# IOT FOR ENVIRONMENTAL MONITORING

45 h – 3 VNC – 6 ECTS

**Contacte lecturers:**

Korakot Suwannarat, [1]*School of Engineering and Technology, Walailak University, Thailand*

Suthira Thongkao [1]*School of Language and General Education, Walailak University, Thailand*

**Objectives**

- Knowledge:
  - Basic embedded system design and microcontroller programming
  - How to collect environmental data and the methods to transmit data to to cloud database with real IoT protocols for communication
  - Define the correlation between type and format of environmental data and widget kinds
  - Understand the situation of agricultural production, fertilizer, pesticide, water supply for irrigation, food chain securities and cleaner production;
- Skills:
  - Students will know how to implement the IoT for environmental monitoring system
- Attitude:
  - Participate in discussion group; build case study to apply the knowledge of IoT to the professional field.

**Content of the courses:**

- Chapter 1: Introduction to embedded systems and IoT
  - Lecture 1:
    - Basic for Embedded System
  - Lecture 2:
    - Basic for Internet of Things

- Chapter 2: Microcontroller programming, sensors interfacing and data collection
  - Lecture 3:
    - Microcontroller programming
  - Lecture 4:
    - Example of Environmental Sensors
  - Lecture 5:
    - Data collection methods
  - Lecture 6:
    - Practice embedded system programming and sensors interfacing

- Chapter 3: Cloud Services
  - Lecture 7:
    - Introduction to Cloud Services

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

  - Lecture 8:
    - Practice cloud storage design and implementation

- Chapter 4: Cloud and IoT Integration
  - Lecture 9:
    - Data Transmission from device to cloud
  - Lecture 10-11:
    - Practice integrating Sensor end device to cloud
  - Lecture 12:
    - Practice Using a HUPI cloud computing

- Chapter 5: Environmental Monitoring System Design
  - Lecture 13:
    - Data visualization
    - Dashboard monitoring design
  - Lecture 14-15:
    - Practice the deploying the environmental monitoring

**Academic Materials**

1. E. A. Lee and S. A. Seshia, Introduction to Embedded Systems - A Cyber-Physical Systems Approach, MIT Press, 2017
2. Steve Heath, Embedded Systems Design, Newnes, 2003
3. Rajkumar B., Amir V.D., Internet of Things: Principles and Paradigms, Elsevier, 2016
4. Ryan Batts, Architecting for the Internet of Things, O'Reilly, 2016
5. John Soldatos, Building Blocks for IoT Analytics Internet-of-Things Analytic, River Publishers, 2017

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

# Introduction to embedded systems and IoT

## Basic for Embedded System

### What is an Embedded System?

An embedded system is a small scale computer system that is part of a machine or a larger electrical/mechanical system. It is often designed to perform certain dedicated tasks and often a real-time system. It is called *embedded* because the computer system is embedded within a hardware device. Embedded systems are important, as they are getting increasingly used in many daily appliances, such as digital watches, cameras, microwave ovens, washing machines, boilers, fridges, smart TVs, and cars. Embedded systems also often need to be small in size, low in cost, and have low power consumption. Figure 1 shows the schematic diagram of a typical embedded system that includes a microcontroller, inputs/outputs, and communication interfaces (P.Xiao, 2018).

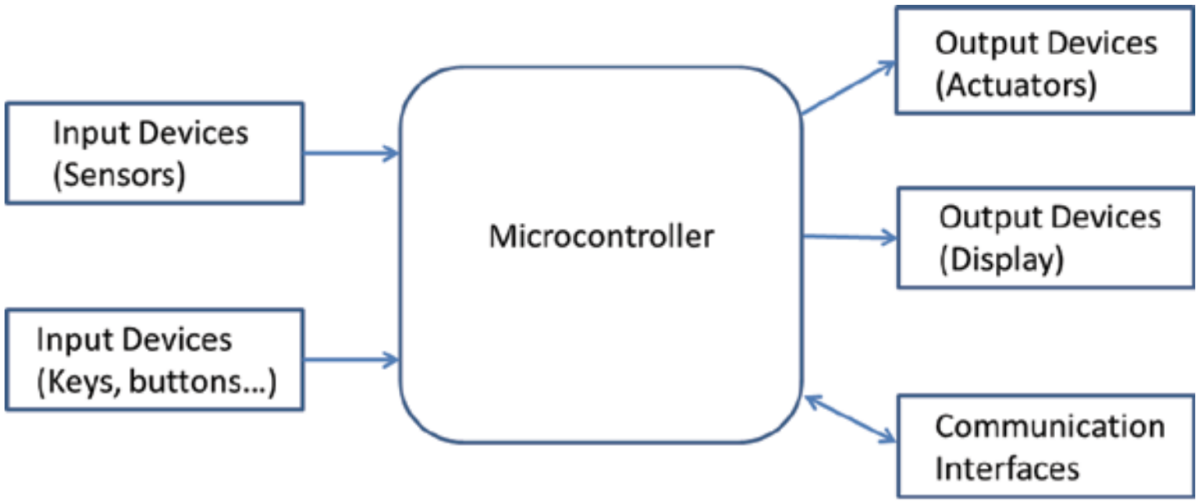



Figure 1 Schematic diagram of a typical embedded system (P.Xiao, 2018).

### Microcontroller

A microcontroller is the brain of an embedded system, which orchestrates all the operations. A microcontroller is a computer processor with memory and all input/output peripherals on it. More details about microcontrollers will be illustrated in the Microcontroller programming, sensors interfacing and data collection section.

### Inputs

An embedded system interacts with the outside world through its inputs and outputs. Inputs can be digital inputs or analog inputs. Inputs are typically used for reading data from sensors (temperature sensor, light sensor, ultrasound sensor, etc.) or other types of input devices (keys, buttons, etc.).

### Outputs

Outputs can also be digital outputs or analog outputs. Outputs are typically used for display, driving motors, or other devices (actuators).

### Communication Interfaces

An embedded system communicates with other devices using communication interfaces, which includes Ethernet, USB (Universal Serial Bus), CAN (Controller Area Network), Infrared, ZigBee, WiFi and Bluetooth, for example.

## Basic for Internet of Things

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

## What is the Internet of Things (IoT)?

The Internet of Things (IoT) refers to the network of physical objects. It is fast growing and already has billions of devices connected. This is different from the current Internet, which is largely a network of computers, including phones and tablets. The "things" in the IoT can be anything from household appliances, machines, goods, buildings, and vehicles to people, animals, and plants. With the IoT, all the physical objects are interconnected, capable of exchanging data with each other without human intervention. They can be accessed and controlled remotely. This is going to completely transform our lives—it will be truly revolutionary (JP.Xiao,2018).

