custom hooks in this second and final article on React Hooks.

useImperativeHandle

```
useImperativeHandle(ref, createHandle, [deps]);

function Button (props, ref) {
    const buttonRef = useRef();
    useImperativeHandle(ref, () => ({
        focus: () => {
            console.log (“Inside focus”);
            buttonRef.current.focus();
        }
    }));
    return (
        <button ref={buttonRef} {...props}>
            Button
        </button>
    );
}
export default forwardRef (Button);
```

Usually, when you use *useRef*, you are given the instance value of the component the ref is attached to. This allows you to interact with the DOM element directly. *useImperativeHandle* is very similar but it lets you do two things -- it gives you control over the value that is returned; so instead of returning the instance element, you explicitly state what the return value will be. Also, it allows you to replace native functions such as *blur* and *focus* with functions of your own, thus allowing side effects to the normal behaviour or different behaviour altogether. And the best point is you can call the function whatever you like. So, basically, *useImperativeHandle* customises the instance value that is exposed to parent components when using ref.

In the example above, the value we get from ref will only contain the function *focus*, which we declared in *useImperativeHandle*. Also, you can customise the function to behave in a different way than you would normally expect. So, for example, in the *focus* function, we can update the document title or log as seen in the code but notice that we have used *forwardRef*. In React, *forwardRef* is a method that allows parent components to pass down *ref* to their children, and it gives a child component a reference to a DOM element, which is created by its parent component.

So, basically, *useImperativeHandle* is used when you might not want to expose native properties to the parent or (as in our case) want to change the behaviour of a native function. As in the case of additional hooks, you'll rarely use *useImperativeHandle* in your code base, but it's good to know how to use it.

useLayoutEffect

```
const ref = React.useRef()
React.useEffect (() => {
    ref.current = ‘Lorem ipsum’
})

React.useLayoutEffect (() => {
    console.log(ref.current) // This runs first, will log old
value
})
```

As we saw with the *useEffect*, the code inside of *useLayoutEffect* is fired after the layout is rendered. Although this makes it suitable for, let's say, setting up subscriptions and event handlers, not all effects can be deferred. So, basically, we should understand the difference between passive and active event listeners. Let's say we want to make changes to the DOM synchronously before we render the additional components. If we use *useEffect*, this can lead to inconsistencies in the UI. As a result, for these types of events, React provides us with *useLayoutEffect*. It has the same signature as *useEffect* but the only difference is when it is fired. Your code runs immediately after the DOM has been updated but before the browser has had a chance to paint those changes.

For example, if you want to make DOM measurements like getting the scroll position or other styles of an element, and then make mutations or trigger synchronous re-render, *useLayoutEffect* is our guy. If you don't need to do all this, you should use *useEffect*. I'll go as far as to say that 99 per cent of the time, you would be using *useEffect*.

Custom hooks

You may have heard a lot about custom hooks. Let us try and understand what they are exactly. Custom hooks are normal JavaScript functions that can use other hooks inside them, and contain a common stateful logic that can be reused within multiple components. These functions are prefixed with the word 'use'. You can name them anything as long as they have the 'use' prefix. Let's say you have two functions or components which implement some common logic. You can create a third function with this common logic, and implement it in the other two functions. After all, hooks are just functions. They give us the advantage of fewer keystrokes and less repetitive code. I will show how this happens and we'll create our very own hook now.

Let's say we have a *count* state variable and we want to store it in the local storage whenever the value updates. *useEffect* is the perfect candidate here.

```
function App() {
    const [count, setCount] = useState(() => {
        return JSON.parse(
            window.localStorage.getItem('counter-value') ||
'0'
        );
    });
    useEffect (() => {
        window.localStorage.setItem('counter-value', count);
    }, [count]);

    return (
        <div>
            <button onClick={() => {
                setCount(count + 1)
            }}>{count}</button>
        </div>
    );
}
```

But what if we want to update this value in ten other components? The highlighted code will be repeated in all these 10 components. To avoid this repetition, we will use a custom hook.

```
function useLocalStorage(key, default) {
    const [state, setState] = useState(() => {
        return JSON.parse(
            window.localStorage.getItem(key) ||
String(default)
        );
    });
    useEffect (() => {
        window.localStorage.setItem(key, state);
    }, [state]);
}

function App() {
    const [count, setCount] = useLocalStorage('counter-
value', 0);
    return (
        <div>
            <button onClick={() => {
                setCount(count + 1)
            }}>{count}</button>
        </div>
    );
}
```

As you see in the above code, we have created our very own custom hook called *useLocalStorage* and added the repeated code to this hook. Now, in all our 10 components we can simply call the *useLocalStorage* hook, and pass in the key and the default value. It's as simple as that and will work in the exact same way as before.

With the introduction of custom hooks, people started creating hooks for the most common use cases. A simple search for a local storage hook in npm gave 60 already existing packages for various use cases. So whenever you find the use case for a custom hook in your React application, I recommend going through this list; maybe someone has already created a package that you can reuse and save development time.

This is one of the strengths of having such a vast and active community and React has that -- you can surely use it to your advantage.

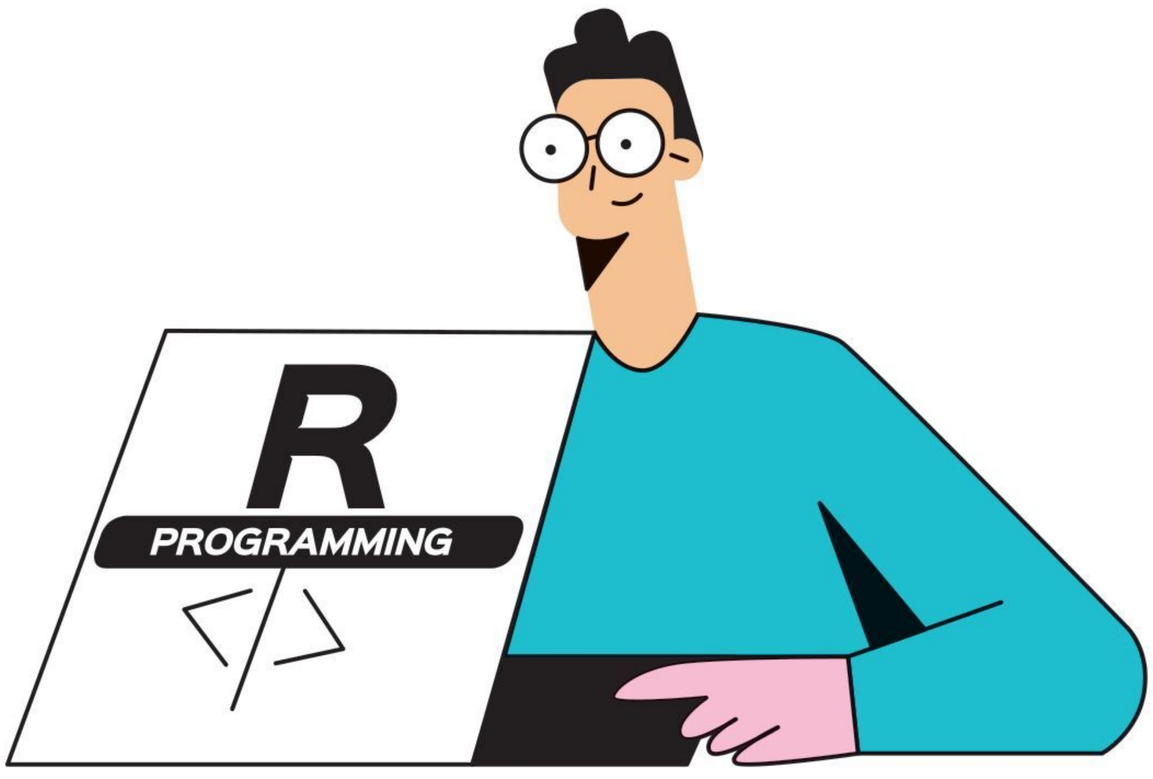
useDebugValue

```
function useLocalStorage (key, default) {
    const [count, setCount] = useState(0);

    // ...

    // Show a label in DevTools next to this Hook
    // e.g. "LocalStorage: Empty" useDebugValue (count == 0 ?
'Empty': 'Not Empty');
    useDebugValue(count == 0 ? 'Empty' : 'Not Empty');

    return count;
}
```

Continued to page...70

R Series: Probability Distributions

This fifteenth article in the R series will explore more probability distributions and introduce statistical tests.



We will use R version 4.2.1 installed on Parabola GNU/Linux-libre (x86-64) for the code snippets.

```
$ R --version

R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under the terms of the
GNU General Public License versions 2 or 3.
For more information about these matters see
https://www.gnu.org/licenses/.
```

Welch two sample t-test

Consider the *mtcars* data set in the *lattice* library:

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1 4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1   4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0  3  1
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02  0  0  3  2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

The displacement values are shown below:

```
> mtcars$disp
 [1] 160.0 160.0 108.0 258.0 360.0 225.0 360.0 146.7 140.8
167.6 167.6 275.8
[13] 275.8 275.8 472.0 460.0 440.0  78.7  75.7  71.1 120.1
318.0 304.0 350.0
[25] 400.0  79.0 120.3  95.1 351.0 145.0 301.0 121.0
```

We can prepare two data sets, P and Q, from the displacement data, as follows:

```
> P <- mtcars$disp[0:9]
> P
[1] 160.0 160.0 108.0 258.0 360.0 225.0 360.0 146.7 140.8

> Q <- mtcars$disp[10:19]
> Q
[1] 167.6 167.6 275.8 275.8 275.8 472.0 460.0 440.0 78.7 75.7
```

The boxplot for the data sets is shown in Figure 1:

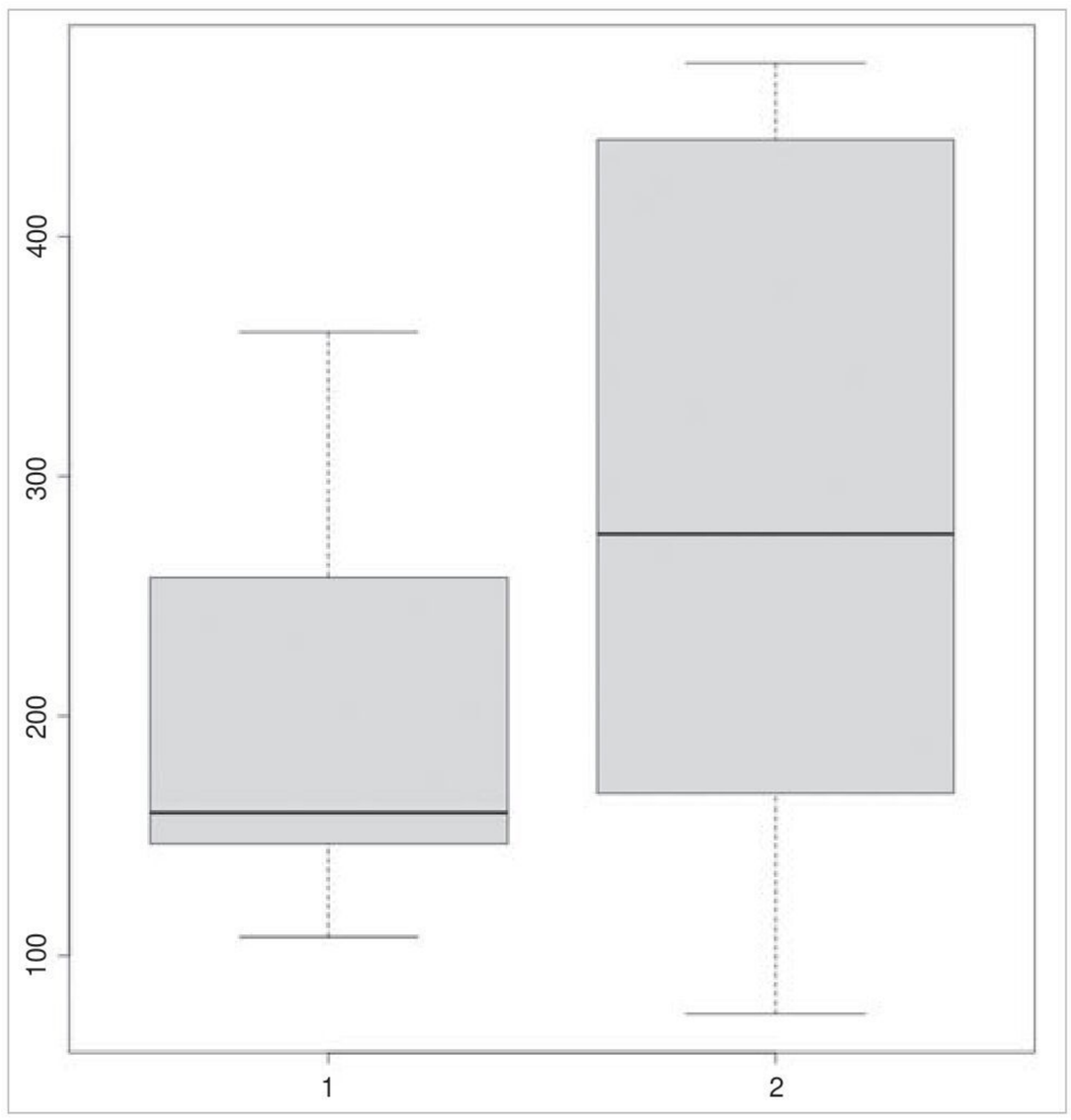


Figure 1: Boxplot *mtcars$disp*

```
> boxplot(P, Q)
```

The Welch two sample t-test can be performed using the *t.test()* function, as demonstrated below:

```
> t.test(P, Q)

     Welch Two Sample t-test

data:  P and Q
t = -0.98055, df = 15.369, p-value = 0.342
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -176.63004   65.16337
sample estimates:
mean of x mean of y
 213.1667  268.9000
```

We can also run the t-test assuming equal variances by passing *TRUE* to the 'var.equal' argument, as follows:

```
> t.test(P, Q, var.equal=TRUE)

     Two Sample t-test

data:  P and Q
t = -0.95746, df = 17, p-value = 0.3518
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -178.54460   67.07793
sample estimates:
mean of x mean of y

 213.1667 268.9000
```

Two sample Wilcoxon test

The Mann-Whitney-Wilcoxon test is used for randomly selected values from two populations. For our example, we can run the *wilcox.test()* function on the P and Q data sets, as follows:

```
> wilcox.test(P, Q)

     Wilcoxon rank sum test with continuity correction

data:  P and Q
W = 32, p-value = 0.3059
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(P, Q) : cannot compute exact p-value
with ties
```

The empirical CDFs for the data sets can be drawn using the *plot()* function:

```
> plot (ecdf(P), verticals=TRUE)
> plot (ecdf(Q), verticals=TRUE)
```

The *wilcox.test()* function accepts the following arguments:

Arguments	Description
x	A numeric vector of data values
y	Optional numeric vector of data values
alternative	Character string for alternative hypothesis ('two.sided', 'greater', or 'less')
conf.int	A logical value that indicates if confidence interval should be computed
conf.level	Confidence level of the interval
correct	A logical value on whether to apply continuity correction
data	An optional matrix or data frame
exact	A logical value on whether p-value should be computed
mu	Optional parameter for the null hypothesis
na.action	A function to handle data that contains NAs
paired	A logical value to indicate a paired test
subset	An optional vector specifying a subset of observations
tol.root	A positive numeric tolerance

Kolmogrov-Smirnov test

The Kolmogrov-Smirnov test or KS test is used to test if two samples come from the same distribution. It can also be modified to be used as a 'goodness of fit' test. The two-sample Kolmogrov-Smirnov test for the P and Q data sets is as follows:

```
> ks.test(P, Q)
```

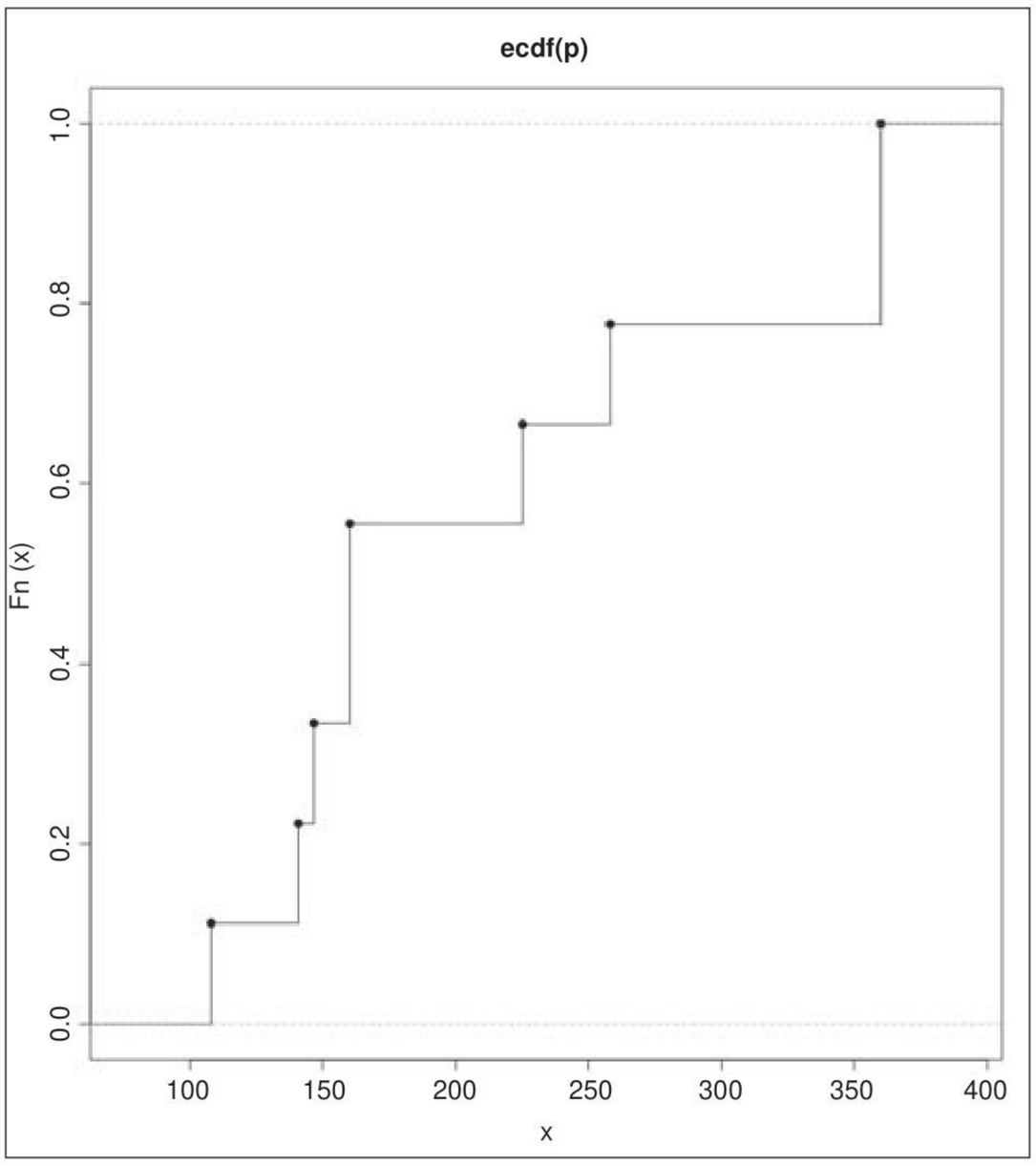


Figure 2: Empirical CDF (P)