The Internet of Things (IoT) is commonly understood to mean the connection of various objects to the Internet and the use of that connection to enable remote monitoring or control of those objects. This definition of IoT is constrained and only takes into account a portion of its development. It essentially represents a rebranding of the current Machine to Machine (M2M) business (J.Chase, 2013).

Figure 2. The symbolic view of the Internet of Things (IoT).
(Source: https://pixabay.com/en/network-iot-internet-of-things-782707)

Today, the Internet of Things is largely seen as the next big thing, the future of the Internet. According to *Internet Society,* there will be about 100 billion IoT devices and a global market of more than $11 trillion by 2025 (JP.Xiao,2018). The IoT will grow exponentially, just like the Internet did about two decades ago.

## How Does IoT Work?

There are several steps in order to make the Internet of Things (IoT) work.

First, each "thing" on the Internet of Things must have a unique identity. Thanks to the Internet Protocol Version 6 (IPv6) address, the 128bit next generation Internet Protocol (IP) address can provide 2128 different addresses—that is about 6.7 × 1023 addresses per square meter. We should be able to assign a unique ID to every physical object on the planet.
Second, each "thing" must be able to communicate. There are a number of modern wireless technologies which make communications possible, such as WiFi, Bluetooth Low Energy, Near-field communication (NFC), RFID, as well as ZigBee, Z-Wave, and 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks), etc.

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

Third, each “thing” needs to have sensors so that we can get information about it. Sensors can be temperature, humidity, light, motion, pressure, infrared, ultrasound sen- sors, etc. The new sensors are increasingly getting smaller, cheaper, and more durable.

Fourth, each “thing” needs to have a microcontroller (or microprocessor) to manage the sensors and communications, and to perform the tasks. There are many microcontrollers exist that could be used for IoT.

Finally, we will need cloud services to store, analyze, and display data so that we can see what’s going on and take action via phone apps. There are already a lot of big companies working on this, such as IBM’s IBM Watson, Google’s Google Cloud Platform, Microsoft’s Azure, and Oracle’s Oracle Cloud etc.

# Microcontroller programming, sensors interfacing and data collection

## Microcontroller programming

### THE IDE

When it comes to writing your code, the IDE (integrated development environment) is where you’re going to have the most choices available to you. As a matter of fact, it’s kind of a dirty little secret among programmers that you don’t need to use a fancy development environment like Microsoft’s Visual Studio or Eclipse at all. You can write perfectly good, functional C code with any standard text editor, like emacs, vim, Sublime Text, or even Windows’ Notepad or Mac’s TextEdit.

Important subjects in microcontroller programming include microcontroller architecture, data types, if/then/else conditions, loops, GPIO, Interfacing, and interrupts.

## Sensors

### What are sensors?

Sensors are electrical components that function as input devices. Not all inputs are explicitly sensors, but almost all inputs use sensors! Consider your computer mouse or trackpad, a keyboard, or even a webcam; these are not sensors, but they definitely use sensors in their de- sign. More abstractly, you can frame sensors as a component to measure a stimulus that is external to the system it is in (its environment). The output data is based on the measurement. For example, when you type at a keyboard, the letter that appears on your screen (the output) is based on the measurement (which switch, or key, you pressed on the keyboard). How many letters appear on screen is based on another measurement (how long you keep the key pressed). Figure 3 illustrates the use of environmental sensors, such as an infrared sensor and an ultrasonic sensor, to measure distance. A flex sensor can be used to evaluate an object's curvature. Photoresistors, also known as light dependent resistors (LDR), are light-sensitive devices that are commonly used to detect the presence or absence of light or to quantify the intensity of light.

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

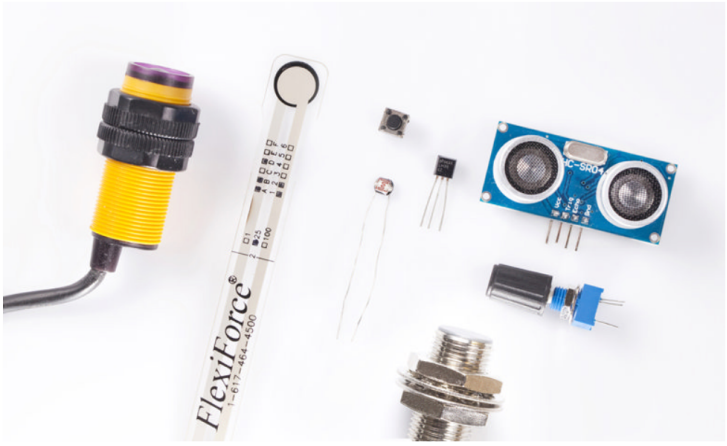



Figure 3. Examples of Sensors

## Data collection methods

Aside from traditional communication technologies like Ethernet, WiFi, and Bluetooth, there are several more technologies that may be utilized for Internet of Things connections. In IoT systems, end devices are frequently deployed remotely. Furthermore, certain places lack a power source for all end devices. For today's modern IoT applications, LPWAN communication is prevalent. Table 1 shows the kinds of wireless communication that are used in the IoT. In short-range communication, LiFi, WiFi, cellular, and Bluetooth are employed. All of them, however, have the benefit of being able to send data at a high data rate. Wide Area Network (WAN) networks include Z-bee, Z-wave, and LoRa. LoRa, in particular, can transfer data up to 15km but has a limited data transmission rate.

Table 1. Wireless technologies in IoT applications (P.Xiao, 2018)].

| | Standard | Frequency | Range | Data Rate |
|---|---|---|---|---|
| **LiFi** | Similar to 802.11 | 400–800 THz | <10 m | <224 Gbps |
| **WiFi** | 802.11a/b/g/n/ac | 2.4 GHz and 5 GHz | ~50 m | <1 Gbps |
| **Cellular** | GSM/GPRS/EDGE (2G), UMTS/HSPA (3G), LTE (4G), 5G | 900, 1800, 1900, and 2100 MHz<br>2.3, 2.6, 5.25, 26.4, and 58.68 GHz | <200 km | <500 kps (2G),<br><2 Mbps (3G),<br><10 Mbps (4G)<br><100 Mbps (5G) |
| **Bluetooth** | Bluetooth 4.2 | 2.4 GHz | 50–150 m | 1 Mbps |
| **RFID/NFC** | ISO/IEC 18000-3 | 13.56 MHz | 10 cm | 100–420 kbps |
| **6LowPAN** | RFC6282 | 2.4 GHz and ~1 GHz | <20 m | 20–250 kbps |
| **ZigBee** | ZigBee 3.0 based on IEEE802.15.4 | 2.4 GHz | 10–100 m | 250 kbps |
| **Z-Wave** | Z-Wave Alliance ZAD12837 / ITU-T G.9959 | 868.42 MHz and 908.42 MHz | <100 m | <100 kbps |
| **LoRa** | LoRaWAN | 868 MHz and 915 MHz | <15 km | 0.3–50 kbps |

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

The Internet of Things networks can be divided into two categories based on the transmission data rate.

a. Low transmission data rate
   The low transmission rate is suitable for Internet of Things applications in which the end devices upload the message data, such as NB-IoT, LoRa, and Sigfox...
b. High transmission data rate
   For some IoT applications to send large amounts of data, like pictures or videos, a high transmission data rate is required.

## Network Protocols in the IoT

Protocols, or communication protocols, are a set of rules that allow devices to communicate with each other. Protocols define the syntax, semantics, and synchronization of communication. A close analogy to protocols is human languages. There are many communication protocols available for IoT applications. The following are commonly used protocols: HTTP, Websocket, and MQTT(P.Xiao, 2018).

**HTTP**
The Hypertext Transfer Protocol (HTTP) is the communication protocol behind the World Wide Web (WWW). It is based on client–server architecture, and operates in a request and response fashion. HTTP uses TCP (transmission control pro-tocol) to provide reliable connections.

**WebSocket**
WebSocket is a communication protocol designed for web browsers and web servers, but unlike HTTP, WebSocket provides full-duplex communication over a single TCP connection. WebSocket is stateful, as the client and server do maintain a connection during the communication. The WebSocket makes more interaction between a browser and a web server possible, enables real-time data transfer and streams of messages.

**MQTT**
MQ Telemetry Transport (MQTT) is a lightweight, machine-to-machine communication protocol designed for IoT devices by IBM. MQTT is based on a publisher–subscriber model, where the publisher publishes data to a server (also called broker), and the subscriber subscribes to the server and receives data from the server. The MQTT broker is responsible for distributing messages and can be somewhere in the Clouds.

**CoAP**
The Constrained Application Protocol (CoAP) is a specialized application layer proto-col for constrained IoT devices, i.e., devices with limited computing power, power consumption, and network connectivity, etc. It is based on request and response messages, similar to HTTP, but it uses UDP (user datagram protocol)rather than TCP (transmission control protocol). Although UDP does not provide reliable transmissions, it is much simpler, has much smaller overhead, and hence it is much faster. CoAP is designed for machine-to-machine (M2M) applications such as smart energy and home / building automation.

**XMPP**
Extensible Messaging and Presence Protocol (XMPP) is an open standard, real-time communication protocol based on XML (Extensible Markup Language). It can provide a wide range of services including instant messaging, presence and collaboration. It is decentralized and has security features. It is also extensible, which means it is designed to grow and accommodate changes. XMPP software includes servers, clients, and libraries.

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

# Cloud Services

## Understanding Serverless Computing

Serverless architecture encompasses many things, and before jumping into creating serverless applications, it is important to understand exactly what serverless computing is, how it works, and the benefits and use cases for serverless computing. Generally, when people think of serverless computing, they tend to think of applications with back-ends that run on third-party services, also described as code running on ephemeral containers. Many businesses and people who are new to serverless computing will consider serverless applications to be simply "in the cloud." While most serverless applications are hosted in the cloud, it's a misperception that these applications are entirely serverless. The applications still run on servers that are simply managed by another party. Two of the most popular examples of this are AWS Lambda and Azure functions. We will explore these later with hands-on examples and will also look into Google's Cloud functions[9].

## 1. What Is Serverless Computing?

Serverless computing is a technology, also known as function as a service (FaaS), that gives the cloud provider complete control over the container the functions run on as necessary to serve requests. By doing so, these architectures remove the need for continuously running systems and serve as event-driven computations. The feasibility of creating scalable applications within this architecture is huge. Imagine having the ability to simply write code, upload it, and run it, without having to worry about any of the underlying infrastructure, setup, or environment maintenance. The possibilities are endless, and the speed of development increases rapidly. By utilizing the serverless architecture, you can push out fully functional and scalable applications in half the time it takes you to build them from the ground up[9].

## Cloud Services

Cloud services can be classified into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS)[10].

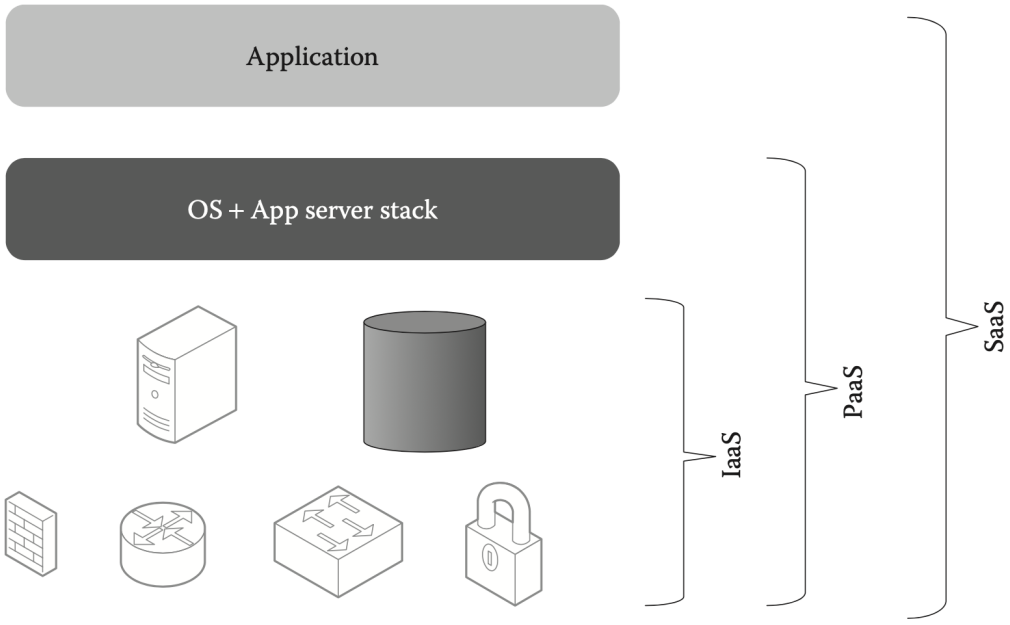



Figure 4. Cloud services models[9].

• IaaS provides customers with the capability to rent hardware components such as CPUs, storage, and networks. The customers are allowed to run their selected OS and applications on the

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

hardware components. The customers pay for hardware components usage, such as CPU usage, storage usage, and network usage. Amazon EC2 is an example of IaaS[10].