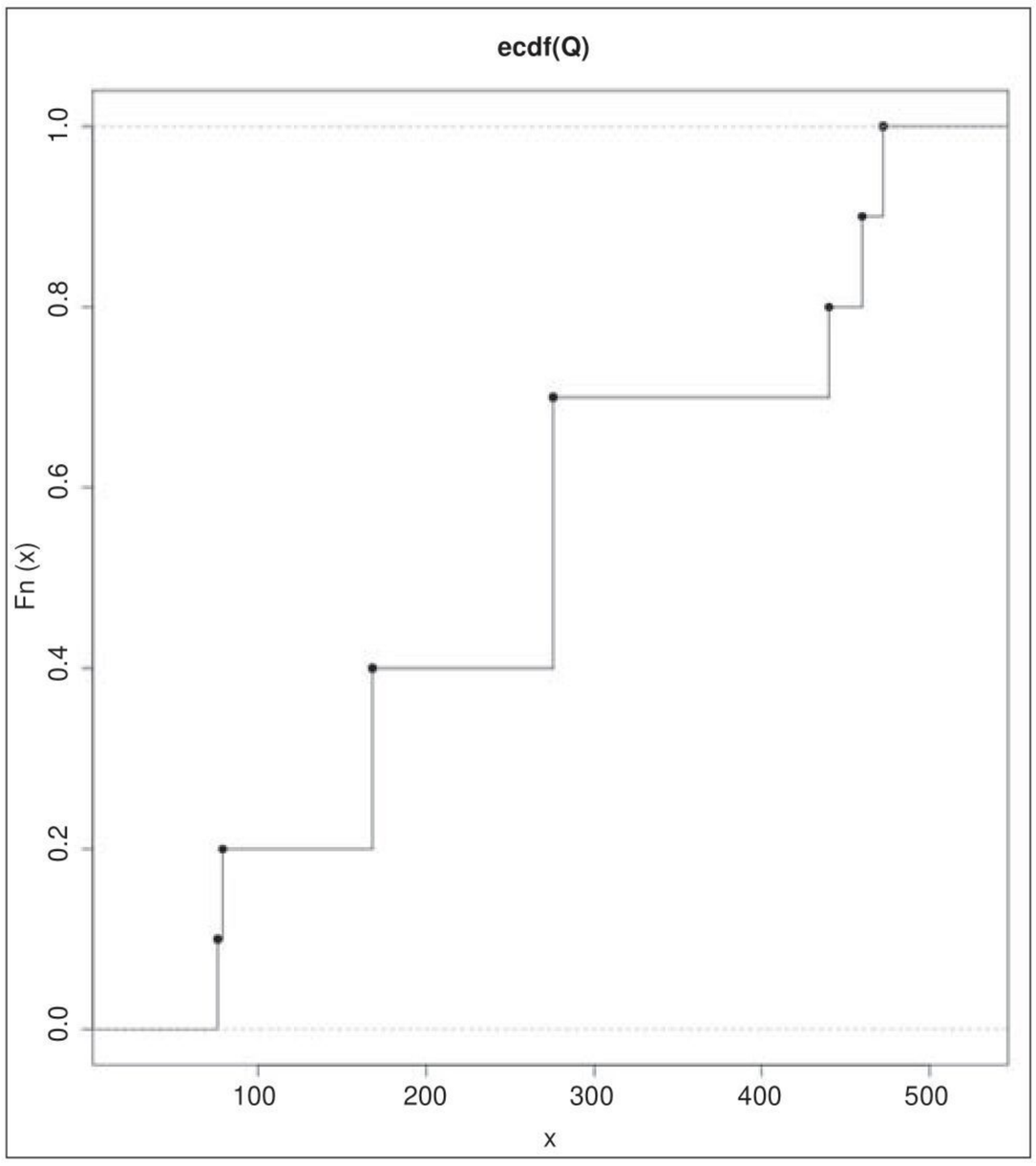


Figure 3: Empirical CDF (Q)

```
        Exact two-sample Kolmogorov-Smirnov test

data:  P and Q
D = 0.37778, p-value = 0.2634
alternative hypothesis: two-sided
```

You can also perform a one-sample Kolmogrov-Smirnov test. The function accepts the following arguments:

Arguments	Description
x	Numeric vector of data values
y	Numeric vector or character string name of a CDF
alternative	Alternate hypothesis should be 'two-sided', 'less' or 'greater'
exact	NULL or logical value to specify if p-value should be computed
simulate.p.value	A logical value to compute p-values by Monte Carlo simulation
data	An optional data frame
subset	A subset vector of data to be used
na.action	A function to handle data that contains NAs

The alternative hypothesis argument accepts the string argument values: 'two.sided', 'less', and 'greater'. These indicate if the null hypothesis of the distribution of 'x' is equal to, less than, or greater than the distribution of 'y'. Any missing values are ignored from both the samples.

Chi-Square test

The Chi-Square test is a statistical hypothesis test to determine the correlation between two categorical variables. We can determine if the cylinder and horse power variables from the *mtcars* data set are significantly correlated.

```
> data <- table(mtcars$cyl, mtcars$hp)
> print(data)
```

```
   52 62 65 66 91 93 95 97 105 109 110 113 123 150 175
180 205 215 230 245 264
  4  1  1  1  2  1  1  1  1   0   1   0   1   0   0   0
0   0   0   0   0   0
  6  0  0  0  0  0  0  0  0   1   0   3   0   2   0   1
0   0   0   0   0   0
  8  0  0  0  0  0  0  0  0   0   0   0   0   0   2   2
3   1   1   1   2   1
```

The *chisq.test()* function can be performed on the data variables, as shown below:

```
> chisq.test(data)

        Pearson's Chi-squared test

data:  data
X-squared = 59.429, df = 42, p-value = 0.0393
```

A p-value of less than 0.05 shows that there is a strong correlation between cylinder and horsepower.

The *chisq.test* function accepts the following arguments:

Arguments	Description
x	A numeric vector or matrix
y	A numeric vector, or ignored by x is a matrix
correct	A logical value on whether to apply continuity correction
p	A vector of probabilities
rescale.p	A logical scalar to sum to 1
simulate.p.value	A logical value to whether compute p-values by Monte Carlo simulation

Beta

The Beta function is mathematically defined by the integral for two inputs, alpha and beta, as shown below:

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt$$

Figure 4: Beta distribution

You can use the *beta()* function for the P and Q data sets, as follows:

```
> beta(P, Q)
 [1] 7.307338e-100 7.307338e-100 2.515827e-100 5.970038e-162
2.168131e-190
 [6] 8.246568e-192 1.139582e-245 1.211659e-144  2.186038e-63
1.931142e-65
```

The *lbeta()* function provides the natural logarithm of the beta function, as demonstrated below:

```
> lbeta(P, Q)
 [1] -228.2696 -228.2696 -229.3359 -371.2320 -436.7173
-439.9865 -564.0027
 [8] -331.3803 -144.2808 -149.0099
```

The *dbeta()* function provides the density distribution. An example is given below with a plot (Figure 5).

```
> p = seq(0, 5, length=50)
> plot(p, dbeta(p, 3, 5), type='l')
```

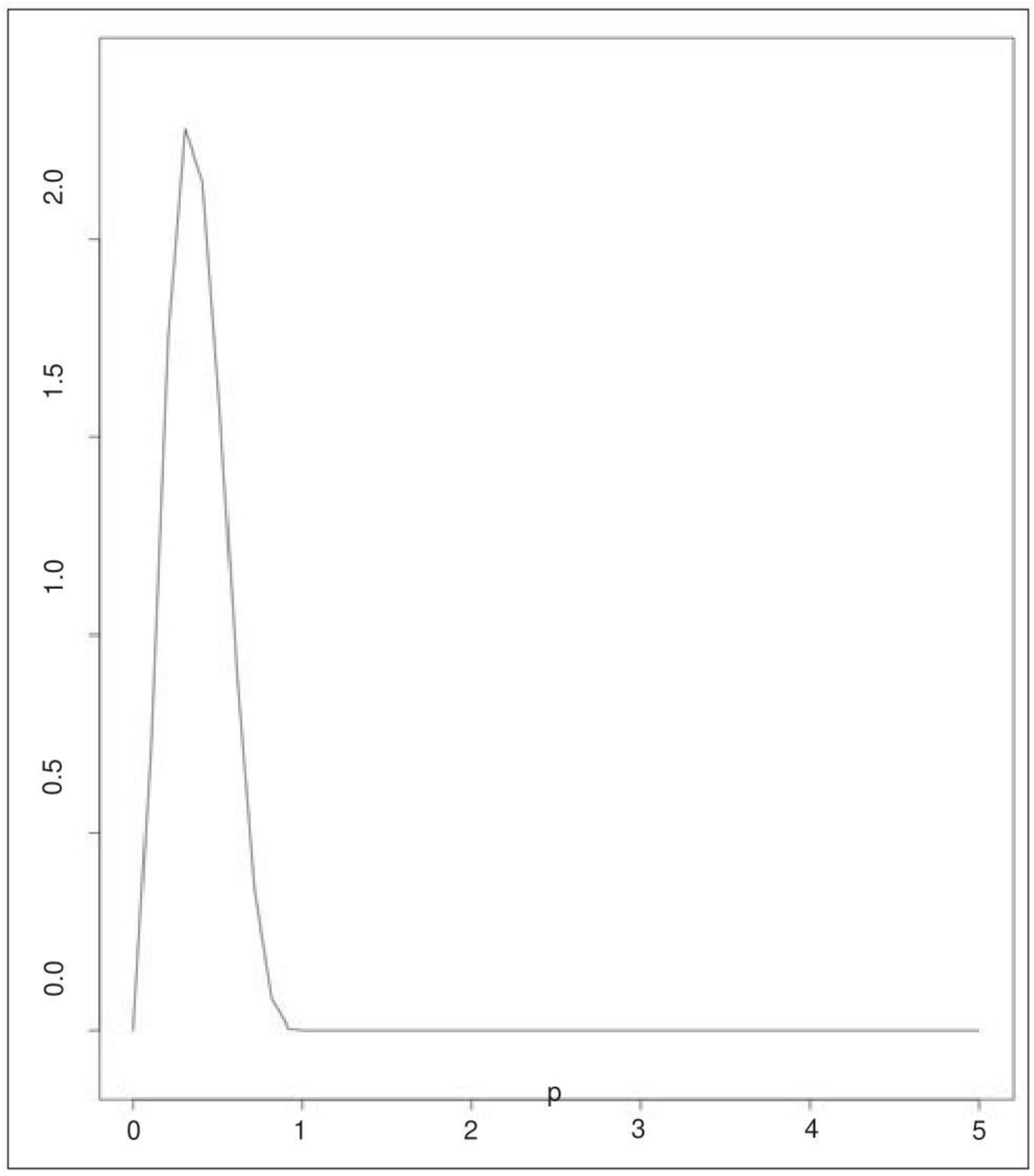


Figure 5: DBeta distribution

Gamma

The Gamma distribution is a continuous probability distribution that has a shape (z) and scale parameter (k), which is defined as shown in Figure 6.

$$\Gamma(\alpha) = \lambda^{\alpha}\int_0^{\infty} y^{\alpha-1}e^{-\lambda y}dy$$

Figure 6: Gamma distribution

The *dgamma()* function provides the density function. In the following example, a sequence of values is produced and a plot for its gamma distribution is generated.

```
> a <- seq(0, 50, by=0.01)
> y <- dgamma(a, shape=3)
> plot(y)
```

The *lgamma()* function provides the natural logarithm of the absolute value of the gamma function. For example:

```
> l <- lgamma(a)
> head(l)
[1]      Inf 4.599480 3.900805 3.489971 3.197078 2.968879
```

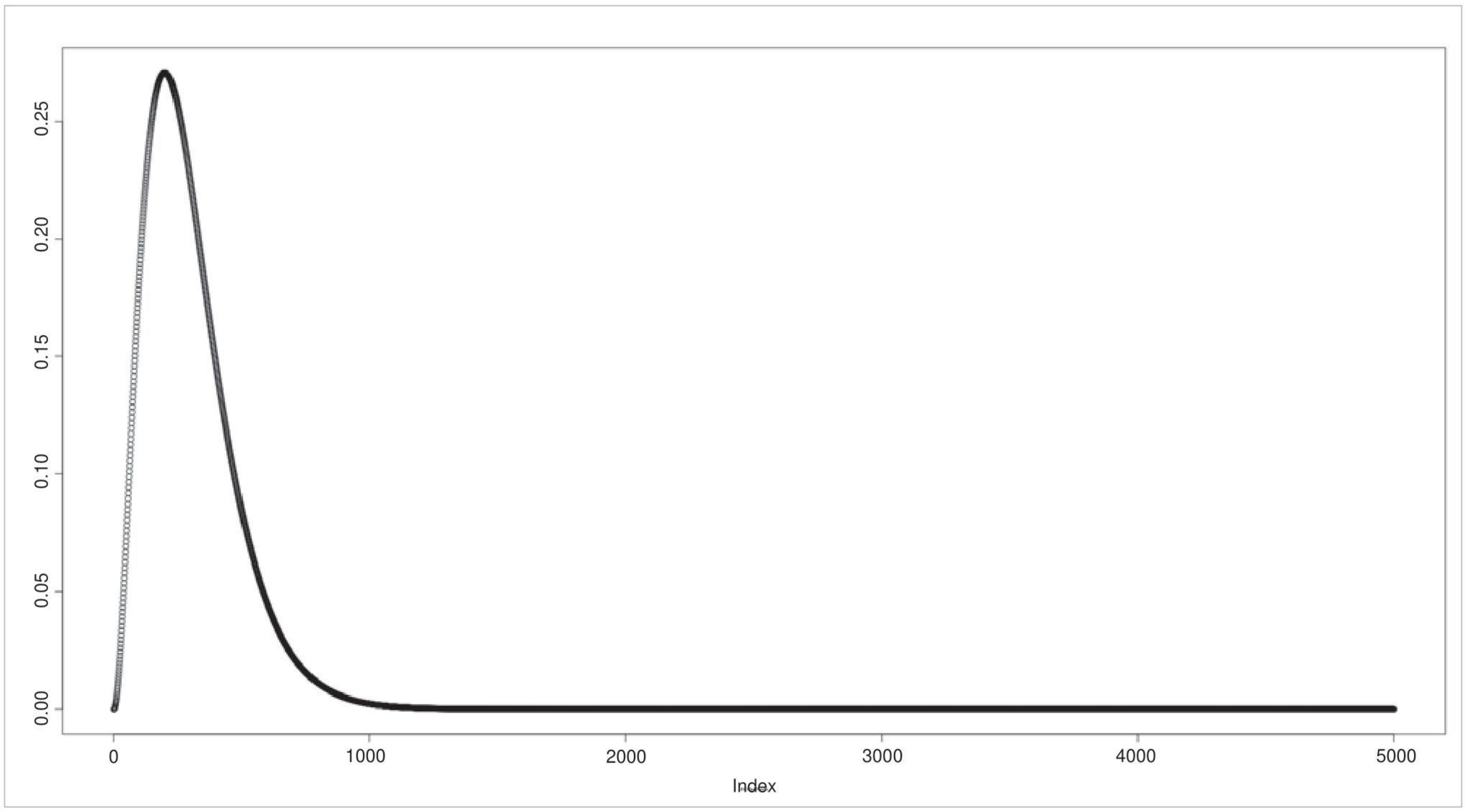


Figure 7: Gamma function

These functions accept the following arguments:

Arguments	Description
a, b	Positive numeric vectors
x, n	Numeric vectors
k, d	Integer vectors

Do read the manual pages for the above R functions to learn more on their arguments, options and usage. END

By: Shakthi Kannan

The author is a free software enthusiast.

Continued from page...65

As you may know, React dev tools are an amazing way to debug apps. React shows a component tree with props values while running the app. *useDebugValue* is used to display labels for custom hooks in the React dev tools. For example, if you look at a custom hook in the code above, we can display the state value of our custom book by using *useDebugValue*, which will show us the label with the key as the hook name (it will just remove the 'use' word; so it says local storage here instead of 'useLocalStorage') and the value will be dependent on the state value.

We can pass an optional function to this hook solely for formatting purposes, i.e., for data to look good when we inspect the components. Keep in mind that this is only for debugging and it won't be seen on the UI. This is just made to make the lives of developers easy when they inspect elements. *useDebugValue* takes in the debug value and returns the formatted display just for the developers to see the data. So the data looks pretty and is easier to understand.

Some rules

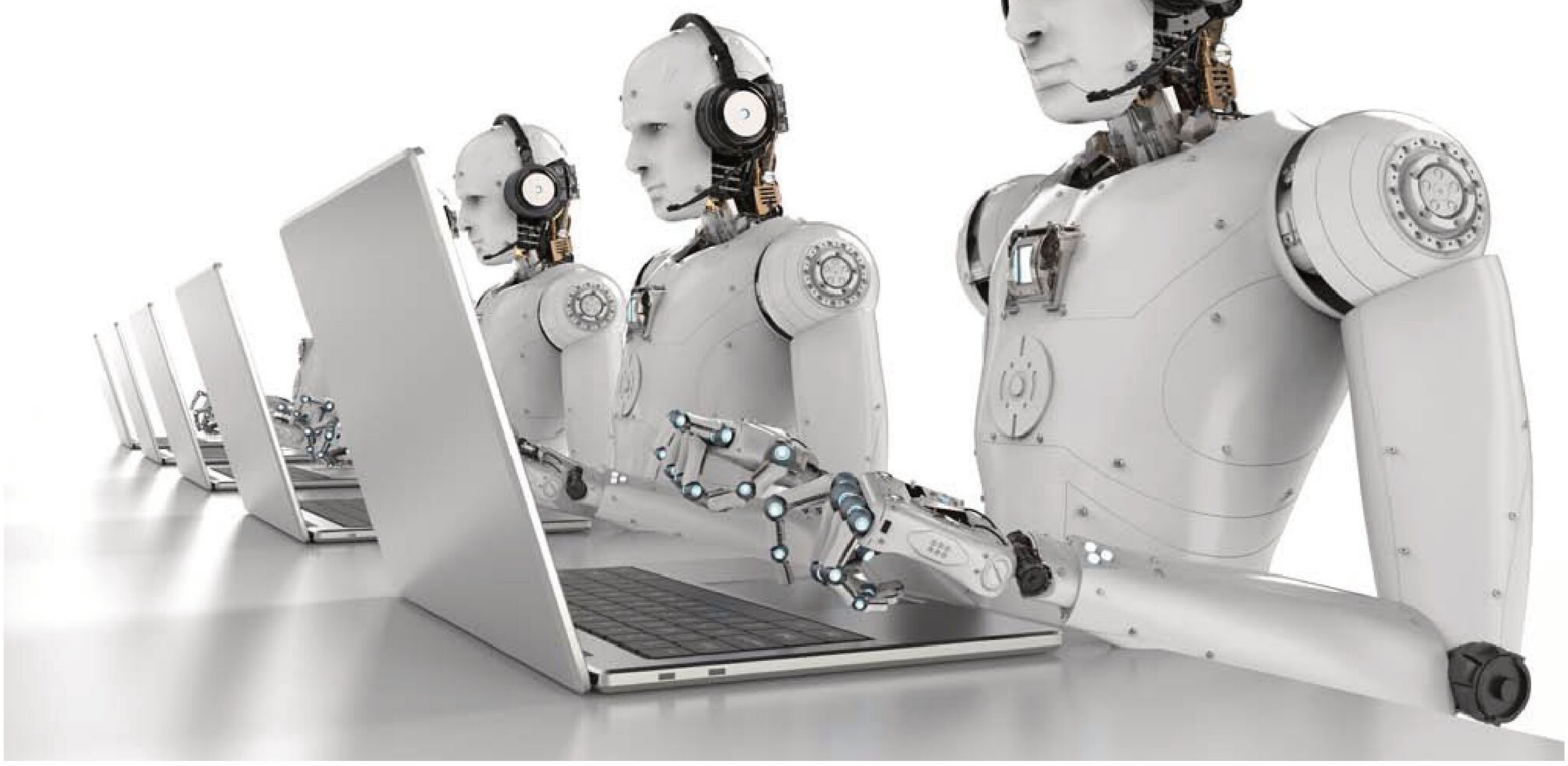
Now that you know all about hooks, you should know two additional rules that are imposed by them. First, we have to call hooks only at the top level. That is, don't call hooks inside loops, conditions, or nested functions. Second, call hooks only from React function components, and not from regular JavaScript functions. There is just one other valid place to call hooks, and that is your own custom hook as we saw in the custom hook section. Basically, follow these two rules and everything will work as expected for you. END

By: Janhavi Dahihande

The author is a software engineer based in San Francisco. She is interested in learning and talking about new technologies, and likes contributing to open source. When not coding, she likes to attend book club events, travel, and try out new cuisines.

How Contact Centres can Benefit from Artificial Intelligence

Classifying text and processing it for inference is very important in artificial intelligence based solutions. This article is about how artificial intelligence can be used in the contact centre and the benefits it brings.