• PaaS allows customers to use cloud-provided programming tools to develop their applications and deploy them on the PaaS platform. Elasticity and scalability of the application are guaranteed by the PaaS platform. The customers cannot control the underlying hardware components. Customers pay only for the platform software components, such as databases, OS, and middleware, which include its associated hardware costs. Microsoft Azure and Google App Engine are examples of PaaS.

• In the SaaS model, an application, such as email and office software, is offered as a service by the cloud provider. Customers can access the service by using various devices through a thin client interface such as a web browser. The cloud provider hosts and manages the required software and hardware to support the service. The customers pay a subscription fee for the usage of the service. SaaS reduces the need to deploy on-premise applications, which are usually expensive. It also reduces the need for manual updates because the SaaS pro- viders can perform those tasks automatically. Microsoft Outlook 365 and Google Docs are examples of SaaS[10].

# Cloud and IoT Integration

## Wireless Sensor Networks (WSN)

A WSN can be defined as a network of devices, denoted as nodes, which can sense the environment and communicate the information gathered from the monitored field (e.g., an area or volume) through wireless link. The data is forwarded, possibly via multiple hops, to a sink (sometimes denoted as controller or monitor) that can use it locally or is connected to other networks (e.g., the internet) through a gateway. The nodes can be stationary or moving. They can be aware of their location or not and they can be homogeneous or not.

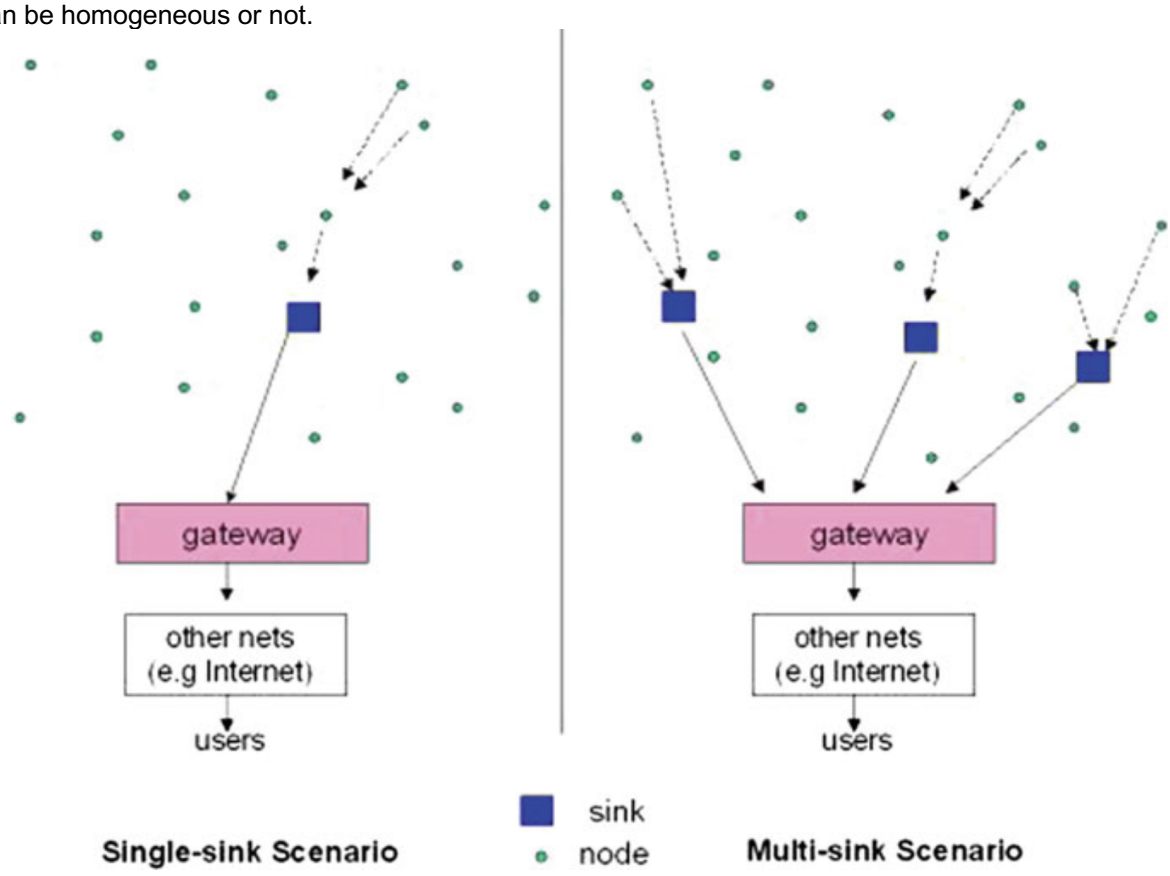



Figure 5 Left part: single-sink WSN. Right part: milti-sink scenario

This is a traditional single-sink WSN (see Figure 5, left part). Almost all scientific papers in the literature deal with such a definition. This single-sink scenario suffers from the lack of scalability: by increasing the number of nodes, the amount of data gathered by the sink increases, and once its capacity is

reached, the network size cannot be augmented. Moreover, for reasons related to MAC and routing aspects, network performance cannot be considered independent of the network size.

A more general scenario includes multiple sinks in the network (see Figure 5, right part). Given a level of node density, a larger number of sinks will decrease the probability of isolated clusters of nodes that cannot deliver their data owing to unfortunate signal propagation conditions. In principle, a multiple-sink WSN can be scalable (i.e., the same performance can be achieved even by increasing the number of nodes), while this is clearly not true for a single-sink network.

However, a multi-sink WSN does not represent a trivial extension of a single- sink case for the network engineer. In many cases, nodes send the data collected to one of the sinks, selected among many, which forwards the data to the gateway, toward the final user (see Figure 5, right part). From the protocol viewpoint, this means that a selection can be done, based on a suitable criteria that could be, for example, minimum delay, maximum throughput, minimum number of hops, etc. Therefore, the presence of multiple sinks ensures better network performance with respect to the single-sink case (assuming the same number of nodes is deployed over the same area), but the communication protocols must be more complex and should be designed according to suitable criteria.

## Data Transmission from device to cloud

IoT platforms connect the sensors and data network to one another, integrating with backend applications to provide insights using backend applications to make sense of the plethora of data generated by hundreds of sensors. With IoT platforms, you can connect and monitor your devices and sensors, display and analyze sensor data, control your devices, and develop software applications for your devices. The following is a list of commonly used IoT platforms

### AWS IoT

Amazon's AWS IoT platform provides secure communications between IoT devices and AWS. AWS IoT supports HTTP, WebSockets, and MQTT. Figure 6 shows the schematic diagram of Amazon's AWS IoT platform from the Amazon website. Its Rules Engine can route messages to AWS endpoints, including AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch, and Amazon Elasticsearch Service with built-in Kibana integration. It can also create a persistent, virtual version, or "shadow," of each device that includes the device's latest state, so that users can interact with devices even when they are offline. Additional AWS IoT information is available at https://aws.amazon.com/iot/.

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

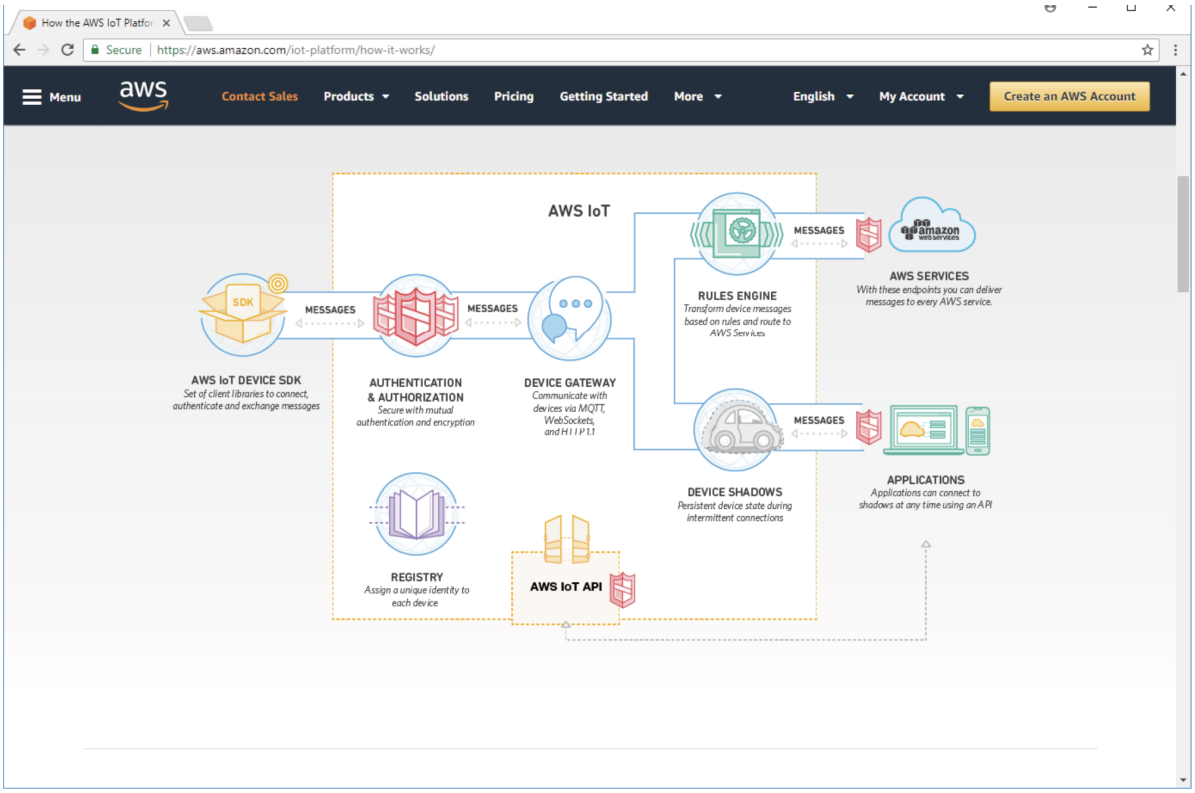



Figure 6. AWS IoT platform (Source: https://aws.amazon.com/iot/)

## Microsoft Azure IoT Suite

Microsoft Azure IoT Suite can be easily integrated with your systems and applications, including Salesforce, SAP, Oracle Database, and Microsoft Dynamics. It packages together Azure IoT services with preconfigured solutions. The Azure IoT Suite supports HTTP, Advanced Message Queuing Protocol (AMQP), and MQTT. A set of device SDKs for .NET, JavaScript, Java, C and Python are available. Figure 7 shows the schematic diagram of the Microsoft Azure IoT solution architecture. Additional Microsoft Azure IoT Suit information is available at https://azure.microsoft.com/en-us/suites/ iot-suite/

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

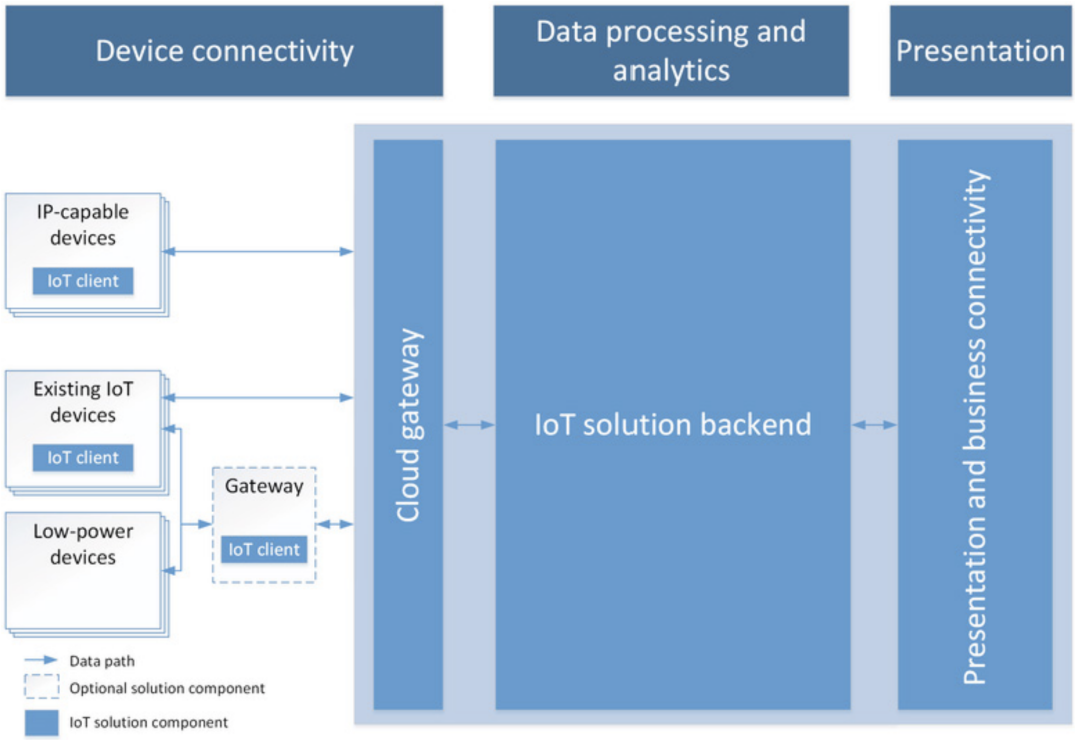



Figure 7. Microsoft Azure IoT architecture. (Source:https://azure.microsoft.com/en-us/suites/ iot-suite/)