Contact centre automation creates hassle-free customer support and helpdesk management. The application of artificial intelligence (AI) in a contact centre is still new and evolving. However, it can reduce operational costs, personalise the customer experience, increase agent efficiency, and provide more actionable analysis in the following ways.

Chatbots: Chatbots may be the most visible use of artificial intelligence (AI) in the customer service process. When customers choose to chat online with a business, chatbots greet them, collect some background information, and try to solve their issues. AI chatbots are good at solving very simple problems, but the more complicated issues still need an agent's touch. The AI chatbot will pass along the information it collected to make the transition as seamless as possible and to boost agent efficiency.

More intelligent routing: AI makes the routing of the ACD (automatic call distributor) even smarter. Now contact centres can route enquiries based on additional criteria, such as customer personality and information gleaned from previous contacts. Matching a customer with the right agent at the right time can do a lot to improve the customer experience.

Insightful analysis: AI can also elevate analysis and provide business leaders with information they can act upon. For example, using AI powered analysis, businesses can analyse customer behaviour, identify ones who are at risk of churning, and reach out to them with a compelling, personalised offer.

AI can bring down costs in the contact centre due to increased self-service. It also helps to improve the customer experience, as customers can

communicate using their channel of choice any hour of the day or night.

However, organisations must realise that the role of a contact centre agent will become more challenging with the introduction of AI. Automation remains most effective with interactions that involve simpler tasks and that do not pull in a lot of different pieces. Here, the human touch is not that critical, and AI can provide straightforward information to customers overnight, during holidays and other times when it can be costly to staff a contact centre. This means that agents will receive a larger percentage of calls that are more complicated and challenging.

When customers need information about a specific problem, recovery from a mistake or a more detailed solution that is too complicated for automation, there is a significant need for a human agent. Here, human interaction is necessary to help relieve frustrations and provide correct responses with the appropriate empathy and emotions (which, at times, is more difficult than just navigating a system).

How AI can be used in a contact centre

Use of AI to assist agents: While an agent listens to a customer during a phone call, AI can run in parallel and do some of the heavy lifting of searching a knowledge base for answers while allowing the agent to focus on the customer interaction. Once the information is presented to the agents, they can review the results in real-time, decide what will work best in each situation, and communicate the response to the customer.

AI also works better if information is coming in from the customer via the written word and/or images in the form of an email or chat session. This removes accents, translations, and other questionable input so that automated processes can easily understand the data and then respond to those interactions. In either case, once AI is providing accurate information in a consistent manner, the technology can interact directly with customers via a bot.

Use of AI to run analytics in the background: First, predictive contact routing analyses key pieces of information such as whether the caller is a high spender, what kind of products they buy, their sales history, projected future behaviour, and other key factors to route calls to the agent who has the best set of skills to successfully work with the customer.

Second, contact centres can use speech analytics technology to analyse recorded calls with customers (post-call analytics). After listening to a call, AI can identify when a voice inflection goes above a specific threshold level, and discover when and why customers get angry and potentially churn.

Speech analytics technology can also be used on a real-time basis to analyse the tone and inflections of a caller to indicate when they appear frustrated, show anger, and need intervention from a higher skilled agent/supervisor. This is a more expensive use of AI in contact centres, but can provide an opportunity to avoid a negative interaction, making customers happier and increasing company loyalty.

There will always be a percentage of calls that will be fully automated using a form of AI. If an automated system works well, customers may even reach out more frequently to get their answers quickly and easily. This will reduce the time spent waiting in a queue, eliminate the time involved for a human response, and drive down overall costs. However, on the flip side, agents will need to be better trained and require higher skills to deal with the more complicated and sensitive calls that can't be handled through automation, which will offset some of the savings.

Businesses with large budgets can test various AI processes to see what works and what doesn't. But smaller contact centres really need to look at the time and costs involved before implementing any kind of AI system. To take this step, it is important to ensure that effective training and coaching, workforce management, and other technologies are already in place. Customer satisfaction should be a top priority.

A number of AI services for contact centres are available in the market. Let's take a look at some of the best ones.

Amazon Connect

AWS provides Amazon Connect as a fully managed cloud solution for contact centre automation.

Contact center AI is integrated with DialogFlow, which uses natural language processing (NLP) for voice based heuristic search for customer queries. It also helps to do contextual routing to the required support service, based on the query typed by the end user. A dashboard and analytics are on offer for query related metrics on a day-to-day basis based on ticket status like 'active', 'closed', 'waiting', to name a few. It is a hybrid 'build once and deploy anywhere' solution, and can work on smart devices including Alexa and Siri enabled devices.

AWS Connect helps to build a scalable, resilient IVR solution without any worry about infrastructure, deployment models and integration with other services for a public cloud environment. It is an omni-channel cloud contact centre service with unified machine learning (ML) capabilities to manage the customer and contact centre client (agent) experience seamlessly. It makes setting up a contact centre and chatbots very easy. This can be done without much technical experience as Amazon takes care of most of the technical configuration based on your inputs.

If you don't have much experience in contact centre development, you can make use of the Contact Center Intelligence (CCI) support facility in Amazon, which can help in designing a strategy for the centre. You can choose

any AWS CCI partner for setting up the contact centre, and integrate it with other AWS native services like data analytics, ML based data processing and data storage, to name a few.

Azure Cognitive Services

Microsoft's Azure Cognitive Services can help you realise partial or full automation of telephony based customer interactions, and provide accessibility across multiple channels. With its language and speech services, you can analyse call centre transcriptions, extract and redact personally identifiable information (PII), summarise the transcription, and detect the sentiments.

Azure Cognitive Services can be implemented in call and contact centres in various ways.

- **Virtual agents:** Conversational AI based telephony-integrated voice bots and voice-enabled chatbots.
- **Agent assistance:** Real-time transcription and analysis of a call to improve the customer experience by providing insights and suggesting actions to agents.
- **Post-call analytics:** Post-call analysis for creating insights into customer conversations to improve understanding and support continuous improvement of call handling.

Google Contact Center AI

Google Cloud Platform (GCP) provides loads of AI based solutions, which are integrated with Google Contact Center AI services for virtual assistance. GCP's Contact Center AI is a hybrid solution suitable for on-premises private as well as the public cloud environment.

Some of its important components, which are now in general availablility (GA), are listed below.

Dialogflow CX: This helps with handling complex customer queries and supplemental questions by looking for the required responses from a knowledge base. It supports human agents by providing the relevant details required to respond to a query. Dialogflow CX agents build on Google's natural language understanding (NLU) algorithms to handle these search capabilities.

Agent Assistant: This is built with AI assisted conversational services to prepare the recommended answer choices for human agents to choose from, for replying to customer queries. It uses a centralised knowledge base to prepare ready-to-use responses. It transcribes live calls in different languages and helps train AI models to analyse frequently asked questions, answer patterns and customer groups based on interest.

CCAI Insights: These services include text-to-speech and speech-to-text conversion, sentiment analysis, ranking of responses, integrated IVR services, and natural language processing capabilities to provide unified customer support powered by Google Cloud AI services.

Contact centre AI: The trends

Contact centres are undergoing a massive transformation. During the Covid-19 pandemic, call volumes increased but the number of customer service agents on duty fell. Calls increased by nearly 300 per cent during the first few months of the pandemic, resulting in longer hold times and more escalations to human agents or other support representatives.

With about 90 per cent of contact centres migrating to remote and hybrid workplace models since the beginning of the pandemic, companies are investing in new technologies that allow them to provide high quality and personalised customer service experiences.

In response to the changing landscape, many companies are turning to AI to power customer service operations. Listed below are six trends with respect to the use of AI in contact centres that promise to transform customer service and experience.

Alleviation of shortages in staff: Labour shortages were a side effect of the pandemic, and many industries found it difficult to manage high call volumes with low volumes of agents. Many businesses are turning to AI to account for the gaps within their workforces.

AI is helping to fill skills shortages of the existing workforce through career transition support tools. It is also helping employees do their jobs better and faster using digital assistants and in-house AI-driven training programmes.

Understanding the nuances of human conversations: AI solutions that are perceptive to caller intent, language, and sentiment are vital. Contact centre AI helps to understand the nuances of human conversation because it can pick up on the tone of voice, inflection, etc, to detect mood and modify behaviour accordingly. Understanding voice nuances and serving customers on that basis improves the overall experience.

Self-service: Self-service helps to deliver customer support quickly and efficiently since customers don't have to wait for a human agent. Companies continue to provide human backup, but prioritise AI support for their contact centres. One way businesses can use AI to deliver round-the-clock support is through chatbots and interactive voice response (IVR) systems.

Chatbots automate a personalised experience, connect meaningfully with

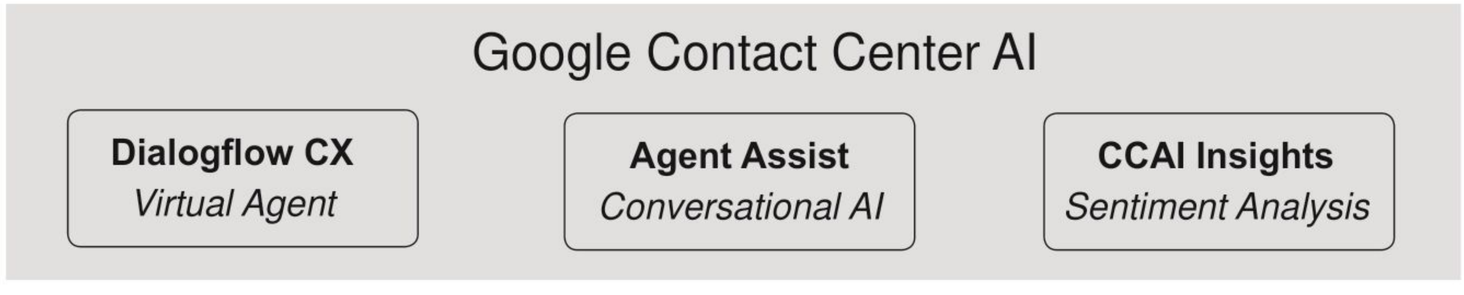


Figure 1: Google Contact Center AI

Amazon Transcribe Medical

In a crisis like Covid-19, remote medical consultation is a real need. Amazon Transcribe Medical is a machine learning based service, which can convert audio instructions given by doctors into text, and can interpret the technical medical terms used in the audio. It uses machine learning based automatic speech recognition (ASR) to capture the conversation between the doctor and the patient for making an electronic medical/health record (EHR). This helps doctors to attend to patients quickly.

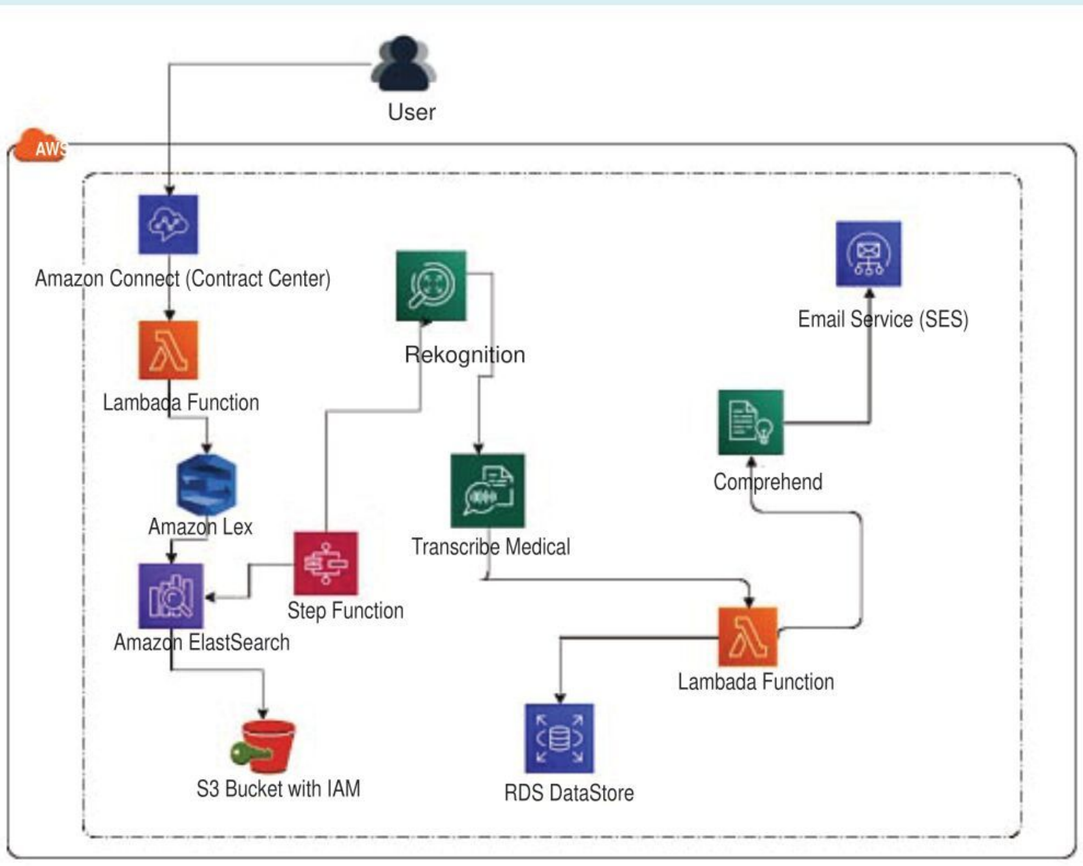


Figure 2: Architecture of Amazon Transcribe Medical based contact centre

As can be seen in Figure 2, a user connects to Amazon Connect for interactive chats, which is a virtual assistant service that works with a given knowledge base for answering queries (e.g., the remedy for a dry eye problem). Lex (chatbot service) is used to fetch the follow-up questions that the user is asked. This is done through a Lambda function invocation, which searches the knowledge base. If the required response is not found, it moves into the next level of search through Rekognition, which helps identify the face of the user, and connects him or her to their preferred doctor's advice. The advice for various medical problems is stored as an audio file and can be searched through Transcribe Medical service to get the solution for the given query. This is then given to Comprehend to convert as a response to the user in Amazon Connect or through email.

customers and deliver engaging content. IVRs are like 'virtual receptionists'. Callers interact with them through a series of menus, which direct them to the most appropriate help for their needs.

24/7 customer service: Contact centre AI enables customer service during every hour of the day, as it does not need to sleep or take breaks.

Companies can strategise staff shifts for 24/7 service, taking the weight off of individuals expected to be on call at all hours of the night. Utilising AI for 24-hour contact service helps companies to allow fewer irregular work hours for employees, while making sure customers receive help whenever they need.

Enabling cost-effective pricing: Like any investment in new technology, businesses are wary of contact centre AI pricing too. However, research shows that investing in quality AI for contact centres will save businesses money and increase the quality of service provided over time. Contact centre AI also helps generate revenue by allowing allocation of employee hours to other key business functions.

The competitive advantage: Since customers like to solve issues at any time and on their own, the quality of a contact centre's AI self-service adds hugely to the value of a contact centre team.

Contact centre AI is intuitive and accurate, and helps businesses deliver experiences that meet customers' demands. Contact center AI technology also helps mitigate stressors like long hold times and repetitive questions. Businesses that use AI for their contact centre achieve higher year-on-year revenue and profit margins. END

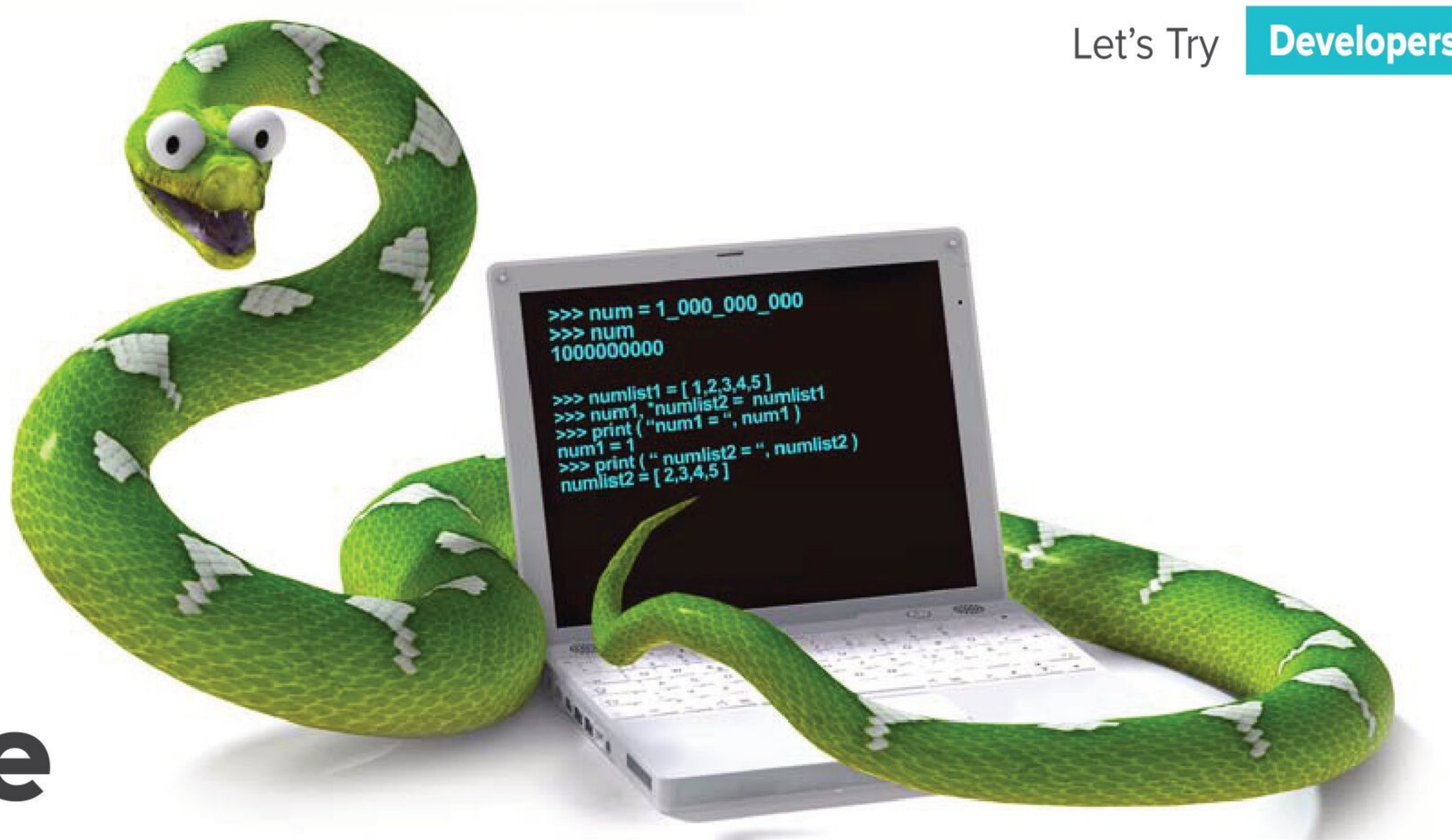
References

- *https://blog.hubspot.com/service/artificial-intelligence-call-center*
- *https://callcenterstudio.com/blog/advantages-of-artificial-intelligence-in-call-centers-blogs/*
- *https://www.vonage.com/resources/articles/artificial-intelligence-contact-center/*
- *https://www.destinationcrm.com/Articles/CRM-News/CRM-Featured-Articles/AWS-Expands-Its-AI-Powered-Contact-Center-Intelligence-(CCI)-144700.aspx*
- *https://azure.microsoft.com/en-in/services/cognitive-services*
- *https://cloud.google.com/solutions/contact-center*
- *https://cloud.google.com/blog/products/ai-machine-learning/discover-advanced-insights-with-google-cloud-natural-language-ai*
- *https://www.vonage.com/resources/articles/the-future-call-center-10-predictions-for-the-next-10-years-2/*

By: Dr Anand Nayyar and Dr Magesh Kasthuri

Dr Anand Nayyar is a PhD in wireless sensor networks and swarms intelligence. He works at Duy Tan University, Vietnam, and loves to explore open source technologies, IoT, cloud computing, deep learning and cyber security.

Dr Magesh Kasthuri is a senior distinguished member of the technical staff and principal consultant at Wipro Ltd. This article expresses his views and not that of Wipro.



Twelve Python Tips for Beginners

In a previous article published in the March 2021 edition of *Open Source For You* ('Python: A Few Useful Tips for Beginners'), the author of this article had written some simple but useful tips for those beginning to learn the Python language. He has now compiled twelve more tips that are interesting and useful for Python programmers. These tips will help write concise and easy-to-read code.

As per both the TIOBE index and the PYPL (PopularitY of Programming Language) site, as on September 2022, Python is the most popular programming language and has retained its popularity for almost three years now. This is because of its ease of programming and the availability of a very large number of libraries covering a large number of areas, right from business and finance to scientific computing and AI. It is also easy to learn compared to other languages like C/C++ and Java.

So let's begin with the Python tips that will help you write concise code easily.

Use of underscore to separate digits of long numbers

It is difficult to count the number of digits if a number, be it an integer or floating-point number, is very long. Python allows the use of underscores (_) as visual separators for grouping digits to make it easier to read the number. Hence, the number 1000000000 can be written as 1_000_000_000.

```
>>> num = 1_000_000_000
>>> num
1000000000
```

List unpacking and partial assignment

If we need to assign some of the elements of a list to particular variables and the remaining values to some other variable, the '*' notation for unpacking can be used in Python.

```
>>> numlist1 = [ 1,2,3,4,5 ]
>>> num1, *numlist2 =  numlist1
>>> print ( "num1 = ", num1 )
num1 = 1
>>> print ( " numlist2 = ", numlist2 )
numlist2 = [ 2,3,4,5 ]
```

Unpacking is used in the assignment statement below to assign values from the list to three variables simultaneously, and the '*' is used to assign all the remaining values.

```
>>> numlist1 = [ 1,2,3,4,5 ]
>>> num1, *numlist2, num3 =  numlist1
>>> print ("num1 = ", num1)
num1 =  1
>>> print ( "numlist2 = ", numlist2 )
numlist2 =  [ 2, 3, 4 ]
>>> print ( "num3 = ", num3 )
num3 =  5
```

Use of underscore in unpacking

The underscore (_) can also be used to ignore specific values while unpacking. The value that has to be ignored is assigned

to underscore (_).

```
>>> num1, _, num2 = [ 1,2,3 ]
>>> print ( num1 )
1
>>> print ( num2 )
3
```

The '*' symbol is used as a prefix to underscore (_) to unpack multiple values and assign them to a list.

```
>>> num1, *_, numlist2 = [ 1,2,3,4,5,6 ]
>>> print ( num1 )
1
>>> print ( numlist2 )
6
```

Getting a password from a user

An easy way to get a password as input from the user is to use the *getpass* function defined in the *getpass* module. The function displays a prompt and then prevents echo on the display when the user types the password.

```
!pip install getpass
from getpass import getpass
uname = input ( "Username: " )
password = getpass ( " Password: " )
```

List comprehension

List comprehension offers an easy way to create a new list in a single line, compared to the traditional methods. An example is given below. If we want to create a new list using the *for* loop, the code will be as follows:

```
#create a list containing the cube of the first 9 integers
>>> cubelist = [ ]
>>> for i in range ( 1,10 ):
>>>     cubelist.append ( i * i  * I )
>>> print ( cubelist )
[ 1, 8, 27, 64, 125, 216, 343, 512, 729 ]
```

The same can be written using list comprehension in the following way in a single line of code:

```
>>> cubelist = [ i * i * i  for i in range (1,10) ]
>>> print ( cubelist )
```

You can even use it for conditional logic and filtering.

```
>>> numlist1 = [ 35, 49, 45, 75, 90, 15, 65, 63, 84, 54 ]
>>> #create a new list whose elements are divisible by 7
>>> numlist2 = [ i for i in numlist1 if i % 7 == 0 ]
>>> print ( numlist2 )
[ 35, 49, 63, 84 ]
```

zip() function

The *zip()* function is an inbuilt function in Python that takes two or more iterable contents passed as arguments like list, dictionary, string or even iterables that are user defined, and combines them to create a single tuple, an iterator object. The *zip* function combines the elements with the same index in pairs (or more if the number of iterators is more) in a tuple. The first item of each argument iterator is combined, then the second is combined, and so on, until the length of the shorter iterable is covered, by default, to create a tuple object. Given below is a simple example:

```
>>> names = [ "Ganesh","Siva Sundar","Gunasekhar","Santosh" ]
>>> marks = [ 90, 88, 75, 72 ]
>>> marklist = tuple ( zip ( names, marks ) )
>>> print ( marklist )
(('Ganesh', 90), ('Siva Sundar', 88), ('Gunasekhar', 75),
('Santosh', 72))
```

The *zip* function is extremely powerful as it can take iterators like strings and dictionaries as arguments. It can also take more than two iterables as arguments, and the iterables can be of varying lengths. It even gives options for iterating all the elements of the longer tuple if the iterable arguments are not of the same length.

Combining two lists into a dictionary

We can combine two lists and create a dictionary by using the *zip* function.

```
>>> names = [ "Ganesh", "Siva Sundar", "Gunasekhar",
"Santosh" ]
>>> marks = [ 90, 88, 75, 72 ]
>>> marklist = dict ( zip ( names, marks ) )
>>> print ( marklist )
{'Ganesh': 90, 'Siva Sundar': 88, 'Gunasekhar': 75,
'Santosh': 72}
```

The *pass* keyword

The *pass* keyword can be used as a placeholder for incomplete code. This keyword is a statement by itself and results in NOP (no operations) as it is discarded during the byte-compile phase and nothing is executed. It is useful for code that you want to add later. Two examples are given below.

Example 1:

```
def passDemo ():
```

```
  pass
passDemo ()
```

Example 2:

```
marks = 80
if marks >= 75:
   pass
```

Counter function in *collections* module

The *collections* module of Python has different types of containers. *Counter*, one of the containers in the *collections* module, is a *dict* subclass that provides an efficient way to count objects. The *keys* component of the dictionary stores the object, whose occurrence has to be counted, and the *values* component of the dictionary holds the object's count, that is, its frequency. *Counter* takes an iterable list as an argument and returns a *dict* object containing each distinct item in the iterable list and its frequency as a key-value pair. For example:

```
>>> from collections import Counter
>>> alphabets = [ 'a', 'b', 'c', 'd', 'b', 'a', 'b', 'c',
'a', 'b', 'c' ]
>>> num_alphabets = Counter ( alphabets )
>>> print ( num_alphabets )
Counter({'b': 4, 'a': 3, 'c': 3, 'd': 1})
```

Notice that the key-value pairs are ordered as per the descending order of the count of the occurrences in the iterable. The *Counter* can also be used to merge two dictionaries. The keys are preserved and the values of matching keys are added. For example:

```
>>> from collections import Counter
>>> marks1 = { 'Ganesh': 90, 'Siva Sundar': 88, 'Gunasekhar':
75, 'Santosh': 72 }
>>> marks2 = { 'Ganesh': 90, 'Siva Sundar': 88, 'Gunasekhar':
75, 'Santosh': 72, 'Shashank':78 }
>>> finalmarks = Counter( marks1 ) + Counter ( marks2 )
>>> print ( finalmarks )
Counter({'Ganesh': 180, 'Siva Sundar': 176, 'Gunasekhar':
150, 'Santosh': 144, 'Shashank': 78})
```

Expression evaluation using the *eval* function

The built-in *eval* function accepts a string as an argument. If the string is an expression, then the expression is evaluated. This is useful for evaluating any Python expression that comes as an input dynamically. For example:

```
>>> x = 7
>>> y= 45
>>> eval('x * y')
315

>>> eval ('2 ** 8')
256

>>> z = eval('x * y + 4')
>>> print (z)
319
```

Factorial

Factorial of a non-negative integer 'n' is the product of all positive integers less than or equal to 'n', but greater than or equal to 1. The math module of Python has a function named *factorial* to compute the factorial of an integer passed as an argument to the function.

```
>>> import math
>>> num = 9
>>> fact = math.factorial ( num )
>>> print ( fact )
362880
```

Playing sound notifications in Python

If you need to notify the user of the sound of some event, you can use the *chime* module to generate the standard sounds.

```
import chime
chime.info ()
chime.success ()
chime.error ()
chime.warning ()
```

The above tips will help programmers code faster, create more efficient and concise code, and will increase their productivity. The use of powerful features like *zip*, *Counter* and list comprehension makes the code concise and easy to understand. Happy Pythoning! END

References

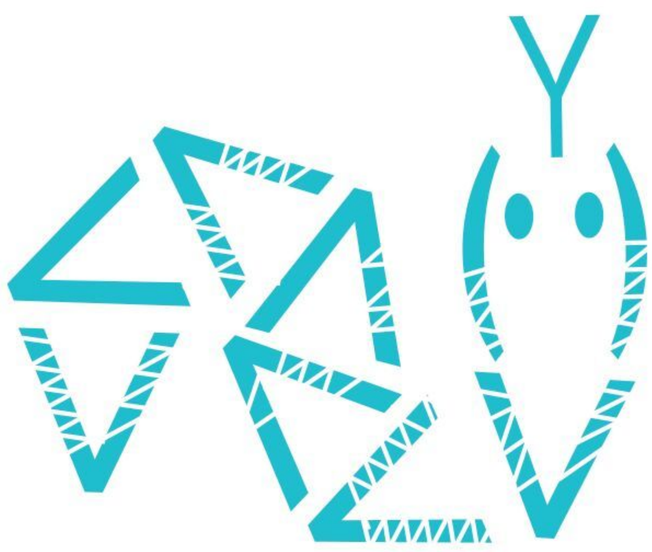
- *https://www.tiobe.com/tiobe-index/*
- *https://pypl.github.io/PYPL.html*
- *https://www.opensourceforu.com/2021/04/python-a-few-useful-tips-for-beginners/*

By: Sathyanarayanan S.

The author is working with Sri Sathya Sai University for Human Excellence, Gulbarga. He has more than 30 years of experience in systems management and in teaching IT courses. He is an enthusiastic promoter of FOSS.

AI: Anaconda and More on Probability

In the previous article in this series on AI and machine learning, we began by exploring the intricacies of using TensorFlow, a very powerful library used for developing AI and machine learning applications. Then, we discussed probability and paved the way for many of our future discussions. In this article, the fifth in the series, we will continue exploring concepts in probability and statistics.



To begin with, let us first install Anaconda, a distribution of Python for scientific computing especially useful for developing AI, machine learning and data science based applications. Later, we will also introduce a Python library called Theano in the context of probability theory. But before doing that, let us discuss the future of AI a little bit.

While going through the earlier articles in this series for evaluation and course correction, I wondered whether my tone was a bit sceptical at times while discussing the future of AI. By being brutally honest about some of the topics we discussed, I may have inadvertently demotivated a tiny fraction of the readers and potential practitioners of AI. This made me research the financial aspects of AI and machine learning. I wanted to identify the kind of companies involved in the AI market. Are the bigwigs heavily involved or are there just some startups trying to move away from the pack? Another important question was: how much money will be pumped into the AI market by these companies in the near future? Is it going to be just a couple of million, or a few billion or maybe even a few trillion?

For the predictions and data stated here, I depended on articles published in respected newspapers in the recent past rather than on scholarly articles which I couldn't grasp well to understand the complex dynamics behind the development of an AI based economy. An article published by *Forbes* in had 2020 predicted that companies would spend 50 billion dollars on AI in financial year 2020. Well, that is a really huge investment, even if we acknowledge that the American billion (10^9) is just one-thousandth of the British billion (10^{12}). An article published in *Fortune*, another American business magazine, reported that venture capitalists are shifting some of their focus from artificial intelligence to newer and trendier fields like Web3 and decentralised finance (DeFi). But *The Wall Street Journal* (an American business-focused newspaper) in 2022 confidently predicted that, 'Big Tech Is Spending Billions on AI Research. Investors Should Keep an Eye Out'.

What about the Indian scenario? *Business Standard* reported in 2022 that 87 per cent of the Indian companies will hike their AI expenditure by 10 per cent in the next three years. So, overall, it looks like the future of AI is very safe and bright. But which are the top companies investing in AI? As expected, all the giants like Amazon, Meta (Facebook), Alphabet (Google), Microsoft, IBM, etc, are doing so. But, surprisingly, companies like Shell, Johnson & Johnson, Unilever, Walmart, etc, whose primary focus is neither computing nor information technology, are also in the list of big companies heavily investing in AI. And yes! Amazon is a tech company and not an e-commerce company.

So it is clear that many of the giant companies in the world believe that AI is going to play a prominent role in the near future. But what changes and new trends are these leaders expecting in the future? Again, to find some answers, I depended on news articles and interviews, rather than on scholarly articles. The terms frequently mentioned in context with future trends in AI include responsible AI, quantum AI, IoT with AI, AI and ethics, automated machine learning, etc. I believe these are vast topics and we will discuss some of these terms (except AI and ethics, which we already discussed in the last article) in detail, in the coming articles in this series.

An introduction to Anaconda

Let us now move to discussing the necessary tech for AI. From this article onwards, we will start using Anaconda whenever necessary. Anaconda is a distribution of the Python and R programming languages for scientific computing. It helps a lot by simplifying package management.

Now, let us install Anaconda. First, go to the web page *https://www.anaconda.com/products/distribution#linux* and download the latest version of the Anaconda distribution installer. Open a terminal in the directory to which the distribution installer is downloaded. As of writing this article (October 2022), the latest Anaconda distribution installer for systems with 64-bit processors is *Anaconda3-2022.05-Linux-x86_64.sh*. In order to check the integrity of the installer, run the following command on the terminal:

```
shasum -a 256 Anaconda3-2022.05-
Linux-x86_64.sh
```

If you have downloaded a different version of the installer, then use the name of that version after the command '*shasum -a 256*'. You will see a hash value displayed on the terminal, followed by the name of the installer. In my case, the line displayed was:

```
a7c0afe862f6ea19a 596801fc138bde0463
abcbce1b753e8d5c474b506a2db2d
Anaconda3-2022.05-Linux-x86_64.sh
```

Now, go to the web page *https://docs.anaconda.com/anaconda/install/hashes* and go to the hash value of the corresponding version of the installer you have downloaded. Make sure that you continue with the installation of Anaconda only if the hash value displayed on the terminal and the hash value given in the web page are exactly the same. If the hash values do not match, then the installer downloaded is corrupted. In this case, restart the installation process once again. Now, if the hash values match, execute the following command on the terminal to begin the installation:

```
bash Anaconda3-2022.05-Linux-x86_64.sh
```

But in case you have downloaded a different installer, then use the appropriate name after the command *bash*. Now, press *Enter*, scroll down to go through the agreement and accept it. Finally, type '*yes*' to start the installation. Whenever a prompt appears, stick to the default option provided by Anaconda (unless you have a very good reason not to do so). Now, the Anaconda installation is finished.

Anaconda installation by default also installs Conda, a package manager and environment management system. Wikipedia says that the Anaconda

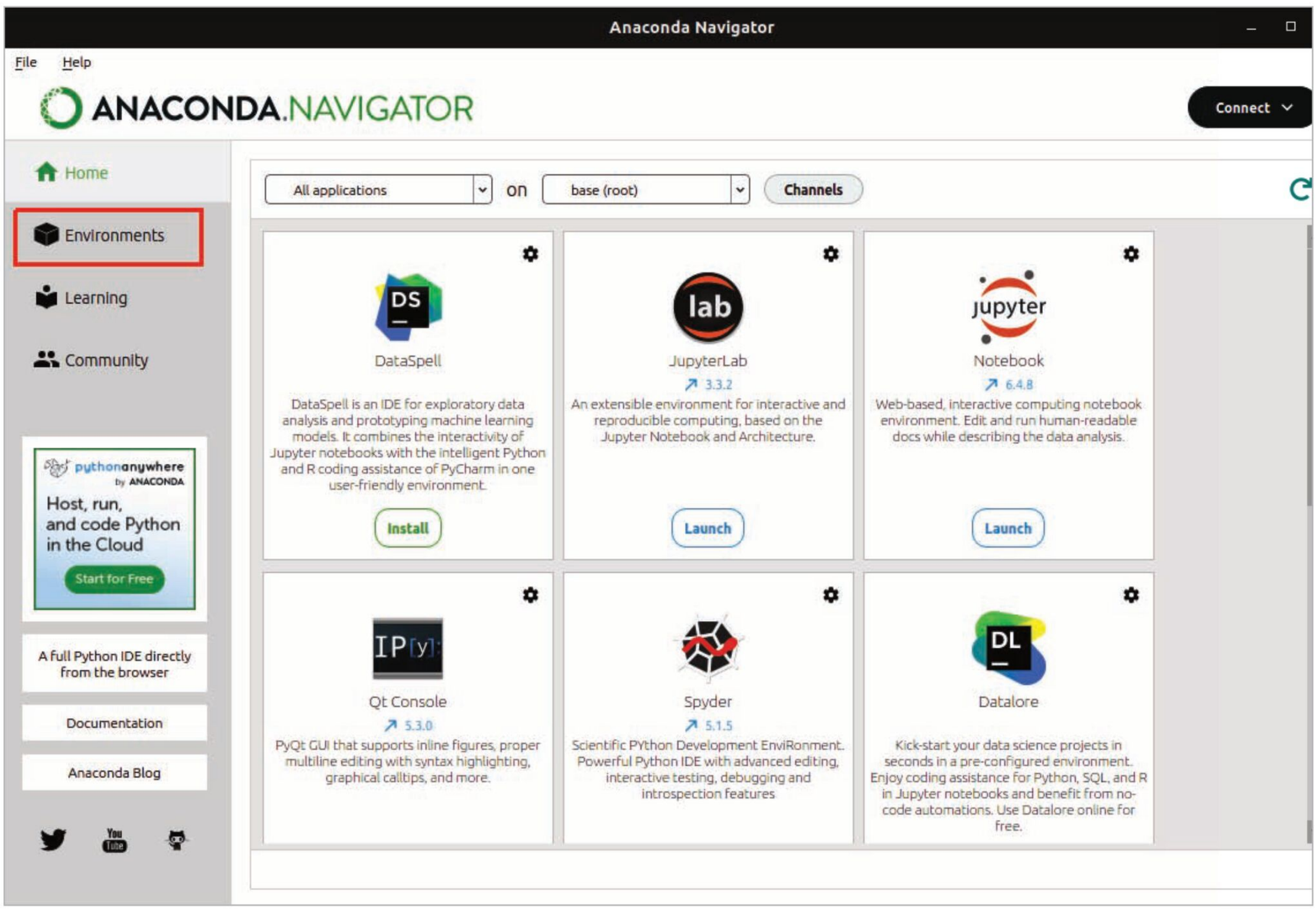


Figure 1: Anaconda Navigator

distribution comes with over 250 packages automatically installed, with the option of installing over 7500 additional open source packages. The most important thing to remember is that whatever package or library you have installed by using Anaconda can be used in your Jupyter Notebook also. Whatever updation is required by other packages or libraries during the installation of a new package will be automatically handled by Anaconda carefully, without bothering you in the least.

So, finally, we have reached a place in our journey where we no longer need to worry about installing the new packages and libraries necessary to continue our quest for developing AI and machine learning based applications. Notice that Anaconda only has a command-line interface (CLI). Are we in trouble? No! Our installation also gives us Anaconda Navigator, a graphical user interface (GUI) for Anaconda. Execute the command '*anaconda-navigator*' on the terminal to run Anaconda Navigator (Figure 1). Soon, we will see an example of its power.

Introduction to Theano

Now, let us try to work with Theano, a Python library and optimising compiler for evaluating mathematical expressions. The installation of Theano is very easy since we have Anaconda Navigator. First, open Anaconda Navigator. On the top right corner you will see the button *Environments* (marked with a red box in Figure 1). Press that button. You will go to a window showing the list of all currently installed packages. From the drop-down list on top, choose the option *Not installed*. Scroll down, find the option *Theano*, and click on the checkbox on the left. Now press the green button named *Apply* at the bottom right corner of the window. Anaconda will find out all the dependencies for installing Theano and show them in a pop-up menu. Figure 2 shows the pop-up menu I got during the installation of Theano in my system. You can see that in addition to Theano, one more new package is installed and eight other packages are modified. Imagine how difficult this would have been if I was trying to manually install Theano. But with the help of Anaconda, all we need to do is press the button *Apply* on the bottom right corner of this pop-up menu. After a while, the installation of Theano will be completed smoothly. Now, we are ready to use Theano in our Jupyter Notebooks.