## Google Cloud IoT

Google Cloud IoT takes advantage of Google's heritage of web scale processing, analytics, and machine intelligence. It utilizes Google's global fiber network (70 points of presence across 33 countries) for ultra low latency. Software libraries are available for Go, Java (Android), .NET, JavaScript, ObjectiveC (iOS), PHP, Python, and Ruby. Figure8 shows the schematic diagram of the Google Cloud IoT platform. Additional Google Cloud IoT information is available at https://cloud.google.com/solutions/iot/

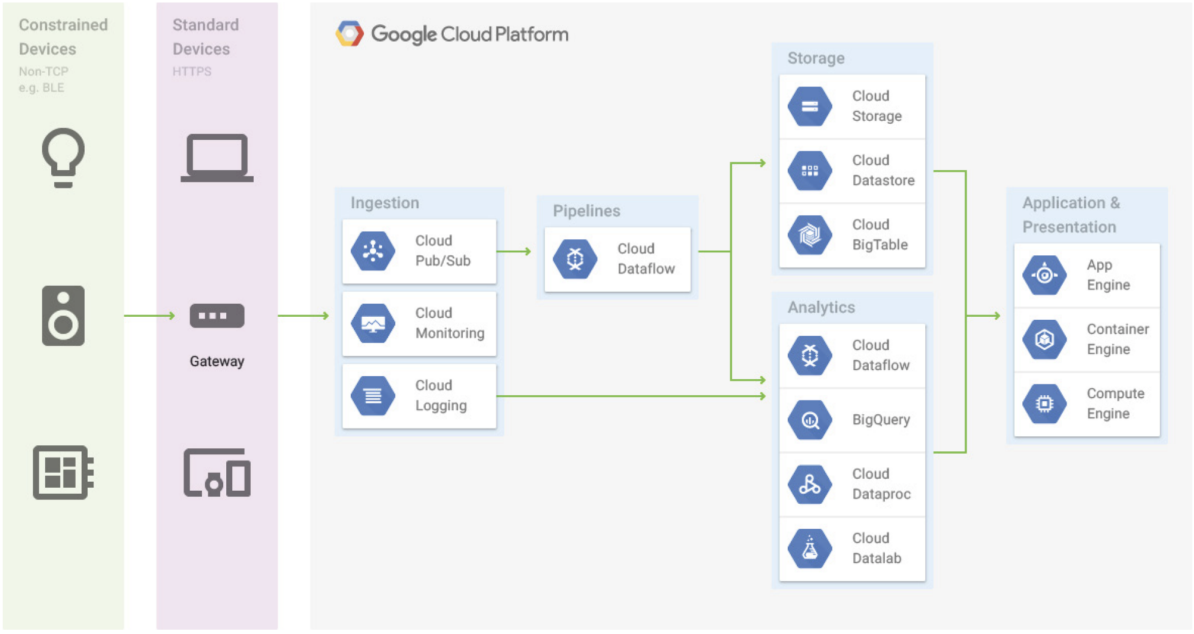



Figure 8. Google Cloud IoT Platform. (Source: https://cloud.google.com/solutions/iot)

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

# Environmental Monitoring System Design

## Data visualization

When it comes to gaining valuable insight in a company setting, the use of data visualization is critical. Companies are desperate to view and learn from their Big Data. Data visualization, however, is a growing field with a critical shortage of true experts(M.Yuk and S.Diamond, 2014).

Big Data refers to the voluminous amounts of infor- mation that can be collected from social media data as well as internal company data. Analyzing and extracting insights from it is the goal.

Here's a simple definition of data visualization: It's the study of how to represent data by using a visual or artistic approach rather than the traditional reporting method.

Two of the most popular types of data visualization are dashboards and infographics, both of which use a combination of charts, text, and images to communicate the message of the data. The practice of transforming data into meaningful and useful information via some form of visualization or report is called Business Intelligence (BI)(M.Yuk and S.Diamond, 2014).

Data visualizations (you can call them data viz for short) are widely used in companies of all sizes to communicate their data stories. This practice, known as BI, is a multibillion-dollar industry. It continues to grow exponen- tially as more companies seek ways to use their big data to gain valuable insight into past, current, and future events.

## Dashboard monitoring design

### Choosing simple and effective charts

Although you have many chart types to choose among, we recommend starting with some of the simple and most commonly used charts for the most chance for success: bar and column charts, line charts, and pie charts. No doubt you're familiar with them and have seen many examples. In the following sections, we discuss these chart types and show you when to use them(M.Yuk and S.Diamond, 2014).

### Bar and column charts

Some people use the term bar chart when speaking about a chart that shows the data horizontally or vertically; others call a chart that displays the data vertically a column chart. Whatever you call them, these charts are best used for comparison (M.Yuk and S.Diamond, 2014).

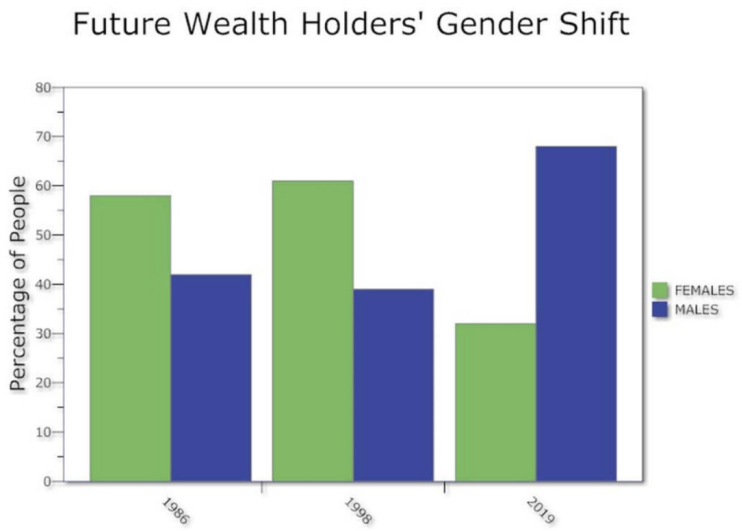



Figure 9. An example of a bar chart(M.Yuk and S.Diamond, 2014).

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

When you use a column chart, be sure to shorten or use smaller labels on your x-axis below each bar to ensure they display horizontally. Utilizing longer labels will result in the need to display the title vertically, which is hard for the user to read.

## Line charts

A line chart (Figure 10) connects data points over a period of time. Line charts are best used for something like a trend to show movement. These charts are easy to read and fairly easy to create. This type of chart should be one of your staples.