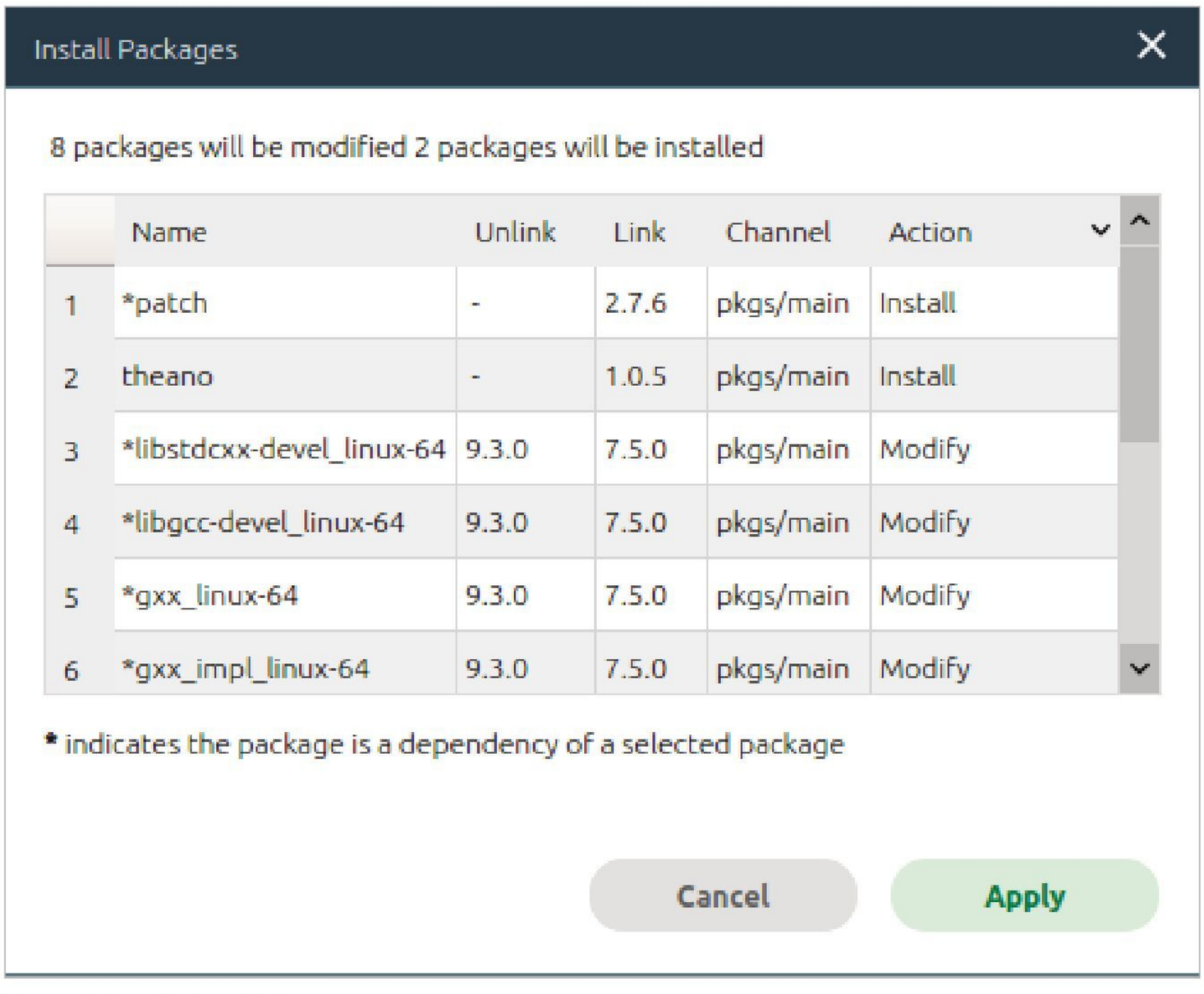


Figure 2: Installation of Theano

We are already familiar with the Python library called *SymPy* used for symbolic calculation but Theano takes it to the next level. Let us see a simple example to understand the power of Theano. Consider the code shown in Figure 3. First, let us try to understand the code. Line 1 of the code imports Theano. Line 2 imports *tensor* and names it *T*. We have already encountered tensors when we discussed TensorFlow. Mathematically, tensors can be treated as multi-dimensional arrays. *Tensor* is one of the key data structures used in Theano and can be used to store and manipulate scalars (numbers), vectors (1-dimensional arrays), matrices (2-dimensional arrays), tensors (multi-dimensional arrays), etc. In Line 3, a Theano function called *function( )* is imported. Line 4 imports a Theano function called *pp( )*, which is used for pretty-printing (printing in a format appealing to humans). Line 5 creates a symbolic scalar variable of type double called *x*. It is a bit tricky to understand these sorts of symbolic variables. Think of it as a 0-sized double variable with no value or storage associated. Similarly, Line 6 creates another symbolic scalar variable called *y*. Line 7 acts like a function in a sense. This line tells the Python interpreter something like the following, 'if and when symbolic scalar variables *x* and *y* get some values, add those values and store inside me'.

To further illustrate symbolic operations at this point, let us try to understand Line 8. From Figure 3, it is clear that the output of this line of code is *(x+y)*. So, it is clear that actual addition of two numbers is yet to take place. Lines 9 to 11 similarly define symbolic subtraction, multiplication,

and division, respectively. If you want further clarity, use the function *pp( )* to find the values of *b*, *c*, and *d*. Line 12 is very critical. It defines a function named *f( )* using the function *function( )* of Theano. The function *function( )* takes two parameters. Parameter 1 is the data to be operated upon and parameter 2 is the function to be applied on the data given as parameter 1. Here, the data is the two symbolic scalar variables *x* and *y*. The function is given by *[a b c d]* which represents symbolic addition, subtraction, multiplication, and division, respectively. Finally, in Line 13, the function *f( )* is provided with actual values rather than symbolic ones. The output of this operation is also shown in Figure 3. It is very easy to verify that the output shown is correct.

Now, let us see how matrices can be created and manipulated using Theano. Consider the code shown in Figure 4. It is important to note that the first three lines of code shown in Figure 3 should be added here also, if you are writing this program independent of the programs we have discussed so far. Now, let us try to understand the code. Line 1 of the code creates two symbolic matrices *x* and *y*. Think of *x* and *y* as two 0-dimensional arrays. But this time these matrices are of type integer unlike the last time where the data type was double. Further, this time a plural constructor (*imatrices*) is used so that more than one matrix can be constructed at the same time. Lines 3 to 5 perform symbolic addition, subtraction, and multiplication, respectively, on symbolic matrices *x* and *y*. Here again you can use *print(pp(a))*, *print(pp(b))*, and *print(pp(c))* to better understand the symbolic nature of the operations being performed. If you add these lines of code, you will get *(x+y)*, *(x-y)*, and *(x \dot y)*, respectively, as output. Line 5 generates a function *f( )* which takes two parameters as before. Again, parameter 1 is the data to be operated upon and parameter 2

```
1  import theano
2  import theano.tensor as T
3  from theano import function
4  from theano import pp
5  x = T.dscalar('x')
6  y = T.dscalar('y')
7  a = x + y
8  print(pp(a))
9  b = x - y
10 c = x * y
11 d = x / y
12 f = function([x, y], [a, b, c, d])
13 f(333, 222)
```

```
(x + y)
[array(555.), array(111.), array(73926.), array(1.5)]
```

Figure 3: Our first code using Theano

```
1  x, y = T.imatrices('x', 'y')
2  a = x + y
3  b = x - y
4  c = x.dot(y)
5  f = function([x, y], [a, b, c])
6  f([[1, 2],[3, 4]], [[5, 6],[7, 8]])
```

```
[array([[ 6,  8],
        [10, 12]], dtype=int32),
 array([[-4, -4],
        [-4, -4]], dtype=int32),
 array([[19, 22],
        [43, 50]], dtype=int32)]
```

Figure 4: Manipulating matrices with Theano

```
1  import numpy as np
2  print(np.mean(C1), np.std(C1))
3  print(np.mean(C2), np.std(C2))
```

```
21.0 1.0
21.666666666666668 11.813363431112899
```

Figure 5: Arithmetic mean and standard deviation

is the function to be applied on the data given as parameter 1. But this time, the data is the two symbolic matrices *x* and *y*. The function is given by *[a b c]* which represents symbolic addition, subtraction, and multiplication, respectively. Finally, in Line 6, the function *f( )* is provided with actual values rather than symbolic ones. The two matrices given as input to *f( )* are [[1, 2], [3, 4]] and [[5, 6], [7, 8]]. The output of this operation is also shown in Figure 4. It is very easy to verify that the three output matrices shown are correct. Notice that, in addition to scalars and matrices (for both of which we saw examples now), *tensor* also offers constructors like vector, row, column, different types of tensors, etc. Let us stop discussing Theano for now and revisit it while discussing advanced topics in probability and statistics.

Baby steps with probability

Now, let us continue discussing probability and statistics. I had suggested in the last article (while introducing probability) that you carefully go through three Wikipedia articles, and with that assumption of familiarity I tried to motivate you by discussing the Normal distribution. However, we must revisit some of the basic notions of probability and statistics before we start developing AI and machine learning based applications. I hope all of you have heard about arithmetic mean and standard deviation (SD). Arithmetic mean can be thought of as the average of a set of values. SD can be thought of as the variation or dispersion of a set of values. If the SD value is low then the elements in the set tend to be closer to the mean. On the contrary, if the SD value is high then the elements of the set are spread out over a wider range. But how can we calculate arithmetic mean and SD using Python? There is a module called *statistics* available with Python which can be used to find mean and standard deviation. However, experts are of the opinion that this module is very slow and hence we choose *NumPy*. Now, consider the code shown in Figure 5.

The code prints the mean and standard deviation of two lists C1 and C2 (whose values are hidden from you for the time being because of obvious reasons). What interpretations can you make from these values? Nothing! They are just numbers for you right now. But what if I tell you the lists contain the marks of six students, studying in 10th standard in school A and school B, for an exam in mathematics (the exam is out of 50 with a pass mark of 20). The mean value tells us that students from both the schools have relatively poor average marks with

school B slightly outperforming school A. But what does the standard deviation value tell us? To further your understanding, I will give you the two lists, '*C1 = [20, 22, 20, 22, 22, 20]*' and '*C2 = [18, 16, 17, 16, 15, 48]*'. Though hidden by the mean, the large standard deviation value of 11.813363431112899 clearly captures the mass failure in school B. This example clearly tells us that we need more complicated parameters to understand the complex nature of the problems we deal with. Probability and statistics will come to our rescue here by providing more and more complex models which can imitate complex and chaotic data.

Random number generation is an essential part of probability. But, in practice, only pseudorandom number generation (a sequence of numbers whose properties approximate the properties of sequences of random numbers) is possible. Now, let us see a few functions that will help us generate pseudorandom numbers. Consider the code shown in Figure 6. First, let us try to understand the code. Line 1 imports the random package of Python. In Line 2, the function *random.random( )* generates random numbers. This line of code, '*new_list = [random.random() for i in range(2)]*', uses a technique called list comprehension to generate two random numbers and store them in the list named *new_list*. Line 3 prints this list as output. Notice that the two random numbers printed change with each iteration of the code, and the probability of getting the same numbers printed twice consecutively is theoretically zero.

The single line of code in the second cell shown in Figure 6 uses the function *random.choice( )*. This function makes an equally likely choice from all the choices given to it. In this case, the code fragment *random.choice(["Heads", "Tails"])* will make a choice between 'Heads' or 'Tails' with equal probability. Notice that this line of code also uses the technique called list comprehension so that three successive choices between 'Heads' or 'Tails' are made. Figure 6 shows the output of this code, where the option 'Tails' is chosen for three consecutive times.

```
1 import random
2 new_list = [random.random() for i in range(2)]
3 print(new_list)
```

```
[0.18869669339036488, 0.4894201070245773]
```

```
1 print([random.choice(["Heads", "Tails"]) for i in range(3)])
```

```
['Tails', 'Tails', 'Tails']
```

Figure 6: Pseudo random number generation

```
1 import random
2 n=1000
3 ct=0
4 for i in range(n):
5     if random.randint(1, 6) == 6:
6         ct=ct+1
7 print(ct/n)
8 print(1/6)
```

```
0.179
0.16666666666666666
```

Figure 7: Illustrating the law of large numbers

Now, let us try to illustrate a very popular theorem in probability, the law of large numbers (LLN), with a simple example. LLN states that the average of the results obtained from a large number of trials should be close to the expected value. Further, this average tends to come closer and closer to the expected value as more and more trials are performed. We all know that if a fair dice is thrown, the probability of getting the number 6 is 1/6. We now simulate this experiment with a simple Python code shown in Figure 7. Line 1 imports the *random* package of Python. Line 2 sets the number of trials to be performed. In this case, it is 1000. Line 3 initialises the counter *ct* to zero. Line 4 sets a loop, which iterates 1000 times in this case. Line 5 uses the function *random.randint(1, 6)*. In this case, the function generates an integer between 1 and 6 (inclusive of both 1 and 6) randomly. The *if* statement in Line 5 checks whether the number generated is equal to 6; if yes the control goes to Line 7 and increases the counter *ct* by 1. After the loop is iterated a 1000 times, Line 8 will be executed. Line 8 prints the ratio between the number of occurrences of the number 6 and the total number of trials (1000).

Figure 7 also shows this number as 0.179 (slightly more than the expected value 1/6 = 0.1666…which is also printed in the output). Not that close to the expected value, right? In Line 2, set the value of *n* as 10000. Run the code again and observe the new output printed. Most probably you will get a number which is a little bit closer to the expected value. Notice that this could also be a number less than the expected value. Increase the value of *n*, in Line 2, for a few more times. You will observe that the output is inching closer and closer to the expected value. Thus, with a simple code, we have illustrated LLN.

Though a simple statement, you will be amazed to know that the list of mathematicians who worked on a proof for LLN or tried to refine one include Cardano, Jacob Bernoulli, Daniel Bernoulli, Poisson, Chebyshev, Markov, Borel, Cantelli, Kolmogorov, Khinchin, etc. All of them are mathematical giants in their own field.

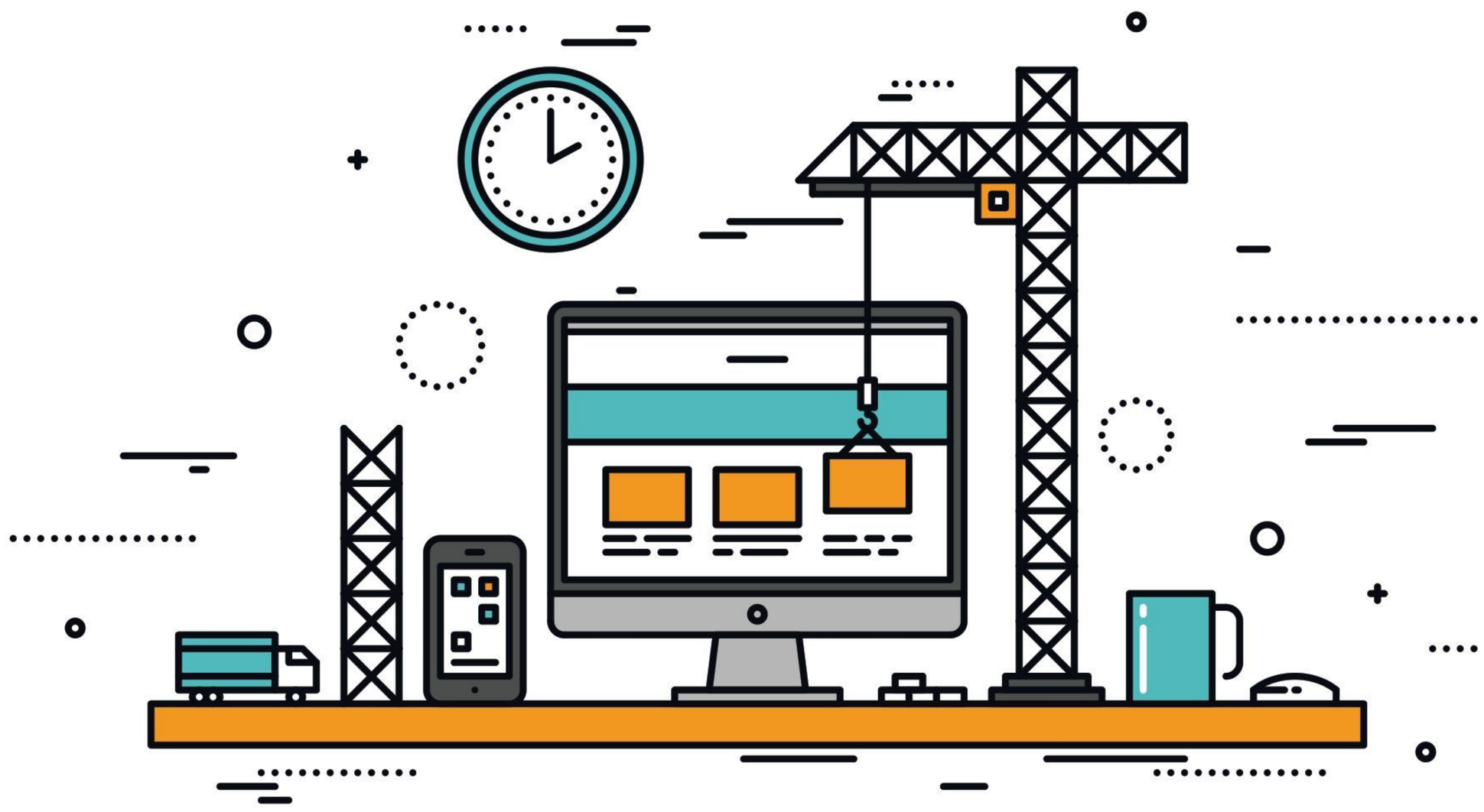
We are yet to cover topics in probability like random variables, probability distributions, etc, which are essential in developing AI and machine learning based applications. Our discussion of topics in probability and statistics is still in the early stages and we will try to strengthen our knowledge of these subjects in the next article also. At the same time, we will revisit two of our old friends we met in this journey earlier, the libraries Pandas and TensorFlow. We will also befriend a relative of TensorFlow called Keras, a library which acts as an interface for TensorFlow. END

By: Deepu Benson

The author is a free software enthusiast. His area of interest is theoretical computer science. The open source tools of his choice include ns-2 and ns-3. He maintains his own technical blog too.

Build a Web Application on Firebase

Firebase is a platform backed by Google for building, growing and deploying web apps, mobile apps and games. This article illustrates developing a complete single-page web application on Firebase using its services, namely, FireStore, authentication and hosting. The readers can follow the steps given here and develop a web application with ease.



This article will handhold you to develop a simple web application to maintain a journal of events.

Requirements

To start with, let us specify a clear requirement. Requirement specification is one of the most important aspects of the development process. Here is the list of requirements.

- The journal should be accessible only via login.
- The login credentials are predefined by the system administrator, and consist of email and password.
- A logged-in user should be able to add a new entry into the journal.
- A logged-in user should be able to view the entries in the journal.
- A logged-in user should be able to sort the entries chronologically.
- Each entry must auto-fill the details like date of entry and author.

Architecture

The application is designed around the Firebase cloud platform. The platform offers services, scale, SDKs, tools and well-maintained documentation. The following services are used for this application.

- ***FireStore:*** The real-time database service for storing the journal entries.
- ***Hosting:*** The service to host the web application.
- ***Authentication:*** The service to offer login-based access to the application.
- ***Firebase SDK:*** The NPM module that offers commands and communication API. The JS client uses this API to interact with the FireStore and authentication services at runtime.

The architecture is depicted in Figure 1.

So what do you need to get started with building the application? Here is the list of basic prerequisites.

- An account with Google (like a Gmail account)
- Node 10+ distribution on the development machine

- An IDE like Visual Studio Code
- A web browser

Procedure

The following sections give the step-by-step procedure to be followed to configure Firebase, and develop and deploy the application.

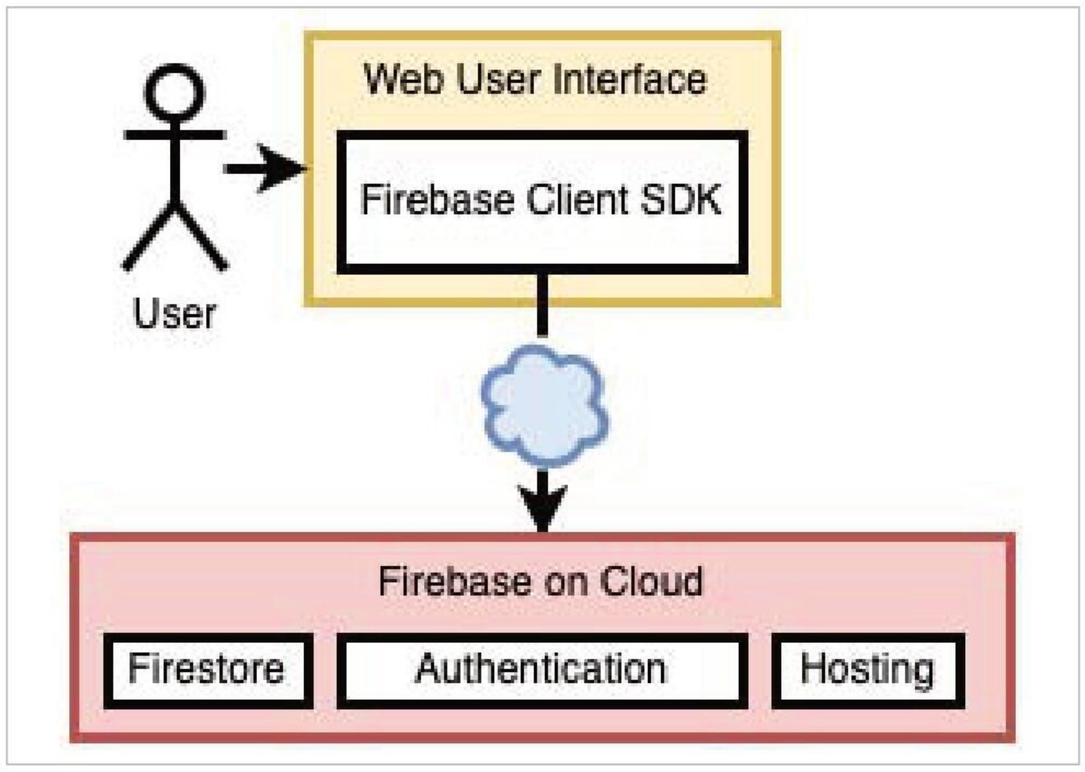


Figure 1: Architecture

1. Create the project

Visit *https://firebase.google.com/* and sign in using your Gmail account. It opens the Firebase console to add or create a new project. We used *glarimy-firebase* as the name for the project. You may or may not select Google Analytics in the project creation wizard. The *Project Review* page opens up once the project is created successfully. This page lists various services and tools available in different categories. Expand the *Build* category on the left pane of the overview page to locate *Authentication*, *Firestore Database* and *Hosting* services that we intend to use.

2. Set up the database

We intend to use *Firestore Database* to store the journal entries. Note that Firestore is a NoSQL database. It organises the data into collections of documents. Once the database is enabled, you can start creating collections.

Click on the *Firestore Database* under the *Build* category. It opens the *Cloud Firestore* page. Create a new database by clicking on the *Create Database* button. Choose *Start in Production Mode* and also choose an appropriate *Cloud Firestore* location. We chose *asia-south1*. With this the database is ready.

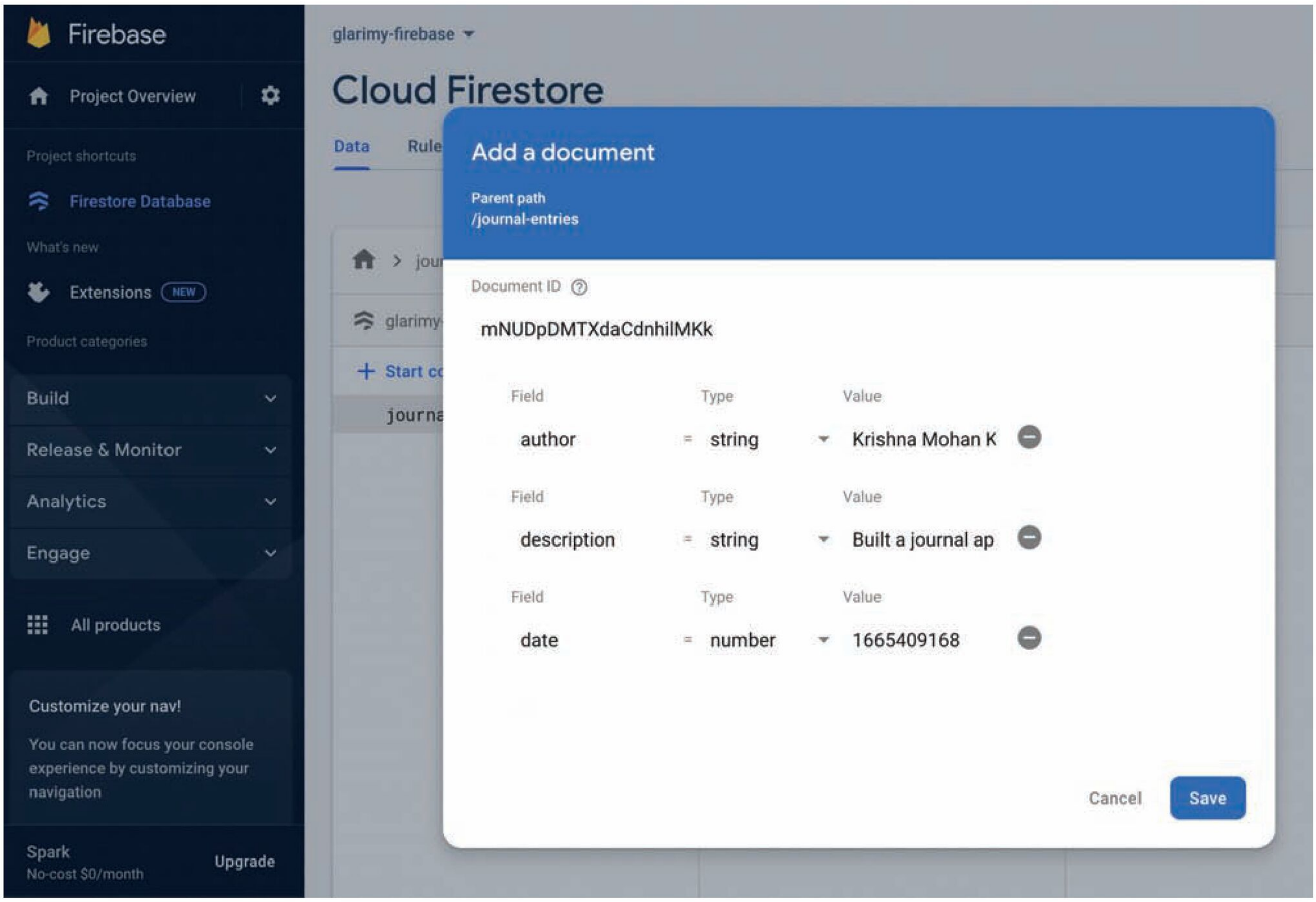


Figure 2: Database setup

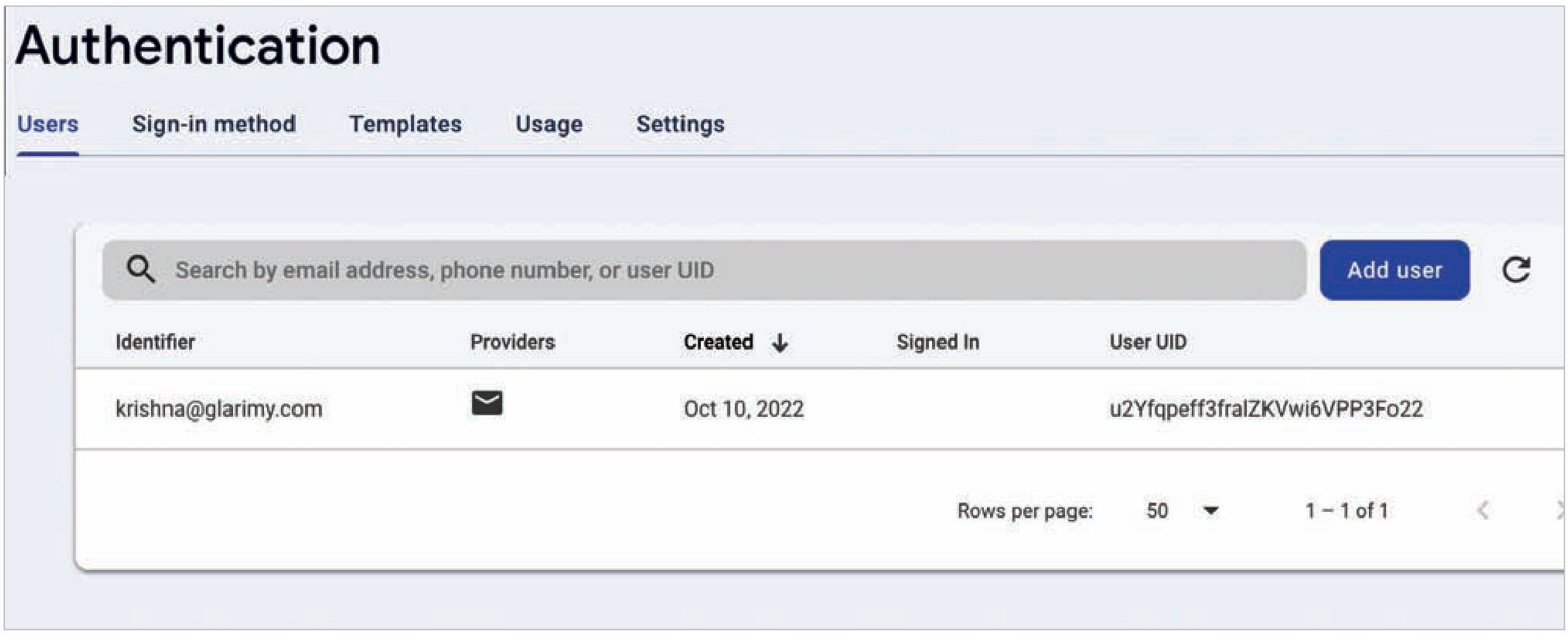


Figure 3: Authentication setup

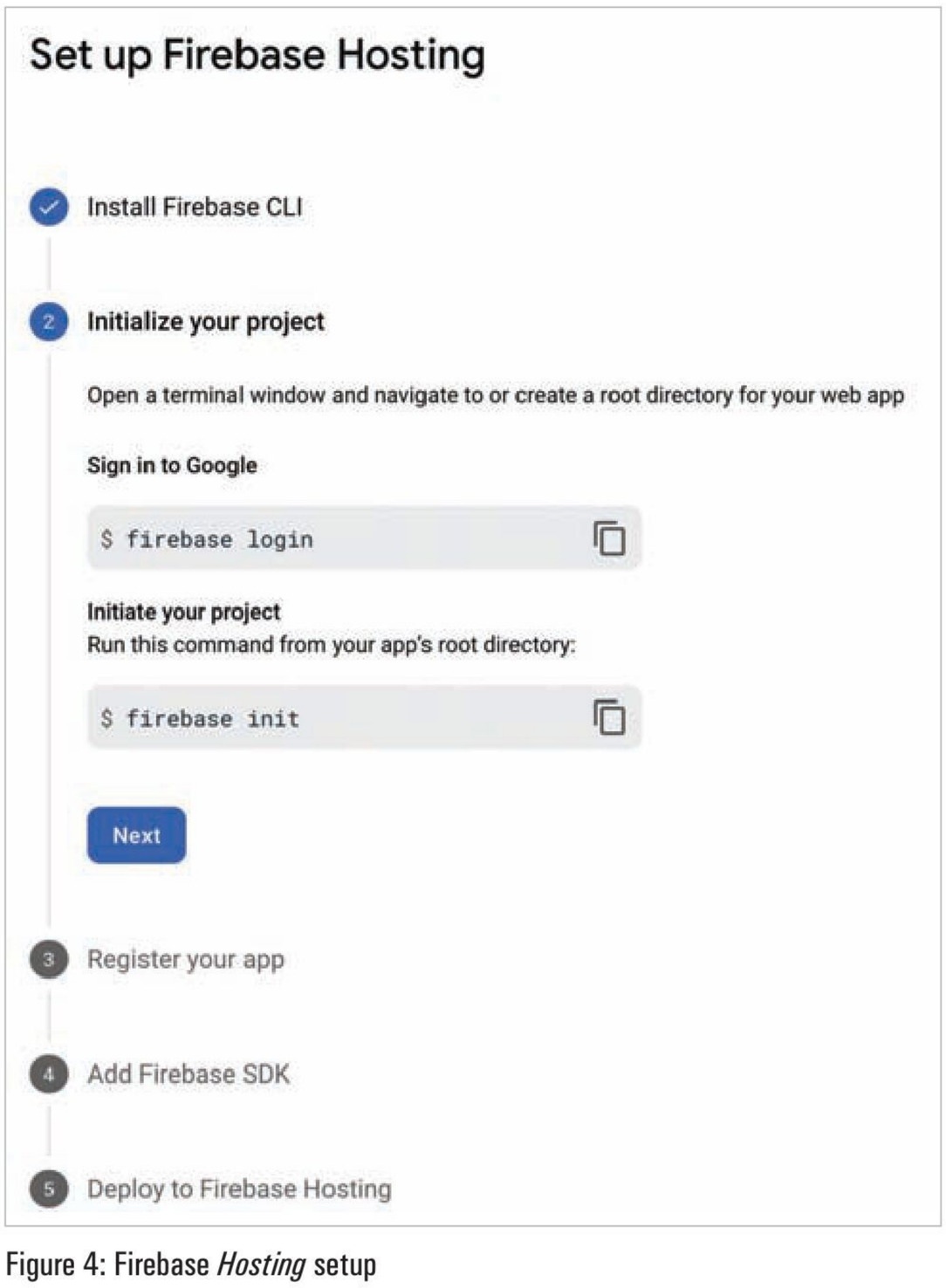


Figure 4: Firebase *Hosting* setup

Now, create a collection with the name *journal-entries*. Create a dummy document with an auto-generated ID along with the structure, as shown in Figure 2. The document consists of three fields, namely, author (string), date (number to store the time in UNIX epoch format) and description (string).

All the journal entries will be stored in this *journal-entries* collection at runtime.

3. Set up authentication

We intend to control access to the application via a login. For that, we need to configure authentication.

Click on *Authentication* under the *Build* category of the *Project Review* pane. It opens the *Authentication* page. Click on *Get Started*. Choose and enable email/password as the sign-in provider. In other words, users must provide an email-id and password to sign in to the web app.

We want to restrict the app to a few predefined users. So, add new users under the *User* tab, as shown in Figure 3. We want only the users in this list to be able to sign in to the application.

Do not worry about the privacy of the passwords. Once the admin enters the details, no one (including the admin) can view the password again in a readable format.

4. Develop the client

Click on *Hosting* under the *Build* category of the *Project Review* pane. It opens the *Hosting* page. Click on *Get Started*. The wizard looks as shown in Figure 4. Follow the instructions to install Firebase SDK on your machine and develop the client application.

At Step 4 of the wizard, make a note of the generated *firebaseConfig*. In our case, it looks somewhat like the following:

```
const firebaseConfig = {
    apiKey: "AIzaSyXXXXXXXXXXXXXwwvZgtbrnLiqKFPE",
    authDomain: "glarimy-firebase.firebaseapp.com",
    projectId: "glarimy-firebase",
    storageBucket: "glarimy-firebase.appspot.com",
    messagingSenderId: "782212347913 9",
    appId: "1:782281879139:web:aad87vdvcdeee3914d256",
    measurementId: "G-F86URYCS0X"
};
```

By now, you will have a project folder created on your development machine with various folders and files generated. Locate the folder named *public* in the project folder. We can develop the client application in this folder. The application consists of four important kinds of files like in any other web client: 1) The HTML files, 2) The CSS files, 3) The JS script files, 4) Other assets like images, etc.

Here is the public/*index.html* file for the application:

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <title>Glarimy Journal</title>
  <link rel="stylesheet" href='./styles.css'>
  <script defer src="/__/firebase/8.4.1/firebase-app.js"></
script>
  <script defer src="/__/firebase/8.4.1/firebase-auth.js"></
script>
  <script defer src="/__/firebase/8.4.1/firebase-database.
js"></script>
  <script defer src="/__/firebase/8.4.1/firebase-firestore.
js"></script>
  <script defer src="/__/firebase/8.4.1/firebase-messaging.
js"></script>
  <script defer src="/__/firebase/8.4.1/firebase-storage.js"></
script>
  <script src='./index.js'></script>
</head>

<body>
  <div id='top'>
    <h1>Our Story</h1>
    <div id='current'></div>
    <div id='nav'>
      <button id='new'> New</button>
      <button id='refresh'>Refresh</button>
      <button id='latest'> Latest</button>
      <button id='oldest'> Oldest</button>
    </div>
  </div>
  <div id='auth'>
    <h2>Login Required</h2>
    <div><input type='text' placeholder="email" id='email'
/></div>
    <div><input type='password' placeholder="password"
id='password' /></div>
    <div>
      <input type='submit' value='Login' id='login' />
      <input type='reset' value='Cancel' id='resent' />
    </div>
  </div>

  <div id='form'>
    <h2>Record Your Story</h2>
    <input type='date' id='date'>
    <div>
      <textarea id='description'></textarea>
    </div>
    <button id='submit'>Record</button>
  </div>

  <div id='journal'></div>
</body>

</html>
```

Observe that the above HTML file is referring to a file named *index.js* in the same folder. The *index.js* file consists of all the logic. It uses the Firebase SDK extensively. Note that this is the file that also consists of the *firebaseConfig*. Here is the code:

```
const firebaseConfig = {
    apiKey: "AIzaSyXXXXXXXXXXXXXwwvZgtbrnLiqKFPE",
    authDomain: "glarimy-firebase.firebaseapp.com",
    projectId: "glarimy-firebase",
    storageBucket: "glarimy-firebase.appspot.com",
    messagingSenderId: "7822123479139",
    appId: "1:782281879139:web:aad87vdvcdeee3914d256",
    measurementId: "G-F86URYCS0X"
};
var db = undefined;
var user = undefined;

async function login(email, password) {
    await firebase.auth().signInWithEmailAndPassword(email,
password);
    user = firebase.auth().currentUser;
}

function activate() {
    document.querySelector('#date').valueAsDate = new Date();
    document.querySelector('#login').
addEventListener('click', async function () {
        let email = document.querySelector('#email').value;
        let password = document.querySelector('#password').
value;
        await login(email, password);
        await fetch('desc');
```

```
        document.querySelector('#current').innerHTML = user.
email;
        document.querySelector('#email').value = "";
        document.querySelector('#password').value = "";
        document.querySelector('#auth').style.display =
'none';
        document.querySelector('#nav').style.display =
'block';
        document.querySelector('#journal').style.display =
'block';
    });
    document.querySelector('#submit').
addEventListener('click', async function () {
        let date = document.querySelector('#date').value;
        let description = document.
querySelector('#description').value;
        if(description.trim().length == 0)
            return;
        if (new Date(date).getTime() > new Date().getTime())
            return;
        await db.collection("journal-entries").add({
            "description": description,
            "author": user.email,
            "date": new Date(date).getTime()
        });
        document.querySelector('#date').valueAsDate = new
Date();
        document.querySelector('#description').value = "";
        await fetch('desc');
        document.querySelector('#journal').style.display =
'block';
        document.querySelector('#form').style.display =
'none';