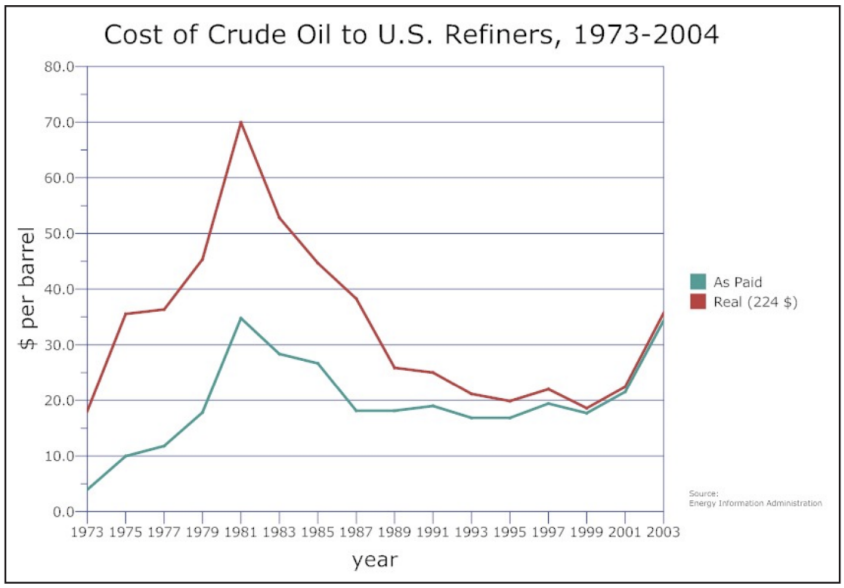



Figure 10. An example of a line chart

## Pie charts

The use of pie charts is controversial, and the debate is more than a decade old. Just type the words avoid pie chart in a search engine, and you'll literally find more than 1 million entries. One of the best-known data design experts, Edward Tufte, refers to pie charts as "dumb" in his book The Visual Display of Quantitative Information (Graphics Press). Tufte argues that pie charts are dumb because they fail to show comparisons and trends as well as bar or line charts do. Many experts argue that the eyes are not good at estimating areas, which you must do when viewing a pie chart.

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

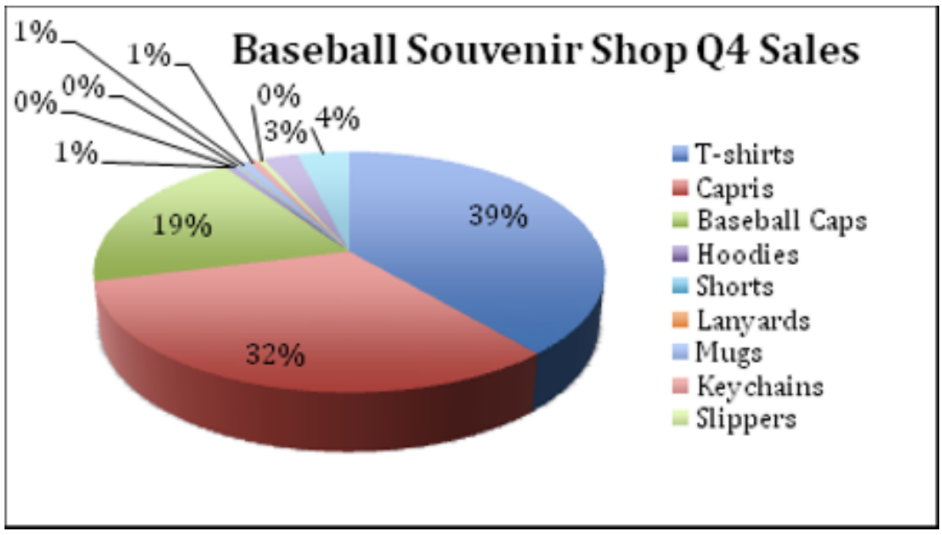



Figure 11. An example of a pie chart

## Using gauges and scorecards to monitor

In data visualizations, gauges are often used to monitor the status of key per- formance indicators or something with known data parameters. If you know the lowest and highest measurements, you can use tick marks to display them and use a pointer to show where the data is at the present time.

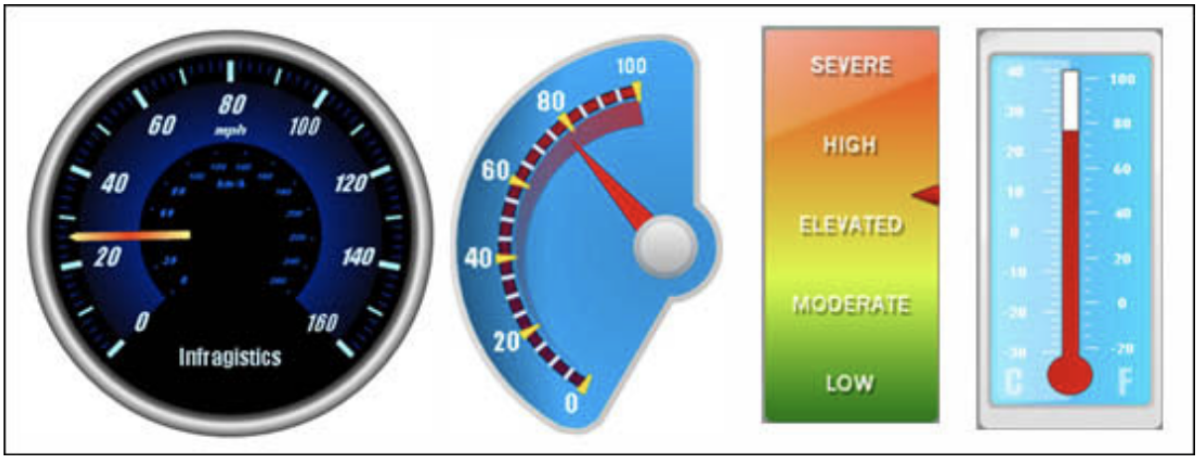



Figure 12. An example of a gauge chart

## Finding online tools for chart making

When you're creating your first chart, you'll probably use Microsoft Excel. But you can use many online tools to accomplish the same task. Here are a few that you may want to consider:

✓ Rich Chart Live (www.richchartlive.com/RichChartLive): Available in both free and fee-based versions

✓ ChartGo (www.chartgo.com/index.jsp): Free

✓ ChartGizmo (http://chartgizmo.com): Free

✓ Online Chart Tool (www.onlinecharttool.com): Free

## An Example of an Environmental Dashboard

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

The first example is the air quality monitoring system that used LabVIEW to create the system. It is not a cloud monitoring system. The dashboard show in Figure 13, the creators used gage with a color to present the level of temperature and used line graphs to compare 2 sources of temperature[8].

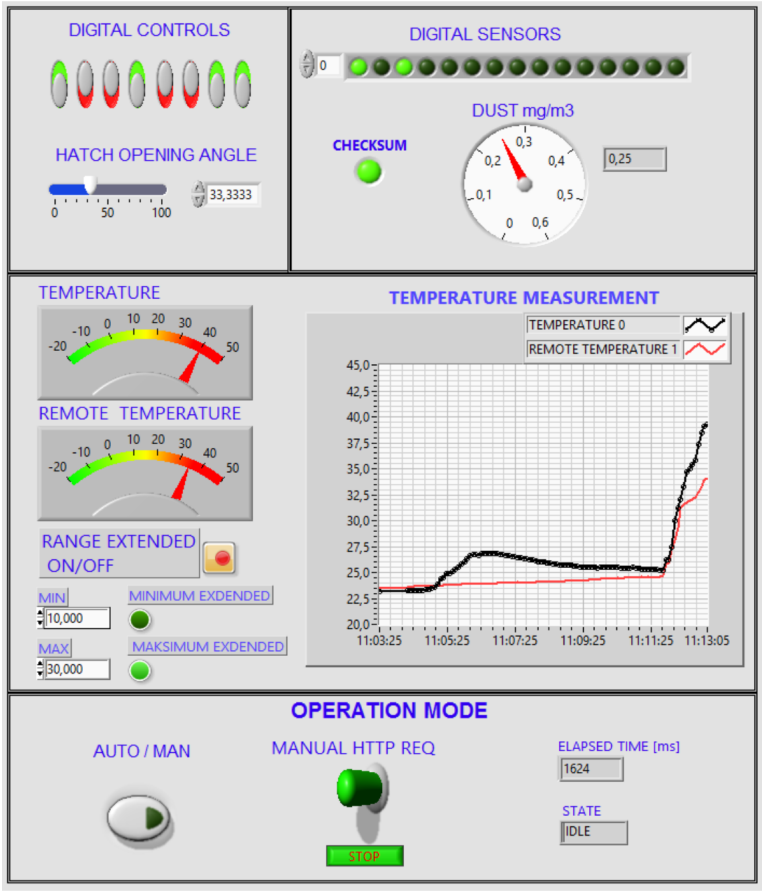
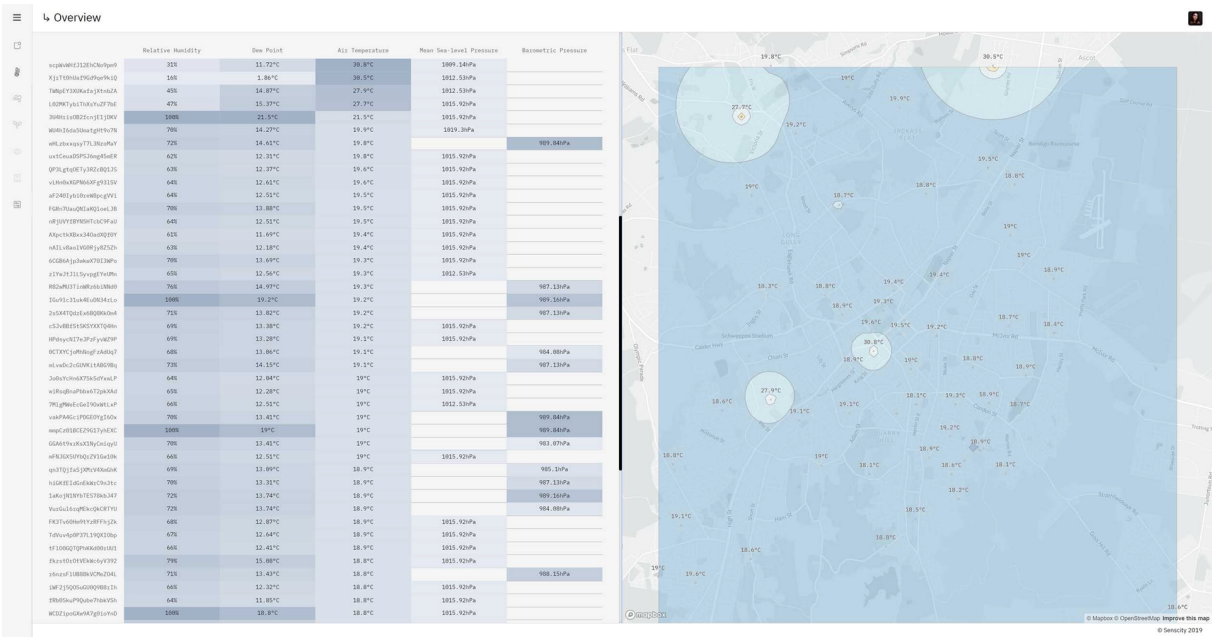



Figure13. The real-time monitoring dashboard created with LabVIEW[8]

The second is that the SensCity platform was produced by the startup company in Australia. Figure 14 (a) shows the line graph and location of the sensor station and figure 14 (b) provides the air parameter in the table and shows the location of the sensor that was installed.

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*


(a)




(b)

Figure 14. The example dashboard on SensCity (Source:https://senscity.com.au)

## REFERENCE

1. E. A. Lee and S. A. Seshia. (2017), Introduction to Embedded Systems - A Cyber-Physical Systems Approach, MIT Press.
2. S.Heath (2013), Embedded Systems Design, Newnes.

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*

3. Rajkumar B., Amir V.D. (2016), Internet of Things: Principles and Paradigms, Elsevier.
4. R.Batts. (2016), Architecting for the Internet of Things, O'Reilly.
5. J.Soldatos (2017), Building Blocks for IoT Analytics Internet-of-Things Analytic, River Publisher.
6. M.Yuk and S.Diamond. (2014), Data Visualization For Dummies, John Wiley & Sons, Inc.
7. J.Chase. (2013), The Evolution of the Internet of Things, Texas Instruments.
8. B. Palczynska and D. Rabczuk. (2018). Low-Cost Embedded Control System for Environmental Monitoring. *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*,1-5, doi: 10.1109/EEEIC.2018.8493727.
9. M.Strigler. (2018), Beginning Serverless Computing, Apress.
10. R.Fox and W.Hao. (2018). Internet Infrastructure Networking, Web Services, and Cloud Computing, CRC Press.
11. P.Xiao. (2018). Designing Embedded Systems and the Internet of Things (IoT) with the ARM Mbed, WILEY.

(JP.Xiao,2018)

*This document was produced within the MONTUS project financed by the European Union in the framework of Erasmus + Capacity Building 598264-EPP-1-2018-1-FR-EPPKA2-CBHE-JP.*