    });

    document.querySelector('#new').addEventListener('click',
async function () {
        document.querySelector('#journal').style.display =
'none';
        document.querySelector('#form').style.display =
'block';
    });

    document.querySelector('#refresh').
addEventListener('click', async function () {
        await fetch('desc');
        document.querySelector('#journal').style.display =
'block';
        document.querySelector('#form').style.display =
'none';
    });

    document.querySelector('#latest').
addEventListener('click', async function () {
        await fetch('desc');
        document.querySelector('#journal').style.display =
'block';
        document.querySelector('#form').style.display =
'none';
    });

    document.querySelector('#oldest').
addEventListener('click', async function () {
        await fetch('asc');
        document.querySelector('#journal').style.display =
'block';
        document.querySelector('#form').style.display =
'none';
    });
}

async function fetch(order) {
    let entries = await db.collection("journal-entries").
orderBy('date', order).get();
    document.querySelector('#journal').innerHTML = "";
    entries.forEach((entry) => {
        let record = entry.data();

        let doe = document.createElement("div");
        let date = new Date(record.date);
        doe.innerHTML = `&#128197; ${date.getDate()} ${date.
toLocaleString('default', { month: 'long' })} ${date.
getFullYear()} ${date.toLocaleDateString('default', {
weekday: 'long' })}`
        doe.classList = ['date'];

        let description = document.createElement("div");
        description.innerHTML = record.description;
        description.classList = ['description']

        let author = document.createElement("div");
        author.innerHTML = `&#9997; ${record.author}`;
        author.classList = ['author']

        let div = document.createElement("div");
        div.id = entry.id;
        div.classList = ['entry'];
        div.appendChild(doe);
        div.appendChild(description);
        div.appendChild(author);

        document.querySelector('#journal').appendChild(div);
    });
```

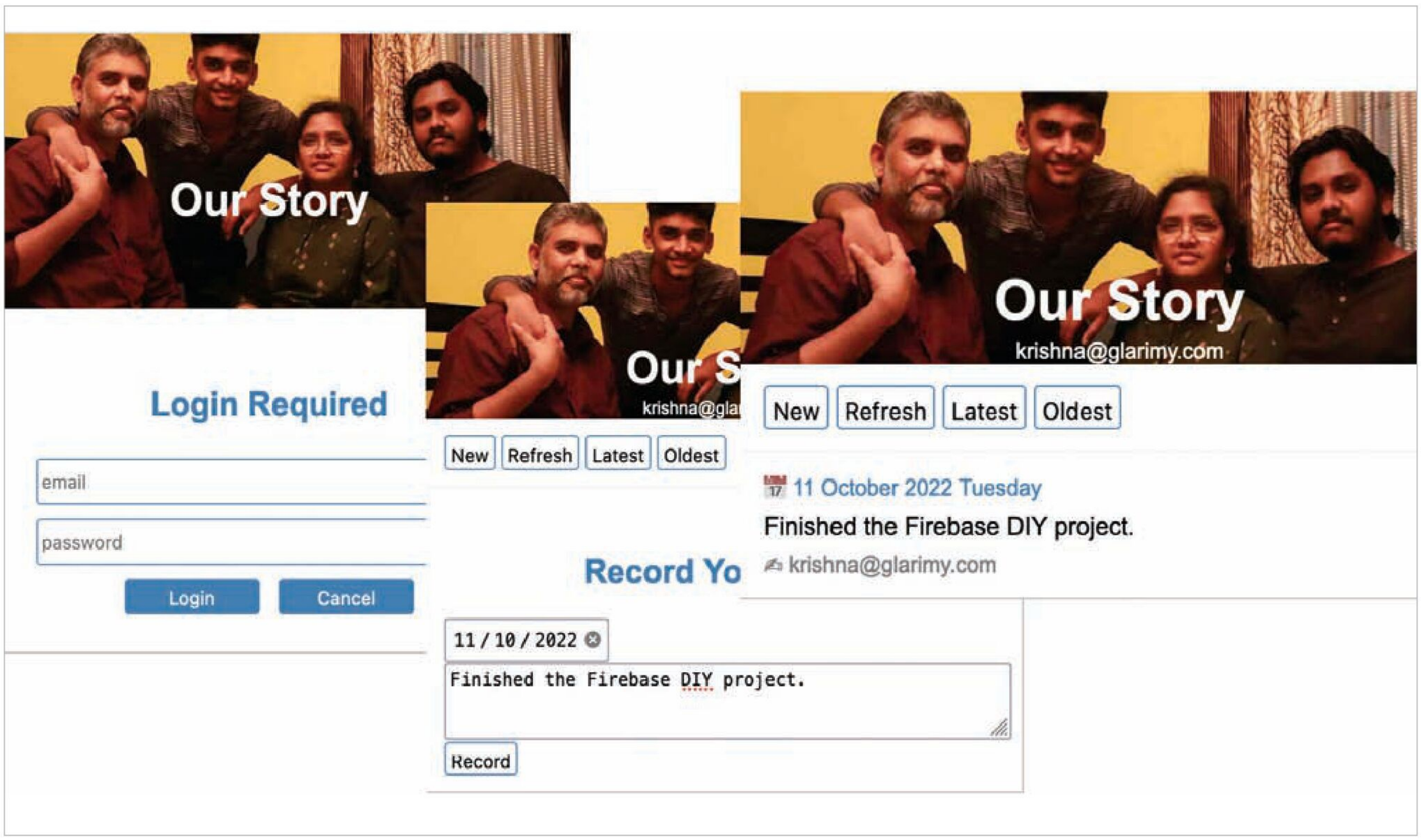


Figure 5: User interface

```
}

window.addEventListener('load', async function () {
    firebase.initializeApp(firebaseConfig);
    db = firebase.firestore();
    activate();
});
```

We also need to develop the stylesheet named *public/styles.css* which is being referenced in the HTML file. Refer to the repository specified towards the end of the article to find the stylesheet.

5. Deploy the web app

Once the application code is ready, deploy the application by running the following command from the root directory of the application on the development machine:

```
firebase deploy
```

It provides the URL at which the application is ready. Access the URL from the browser, and login with one of the user credentials that are configured to use the application. Figure 5 depicts the user interface.

This is the whole process of developing a simple web app backed by database and authentication on the Firebase platform. The code used for this application is available at *https://bitbucket.org/glarimy/glarimy-university/src/master/glarimy-firebase/* for free access.

You can improve this application with the following additional features:

1. Add authentication via sign-in providers like Google, GitHub, Yahoo, Facebook, Twitter, etc.
2. Add a feature to edit the journal entries.
3. Add a feature to upload images also along with the journal entries.
4. Add a feature to add tags to the entries and search based on the tags.
5. Add a feature to export the journal entries into XML, JSON and PDF formats.
6. Redevelop the client using Typescript/React/Angular instead of basic JavaScript.

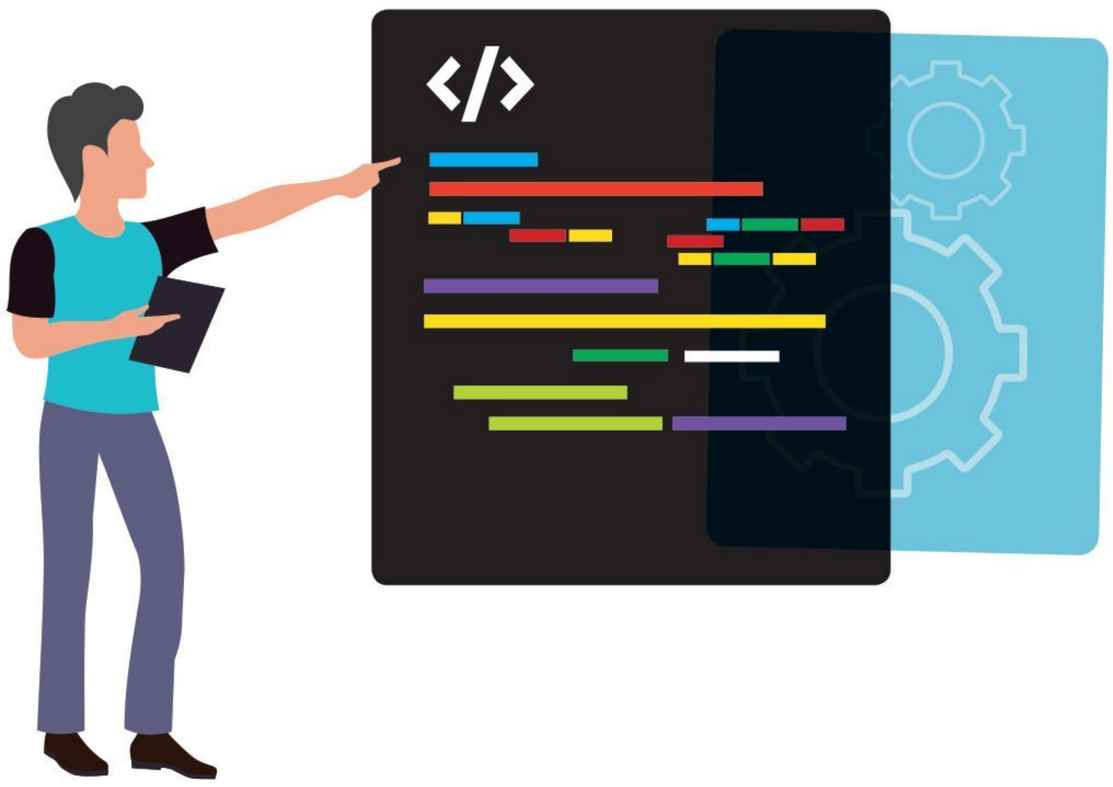
I do hope this article will help you understand the concepts and get started with the development of a single-page web application on Firebase. END

By: Krishna Mohan Koyya

The author is the founder and the principal technology consultant at Glarimy Technology Services, Bengaluru. He has delivered more than 500 programs to corporate clients on design and architecture in the last 12 years. Prior to that, he held various positions in the industry and academia for a decade.

Code Generation: How the Lazy Become Prolific

Code generators bring the fun back into coding. This article lists their benefits and tells how they can be used in different ways, efficiently and effectively.



What does an enthusiastic programmer dream of? Getting to spend every day trying to solve real-world problems using interesting concepts and clever algorithms. And what do we really do as professional developers? Spend hours on a vast set of boring and artificial problems like npm dependency issues.

One of these boring and annoying problems is the creation and maintenance of boilerplate code. That's when we start exploring code generators. They first help us, then bring the fun back in, and transport us to a whole different frontier. What's more interesting? This is a rare case of the lazy becoming prolific!

Let's see what code generators can offer, where they are used, and how to make their use effective and efficient.

Code generators

What exactly do we mean by code generators? Compilers are the most basic kinds of code generators, generating Assembly or machine code from high-level language instructions. There are in-language code generation features too, like macros and generics. However, our focus will be on the kinds like preprocessors and transpilers, which generally convert between high-level languages, and those which generate programs or snippets from the declarative specifications that we give in YAML or JSON.

Code generation or better language?

Isn't code generation like a workaround? Why not pick a supreme language and use it directly instead of generating code in multiple stages?

First, the goal of code generation is not always to overcome language limitations. Just think about auto-generating HTML documentation for your API. You are not trying to overcome the limitation of Go or PHP here, right?

Second, general-purpose languages will always have tradeoffs or weak areas and you'll have to do something about it. Think of Go before generics. Think of powerful languages with unfriendly syntaxes. And even if there is a superior language, what if you are working on an existing codebase?

Also, code generators tend to be problem-specific. Not general-purpose, not even domain-specific, but problem-specific. That makes them so effective.

Finally, who said code generators cannot be languages? There is m4, a macro processor that is Turing-complete, and there is PHP, a templating system, which everybody considers to be more of a programming language.

The advantages of generative programming span from the ease and speed of development to maintainability, consistency, and memory safety and security. They can also be handy with shared code where librarification is hard or less efficient (microservices, maybe?). And then there is fun.

Too much talk. Let's see some examples.

Example: OpenAPI

Consider writing an API server. You not only have to write the handlers, but add code for routing, security checks, parameter extraction, validation, etc, which most of the time don't have anything to do with business logic. But things don't end there. You also have to provide documentation and client SDKs in various languages, keep them updated and in sync.

This is where OpenAPI comes in. With OpenAPI, you specify your API in a YAML or JSON file, and then use different generators to generate boilerplate code, client SDKs, documentation, etc. All you have to write manually is the business logic (well, most of the time).

Figure 1 shows a sample OpenAPI workflow. Here some Go code is auto-generated from the API spec *api.yml*, which is combined with manually written Go code (kept in a separate file) to build the final server binary. The same YAML file can be fed into another tool to generate the HTML documentation.

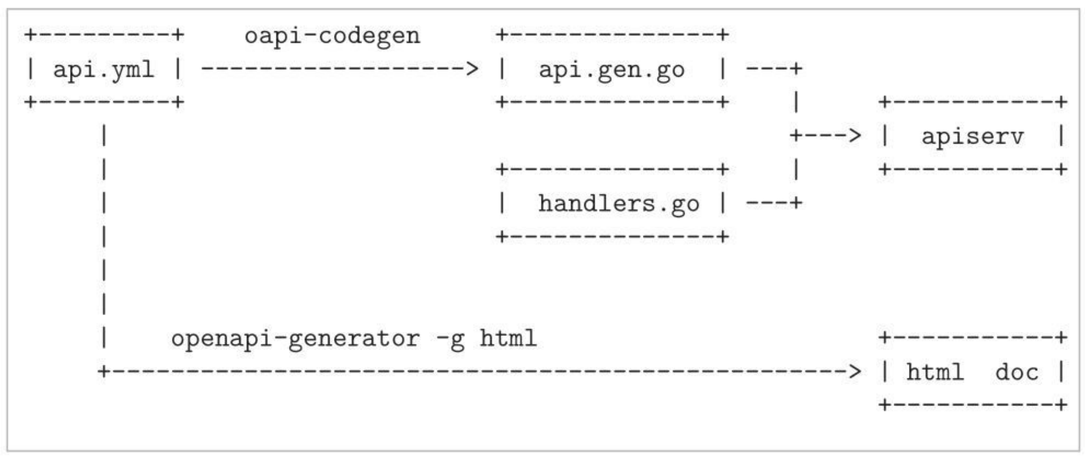


Figure 1: OpenAPI workflow

Here is a part from the YAML input, which is a spec for an API with two endpoints to find the sum and difference of two numbers:

```
/calc/sum:
  get:
    description: Returns the sum of the given numbers.
    parameters:
    - name: x
      in: query
      required: true
      schema:
        type: integer
# “y” needs quoting because y is boolean yes in YAML
    - name: “y”
      in: query
      required: true
      schema:
        type: integer
    responses:
      ‘200’:
        description: The sum.
```

Here are the handwritten handlers in Go, which perform the actual logic:

```
func (calc Calc) GetCalcSum(w http.ResponseWriter,
                            r *http.Request,
                            params GetCalcSumParams) {
  fmt.Fprintf(w, “%d\n”, params.X + params.Y)
}
func (calc Calc) GetCalcDiff(w http.ResponseWriter,
                             r *http.Request,
                             params GetCalcDiffParams) {
  fmt.Fprintf(w, “%d\n”, params.X - params.Y)
}
```

That’s it. The YAML file has 43 lines, and handwritten Go code has less than sixty lines. *oapi-codegen* generated 351 lines that contain parameter extraction, validation, error messages, etc. Here are the auto-generated validation and error messages in action:

```
$ curl ‘localhost:8080/calc/diff?y=12’
parameter “x” in query has an error: \
value is required but missing
$ curl ‘localhost:8080/calc/diff?x=10&y=someString’
parameter “y” in query has an error: \
value someString: an invalid integer: invalid syntax
```

Example: enum stringification in C

Many times we need to print the name of an enum member in C, usually for logging or debugging purposes. But such information is not available to the program at runtime. So we end up writing an array that contains string constants representing the enum members, like this:

```
// XXX Manually keep in sync with the array opstr
typedef enum Operation {
  OP_SUM,
  OP_DIFF,
  OP_LARGEST,
  OP_SMALLEST,
} Operation;
// XXX Manually keep in sync with the enum
const char * opstr[] = {
  “OP_SUM”,
  “OP_DIFF”,
  “OP_LARGEST”,
  “OP_SMALLEST” };
```

Now *opstr[OP_LARGEST]* would give us the string *“OP_LARGEST”*.

But see how we had to put a comment asking future maintainers to keep the blocks in sync? What if they (read: us) missed those comments? That would depend on the mistake. If changes were made in the beginning or middle for either structure, we’d get incorrect logs, without ever realising it. If the enum got a new value *OP_DIV* at the end and the array was never updated, our program would probably crash with a *SEGFAULT* (because *opstr[OP_DIV]* points to invalid memory area now).

Now let’s write a Makefile that invokes *grep* and *sed* to automate the generation of the array.

```
opstr.gen.h: main.c
  (echo ‘// Do not edit this file!’ &&\
    echo ‘const char * opstr[] = { ‘ &&\
    grep -E ‘^\s*OP_[A-Z]+,$$’ main.c|\
```

```
  sed -E 's/^\s*(OP_[A-Z]+),$$/  "\1",/' &&\
echo '};') > opstr.gen.h
```

Once this file is in place, a simple *make* command will make sure the array is generated from the enum if it is missing, or is regenerated if the enum has been updated.

go generate

Go has parallels to both the stringification concept and *make* used in the last example. Since Go has its own build system instead of Makefiles, there is a special annotation called *go:generate* and the *go generate* command to invoke external generators.

Rob Pike, famous programmer and one of the creators of Go, lists its uses in the Go codebase itself in an official blog entry *(go.dev/blog/generate):*

"… generating Unicode tables in the unicode package, creating efficient methods for encoding and decoding arrays in encoding/gob, producing time zone data in the time package, and so on."

One tool used in combination with *go generate* is stringer, whose purpose is similar to the enum stringification we saw above.

In the wild

Code generators are everywhere — from UNIX to Go and k8s ecosystems. Here is a quick listing to give you an idea:

- Compiler compilers: lex, yacc, etc
- Transpilers
- Build system generators: Automake, CMake, qmake, etc
- GUI builders: Glade, Qt Designer, etc
- Configuration generators: update-grub
- In the Web world
  - CSS pre-processors: SaSS, LESS, Stylus
  - k8s: kompose, kustomize, helm, etc
- Interface/binding generators: SIP for Python (used for PyQt, wxPython, etc), PyBindGen, SWIG, etc.
- Flexible: Telosys from Eclipse (DB/model to any kind of code based on templates)
- protoc with gRPC Go plugin
- m4 - a general-purpose macro processor from the 70s, part of UNIX

  I'm yet to try some of the above.

Drawbacks

Code generators are not without drawbacks. To begin with, they can generate truly ugly code. They're so mechanical that not even debuggers like *gdb* can help. Outputs of lex and bison are examples for this. Sometimes this is for better performance. But generators in general can be less efficient too. This is because tools have to be general and won't be able to optimise for individual use cases. (But you can write your own, right?)

Compiler generators are also known for bad error reporting. This could be one reason some major compilers and interpreters use handwritten parsers.

One could argue that the use of generators makes you not know what's happening behind the scenes. But this is not a hard restriction.

The declarative nature of the input to the generators can be an advantage and disadvantage at the same time. Advantage in the sense that it makes sure we have some sort of specification for our APIs, workflows, etc. But the problem is, we might end up ruining those same specs to work around generator limitations.

Some tips

Okay, code generators are helpful. But what can we do on our part to make up for their shortcomings and make the whole process even better? Here are some tips.

- Be aware of the licensing restrictions of the particular tools you use.
- Use tools like *indent* and *gofmt* to reformat the generated code so that it becomes easy to read and debug. This will also get rid of formatting-related compiler warnings.
- Use linters and other static analysis tools to check for code quality and potential bugs.
- Make sure to write and run tests, especially for auto-generated validators. If you're auto-generating tests, check them manually and lock them.
- Although it's an anti-pattern to commit and push generated artefacts, treat auto-generated code like a regular source and include it in the repo. This will be helpful for others trying to build your project on a platform which the generator doesn't support. It'll be helpful to you too when the generator becomes obsolete in future.
- Finally, and most importantly, try to make code generation part of your pipeline, if it makes sense. I'll explain.

Making it part of the pipeline

Is code generation a one-time process or something that is to be done every time you build the software?

For some cases, it doesn't make sense to put the generation in the pipeline. Maybe you are migrating your project from one programming language to another, or choosing a different set of supporting systems so that the configuration files need to be migrated. Here it's clear that you'll be running the converter only once. The original source files are archived, generated files become the new source, and the generator is never run again.

But in some other cases, you are not planning for a migration. Maybe you need something in different formats, or maybe you need boilerplate related to

your handwritten code. In such situations you won't be archiving or throwing away the original source. Here it makes sense to put things in a pipeline. The *Makefile* in our C enum stringification example is an example for this. The source file is checked for updates every time we build the software and the generation commands are run if there is something new.

But what if you had to make manual changes to the generated code and still want it auto-updated whenever the source changes? Let's see an example.

Don't give up on the pipeline: kompose example

kompose is a tool that helps you generate Kubernetes YAML files from Docker Compose files. It is intended to be a one-time process because there is no one-to-one mapping between Docker Compose and Kubernetes. So you are guaranteed to be making changes to the generated files. But what if you want to keep both Docker Compose and Kubernetes versions in parallel, auto-applying changes in the Docker Compose while still keeping your edits to the generated configuration?

I had faced a similar challenge. These were my additions:

1. k8s image pull policy and image pull secret, which were missing in the Docker Compose version
2. Services ignored by kompose because the ports are not exposed in the Docker Compose version
3. Configuration for *initContainers*, which was missing in the Docker Compose version

Instead of choosing to convert once and keep both versions in sync manually, I chose the following solutions, respectively, for each of the above:

1. Use kompose annotations *kompose.image-pull-policy*, *kompose.image-pull-secret* in the Docker Compose file itself, so that kompose knows to add them.
2. Write files for missing services manually.
3. Use a patch file to add the *initContainers* section.

This way, things will be reproducible while still keeping customisations. Of course it doesn't make sense to do this if the differences outnumber similarities.

nguigen

I've been developing desktop applications for more than a decade. Although trivial, they've been used by others, which means I had to regularly update, port and package these apps. *nguigen* (short: *ngg*) is a project that I started as a solution to several difficulties I experienced as part of this.

It is a general-purpose programming language and a code generator that aims to generate code for multiple languages, platforms and GUI toolkits from a single, mostly generic codebase. This is different from solutions like Electron because we're going native here and there is zero overhead. Once mature enough, it could be used to generate C/GTK, C++/Qt, Android and Web apps.

Its C output is already mature and is being used to self-host the compiler (i.e., *ngg* compiler is written in *ngg* itself). *nguigen* offers more expressiveness and compile-time semi-automatic memory management when generating C code. This is important since the C ecosystem remains far superior to many alternatives and one would choose to stick with it.

nguigen is not released yet, but the plan is to release it as free (libre) software. Learn more at *nandakumar.co.in/software/nguigen/*.

Codegen in the nguigen ecosystem

The development of *nguigen* itself depends on code generation. Apart from being written in itself and self-compiled to C, parts of the source are auto-generated from mappings written in TSV (tab-separated values) files.

Makefiles with correct dependencies are auto generated using *ngg* itself and a bash script. This comes handy during major refactorings.

The parser is also auto-generated from grammar specs. Usually lex and bison are used for this purpose, but for *nguigen*, I wrote my own.

And, finally, there is *h2ngg*, a simple tool to auto-generate *ngg* interfaces (like header files) for external libraries like GTK.

Write your own code generators

Maybe you've already noticed it: this talk is not just about existing code generators. It's about the advantages and fun of generative programming, which can only be enjoyed at its maximum if you write your own generators. They benefit from being problem-specific.

How to do that? Maybe you can begin with Makefiles and shell scripts. Use some Python. lex and bison will be overkill, unless you are trying to create a language. END

References

- Slides of the talk: *nandakumar.org/talks/*
- Demo repo: *github.com/nandedamana/lazy-becomes-prolific*

These resources cover several things not covered in this article. The demo repo contains working examples that you can try yourself. It is supposed to be updated also.

By: Nandakumar Edamana

The author is a developer and technology writer who primarily focuses on software freedom, privacy and other ethical issues. He has authored a couple of books and several articles for mainstream publishers. A GNU/Linux user for 15+ years, he releases his software products under free software licences.

Simplified Logging and Analysis with Parseable

Parseable is a free and open source log storage and analysis platform. It is written in the Rust programming language. This article is a quick introduction to the topic.

Written in Rust, Parseable leverages the advances in data compression, storage and networking to bring you a simple but efficient platform.

But before we get into Parseable, let's understand why you should even care about logging.

Why logging?

For us developers, sometimes it is not very clear as to why so much fuss is made about logging. We debug the code, find out the issue, fix it and move on. Why does a log even have to be stored and persisted for a long term?

Log is a loaded term used for a large variety of data. But at a high level, any stream of events that represents the current state of a software or hardware system can be called a log. You can already see the horizon widening now. Consider these scenarios:

- SaaS companies are answerable for any downtime, data access, security breaches, data loss, and so on. In such cases, it makes sense to retain every bit of log data, so that you can not only avoid issues, but also fix them properly if something happens.
- Consumer apps thrive on user behaviour, access patterns and other data. All these are logged and passed through various ML pipelines to generate better responses, notifications, and nudges.
- IoT devices are growing and so is the data generated from these devices. Crunching this data for predictive maintenance and analysis means all this data has to be stored for the long term first.

This is just a small list of possible scenarios where log data is critical for business impact. Now that the need for logging and retention of data for long durations is established, let's take a deeper look at Parseable.

What is Parseable?

Parseable is a free and open source, log storage and analysis platform, released under the AGPL-3.0 licence. Parseable repo is available on GitHub at *https://github.com/parseablehq/parseable*.

The idea behind Parseable is to rethink logging; it's a modern, blazing fast, cloud native (stateless, easily deployable

across cloud providers) platform that is ridiculously easy to deploy and use. Some core concepts on which Parseable is built are listed below.

No need for indexing*:* Traditionally, text search engines like Elastic, etc, have doubled up as log storage platforms. This made sense because log data had to be searched for it to be really useful. But this came at a high cost – indexing is CPU-intensive and slows down the ingestion. Additionally, most of these indexing systems generate index data in the same order of the raw log data. This doubles the storage cost and increases complexity. Parseable changes this. With columnar data formats (parquet), it is possible to compress and query the log data efficiently without indexing it.

Ownership of both data and content: With parquet as the storage format, stored on standard object storage buckets, users not only own their log data but also have unabridged access to the actual content. This means they can simply point their analysis tools like Spark, Presto or TensorFlow to extract more value out of the data. This is an extremely powerful feature, opening up new avenues of data analysis.

Fluid schema: Logs are generally semi-structured by nature, and they are ever evolving, e.g., a developer may start with a log schema like this:

```
{
  “Status”: “Ready”,
  “Application”: “Facebook”
}
```

But as more information gets collected, the log schema may evolve to:

```
{
  “Status”: “Ready”,
  “Application”: {
    “UserID”: “asdadaferssda”,
    “Name”: “Facebook”
  }
}
```

Engineering and SRE teams regularly face schema related issues. Parseable solves this with a fluid schema approach that lets users change the schema on the fly.

SDK-less ingestion: The current ingestion mechanism to logging platforms is quite convoluted, with several protocols and connectors floating around. Parseable aims to make log ingestion as easy as possible. This means you can simply use HTTP POST calls to send logs to Parseable; no complicated SDKs are required. Alternatively, if you want to use a logging agent like FluentBit, Vector, LogStash, etc, almost all the major log collectors already have support for HTTP; hence, Parseable is already compatible with your favourite log collection agent.

Getting started

Let’s see how to get started with Parseable now. We’ll use the Docker image to try out Parseable.

We’ll use Parseable with a publicly accessible object storage, just to help you experience it. Here is the command:

```
cat << EOF > parseable-env
P_S3_URL=https://minio.parseable.io:9000
P_S3_ACCESS_KEY=minioadmin
P_S3_SECRET_KEY=minioadmin
P_S3_REGION=us-east-1
P_S3_BUCKET=parseable
P_LOCAL_STORAGE=/data
P_USERNAME=parseable
P_PASSWORD=parseable
EOF
mkdir -p /tmp/data
docker run \
  -p 8000:8000 \
  --env-file parseable-env \
  -v /tmp/data:/data \
  parseable/parseable:latest
```

You can now log in to the Parseable UI using the credentials we passed here, i.e., ‘parseable’ , ‘parseable’. This will have some data, because Parseable is pointing to the publicly open bucket.

Note that this is a public object storage bucket, which we are using for demo purposes only. Make sure to change the bucket and credentials to your object storage instance before sending any data to Parseable.

To get a deeper understanding of how Parseable works and how to ingest logs, please refer to the documentation available at *https://www.parseable.io/docs/introduction.*

I hope this article helped you get better insights into the logging ecosystem, and that you find Parseable an interesting project to try out and, hopefully, join the community. END

By: Nitish Tiwari

The author is the founder of Parseable. You can find more details at *https://github.com/nitisht.*

More Shell Programming Secrets Nobody Talks About

This is the second part of the article on shell programming secrets. (The first part was carried in the October 2022 issue of *Open Source For You*.) It covers some other important behaviour of bash, particularly that of text expansions and substitutions.

The famed text-processing abilities of the Bourne shell (sh) have been vastly expanded in the Bourne-Again shell (bash). These abilities are efficient to a fault and the code can be even more cryptic. This makes the bash command interpreter and programming language very powerful but also a minefield of errors.

In the last article published in the October 2022 issue of *Open Source For You* titled 'Shell Programming Secrets Nobody Talks About', you learnt that:

- sh and bash are not the same
- *if* statements check for exit codes, not Boolean values
- *[*, *true* and *false* are programs, not keywords
- Presence/absence of space in variable comparisons and assignments makes quite a difference
- *[[* is not the fail-safe version of *[*
- Arithmetic operations are not straightforward as in other languages
- Array operations use cryptic operators
- By default, bash will not stop for errors

I forgot to mention that if you use the *-e* option of the *set* command, as in *set -e*, then you will not be able to check for error codes of previous statements with the *if [$? -eq 0]; then* construct. This option makes bash stop running a shell script if any of its statements encounter an error. If you are doing error handling using *if* statements, then begin your scripts like this:

```
#!/bin/bash
set -u
# The rest of your shell script
```

The *-u* option of the *set* command, if you remember from the last article, ensures that undefined variables are not replaced with an empty string and instead cause an error.

In this article, we will focus on how bash performs text expansions and substitutions. I will only cover what

I think are the most important text-processing features. For comprehensive information, you will have to study the Bash Reference Manual. Bash and several commands such as *sed* and *grep* also use regular expressions to perform text processing. 'Regular expressions' is a separate topic on its own and I will not cover it either.

History expansion character (!)

This feature is available when typing commands at the shell prompt. It is used to access commands stored in the bash history file.

!n	Execute nth command in bash history
!!	Execute last command (Equivalent to !-1)
!leword	Execute last command *beginning* with 'leword'
!?leword?	Execute last command *containing* 'leword'
^search^replace	Execute last command after replacing first occurrence of 'search' with 'replace'

You can modify the history search using certain *word designators*, preceded by a colon (:).

!?leword?:0	Execute with 0th word (usually the command) executable in last command containing 'leword'
!?leword?:2	Execute with second word of last command containing 'leword'
!?leword?:$	Execute with last word in last command containing 'leword'
!?leword?:2-6	Execute with second word to sixth word in last command containing 'leword'
!?leword?:-6	Execute with all words up to 6th word in last command containing 'leword' (Equivalent to !?leword?:0-6)
*!?leword?:**	Execute with all words of last command (but not the 0th word) containing 'leword' (Equivalent to !?leword?:1-$)
*!?leword?:2**	Execute with the second word to the last word in last command (but not the 0th word) containing 'leword' (Equivalent to !?leword?:2-$)
!?leword?:2-	Execute with all words from the 2nd position to last but-one word and not the 0th word in the command containing 'leword'

Remember that bash will execute whatever you have retrieved from the history with whatever you have already typed at the prompt.

You can also use any number of *modifiers*, each preceded by a colon (:).

!?leword?:p	Display (but not execute) last command containing 'leword'
!?leword?:t	Execute with last command containing 'leword' after removing all pathnames of last argument (i.e., leave the tail containing the file name)
!?leword?:r	Execute with last command containing 'leword' after removing the file extension from the last argument
!?leword?:e	Execute with last command containing 'leword' after removing pathname and filename from the last argument (leaving just the extension)
!?leword?:s/search/replace	Execute last command containing 'leword' after replacing the first instance of 'search' with 'replace'
!?leword?:as/search/replace	Execute last command containing 'leword' after replacing all instances of 'search' with 'replace'

If you omit the search text ('leword') and use the history expansion character with the word designators and the modifiers, bash will search the last command. Until you become proficient in using the history expansion character, use the modifier *:p* to display the command before you actually execute it.

Text expansions and substitutions

These features are available at the shell prompt and in shell scripts.

- **Tilde (~)**: In your commands, bash will expand instances of ~ with the value of the environmental variable $HOME, that is, your home directory.
- **? and ***: These are metacharacters. In file descriptors, ? matches any one character while * matches any number of any characters. If they do not match any file names, bash will use their literal values.
- **Brace expansion**: You can use comma-separated text strings within curly brackets to generate combinations of strings with their suffixes and/or prefixes. When I start a new book, I create its folders like this.

```
mkdir -p NewBook/{ebook/images,html/images,image-
sources,isbn,pub,ref}
```

This command creates folders like this.

```
NewBook
NewBook/ref
NewBook/pub
NewBook/isbn
```

```
NewBook/image-sources
NewBook/html
NewBook/html/images
NewBook/ebook
NewBook/ebook/images
```

- **Parameter expansion**: When bash executes a script, it creates these special variables for the script.

Shell variable	Use
$0	Name of the shell script
$1, $2,…	Positional parameters or arguments passed to the script
$#	Total count of arguments passed to the script
$?	Exit status of last command
*$**	All arguments (double-quoted)
$@	All arguments (individually double-quoted)
$$	Process ID of current shell/script

At the terminal, *$0* will usually expand to the shell program *(/bin/bash)*.

On a terminal, you can use the *set* command to *ipso facto* specify parameters to the current shell.

```
# Displays 0
set $?
# Displays an empty string and causes a new line
echo $*
# Sets hello and world as parameters to current shell
set -- hello world
# Displays 2 (the number of parameters)
echo $#
# Displays world
echo $2
# Remove parameters to current shell
set --
# Displays 0 (as earlier)
set $?
```

The option -- represents the end of options and implies that whatever follows it must be command parameters.

- **Command substitution**: Instead of backquotes, you can use the form *$(commands)* to capture the output of those commands for use in some other commands or variables. It makes quoting and escaping much easier.
- **Variable substitution**: You can use these substitutions with command parameters (created by bash for a shell script) or with variables that you have created.

Substitution	Effect
${var1:-var2}	If var1 is null or does not exist, var2 is used
${var1:=var2}	If var1 is null or does not exist, value of var2 is used and set to var1
${var1:?msg}	If var1 is null or does not exist, msg is displayed as error
${var1:+var2}	If var1 exists, var2 is used but not set to var1
${var:offset}	Everything of var after offset number of characters
${var:offset:length}	Length number of characters of var after offset number of characters
${!prefix*}* *${!prefix@}	All variables names beginning with prefix
${!var[@]}* *${!var[*]}	All indexes of array variable var
${#var}	Length of value of var
${var#drop}	Value of var without prefix matching Regex pattern drop
${var##drop}	Empty string if prefix matches Regex pattern drop
${var%drop}	Value of var without suffix matching Regex pattern drop
${var%%drop}	Empty string if suffix matches Regex pattern drop
${var^letter}	Changes first letter of var to uppercase if it matches letter (any alphabet, * or ?) If letter is not specified, all first letter(s) of var will be changed to uppercase
${var^^letter}	Changes any letter of var to uppercase if it matches letter (any alphabet, * or ?) If letter is not specified, all letter(s) of var will be changed to uppercase
${var,letter}	Changes first letter of var to lowercase if it matches letter (any alphabet, * or ?) If letter is not specified, all first letter(s) of var will be changed to lowercase
${var,,letter}	Changes any letter of var to lowercase if it matches letter (any alphabet, * or ?) If letter is not specified, all letter(s) of var will be changed to lowercase
${var/find/replace}	Value of var with instances of *find* replaced with *replace.* If *find* begins with '#', then a match is made at the beginning. A '%' makes it match at the end.

Escaping

You can escape:

- Special characters using the backslash (\). To escape the backslash character, use double backslashes (\\).
- Literal text strings by wrapping them in single quotation marks (‘ ‘). Bash will not perform any expansions or substitutions. The single-quoted string should not have any more single-quotation marks. Bash will not perform any backslash-escaping either.
- Literal text strings by wrapping them in double-quotation marks (“ “) but allowing for
  - $-prefixed variables, expansions and substitutions
  - backslash-escaped characters
  - backquoted (` `) command strings
  - history-expansion characters

```
# Displays Hello World
a=World; echo “Hello $a”
# Displays Hello $a
a=World; echo ‘Hello $a’
# Displays Hello ‘World’
a=World; echo “Hello ‘$a’”
```

Printer’s error

The Bash Reference Manual or even this article may use wrong characters for the quotation marks. The *apostrophe* or *u+0027* used in single-quoted strings may be replaced with the *right single quotation mark* or *u+2019*. The *grave accent* or *u+0060* used in back quoted strings may be replaced with a left *single quotation mark* or *u+2018*. The *quotation mark* or *u+0022* used in double-quoted strings may also be replaced with left and right double quotation marks. They look similar but will result in an error if used in a shell script or in the command line.

I write my books and articles in CommonMark (MarkDown) and output them as HTML, ODT and PDF documents. These documents will not have such errors. When someone edits the document (before it goes to print) in a rich-text editor such as LibreOffice or Microsoft Office, the editor’s autocorrect feature may change ordinary quotation marks and backquotes with inverted quotation marks. Just be aware that this can happen. To avoid mistakes, type the commands by hand. Do not copy-paste them.

Summary

I am sure you will also conclude that bash code can be very cryptic. A lot of production code (industrial-strength shell scripts) is hundreds of lines long. If bash was not so succinct and powerful, it would take forever to write the lines. If you are doing any kind of serious shell scripting, then it is best you know all about bash’s myriad secrets. I think I have covered enough of them to kindle your interest. You are on your own now.

You will find all this and more in my book ‘Linux Command Line Tips & Tricks’. It is free for download from most popular ebook stores. END

By: V. Subhash

The author has written over two dozen books including, *Linux Command-Line Tips & Tricks, CommonMark Ready Reference, Cool Electronic Projects* and *How To Install Solar.*

OSFY Magazine Attractions During 2022-23	
Month	**Theme**
March 2022	Security, Network Management and Monitoring
April 2022	Open Source Programming (Languages and tools)
May 2022	DevOps Special
June 2022	AI, Deep learning and Machine Learning
July 2022	Database management and Optimisation
August 2022	Cloud Special: Everything from management to implementation
September 2022	Mobile/Web App Development, Optimisation and Security
October 2022	Blockchain and Open Source
November 2022	Open Source and IoT and Edge
December 2022	Data science, Data security, Data Management,Storage and Backup
January 2023	Docker and containers
February 2023	Open Source on Windows and Best in the world of Open Source (Tools and Services)

www.IndiaElectronicsWeek.com

India's #1 event for creators of smart solutions based on electronics

INDIA ELECTRONICS WEEK

23-25 NOV, 2022

KTPO, Whitefield, Bangalore

3 WAYS TO GROW YOUR BUSINESS

Explore New Products

that can affect your business positively or negatively

Discover Dealership Opportunities

to expand your product portfolio, your customers and your revenues

Benefit from Deals & Offers

to reduce your costs and identify new vendors

For more information on sponsoring, exhibiting or attending, please call +91-9811155335 or growmybiz@efy.in

R N I No. DELENG/2012/49440, Mailed on 27/28th of Advance month
Published on 27th October 2022
Delhi Postal Regd. No. DL(S)-01/3443/2022-24
Licenced to Post without Pre-Payment Licence No U(S)-56/2022-24
Posted at LPC Delhi RMS Delhi 110006

"I started reading it when I was a student... ...and I am still reading it, as a student"

—CEO, Design House