IT4735
IoT VÀ ỨNG DỤNG
(IoT and Applications)

Phạm Ngọc Hưng, Nguyễn Đình Thuận, Đặng Tuấn Linh
Faculty of Computer Engineering
School of Information and Communication Technology (SoICT)
Hanoi University of Science and Technology
E-mail: [hungpn, thuannd, linhdt]@soict.hust.edu.vn

Giới thiệu học phần

- IT4735 IoT và Ứng dụng (IoT and Applications)
- Credits: 2 (2-1-0-4)
- Đánh giá:
  - Quá trình: 40%, Bài tập tuần, Bài tập Project
  - Cuối kỳ: 60%, Bài tập Project, báo cáo, thuyết trình, vấn đáp, **thi**

Mục tiêu

- Kiến thức tổng quan về IoT
- Kiến trúc, mô hình phân lớp, các thành phần của hệ thống IoT
- Các công nghệ IoT: thiết bị và cảm biến, một số chuẩn truyền thông IoT, các giao thức truyền thông cho ứng dụng IoT, nền tảng đám mây
- Lập trình cho thiết bị, hệ thống IoT
- Bảo mật trong IoT
- Vận dụng kỹ năng thiết kế và xây dựng một hệ thống ứng dụng IoT

Nội dung

- Chương 1. Tổng quan về IoT
- Chương 2. Các công nghệ IoT
- Chương 3. Lập trình ứng dụng IoT
- Chương 4. An toàn và Bảo mật IoT
- Chương 5. Thiết kế và xây dựng hệ thống IoT

Chương 1. Tổng quan về IoT

- 1.1. Tổng quan về IoT
- 1.2. Các công nghệ IoT
- 1.3. Kiến trúc hệ thống IoT
- 1.4. Các ứng dụng IoT
- 1.5. Các thách thức của IoT

1.1. Tổng quan về IoT

- **What is the Internet of Things (IoT) ?**
  - *IoT is the network of things, with clear element identifcation, embedded with software intelligence, sensors, and ubiquitous connectivity to the Internet*
  - *(Book: Internet of Things From Hype to Reality)*
  - *Components:*
    - *Sensors*: to collect information.
    - *Identifiers*: to identify the source of data (e.g., sensors, devices).
    - *Software*: to analyze data.
    - *Internet connectivity*: to communicate and notify.
- "Things" = "anything", "everything"
  - Home appliances, building, car, people, animals, trees, plants, ...

IoE = Internet of Everything (by Cisco)

Tổng quan về IoT

Tổng quan về IoT

- **How does IoT works ?**
  - **IoT devices:** smart devices (that use embedded systems such as processors, sensors and communication hardware) to collect, send and act on data they acquire from their environments.
  - **IoT devices** share the sensor data they collect by connecting to an **IoT gateway**, which acts as a central hub where IoT devices can send data.
  - Before the data is shared, it can also be sent to an **edge device** where that data is analyzed locally. Analyzing data locally reduces the volume of data sent to the cloud, which minimizes bandwidth consumption.

Tổng quan về IoT

- **How does IoT works ?**
  - Sometimes, IoT devices communicate with other related devices and act on the information they get from one another.
  - The devices do most of the work without human intervention, although people can interact with the devices -- for example, to set them up, give them instructions or access the data.
  - The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.
  - **IoT can also use artificial intelligence** and machine learning to aid in making data collection processes easier and more dynamic.

Tổng quan về IoT

- An example of IoT smart home solution

Tổng quan về IoT

- **Why is IoT important ?**
  - IoT helps people live and work smarter.
    - For example: IoT-embedded devices -- such as cars, smartwatches or thermostats -- improve people lives.
  - IoT is essential to business:
    - Real-time supervising of products/objects/system
    - Delivering insights into everything from the performance of machines to supply chain and logistics operations.
  - IoT enables machines to complete tedious tasks without human intervention.
    - automate processes, reduce labor costs, cut down on waste and improve service delivery.
    - less expensive to manufacture and deliver goods,
    - transparency into customer transactions.

Tiến hóa của IoT

- The field of IoT has grown tremendously
- 1999: Kevin Ashton (Auto-ID Center, MIT) first mentioned the internet of things in a presentation about RFID

(Source: IHS)

Tiến hóa của IoT

- IoT has evolved from the convergence of:
  - Wireless technology
  - Microelectromechanical systems
  - Microservices
  - Internet
- M2M (machine to machine) communication, evolved to next level
- 2010: broder consumer use (smart phones, smart TV)
- 2020: cellular IoT (2G/3G/4G/5G. LoRaWAN, LTE)
- 2023: billions of internet-connected devices
  - **Digital twins:** virtual representation of a real-world entity or process

Tiến hóa của IoT

1. Pre-Internet: https://en.wikipedia.org/wiki/History_of_the_telephone

2. Internet of content (WWW, 1989, Tim Berners-Lee)
https://en.wikipedia.org/wiki/History_of_the_World_Wide_Web

3. Internet of services (Web 2.0, Yahoo, Amazon, … ~2000, dotcom companies)

4. Internet of people (smart phones, social networks, iPhone1 2007)

Internet of Things named 1999 https://iot-analytics.com/internet-of-things-definition/

Tiến hóa của IoT

Idea: Move from Internet of People → Internet of Things

- Internet xuất hiện ở khắp nơi trên thế giới
- Ban đầu là để kết nối con người – con người

- Internet of Things kết nối mọi thứ (“things”) sử dụng các phương tiện hạ tầng đã có.

Tiến hóa của IoT

IoT: Human connecting with Things

Tiến hóa của IoT

IoT: Things connecting with Things

- Complex and heterogeneous resources and networks

4th IR and IoT

- The Fourth Industrial Revolution and IoT
  - 1st IR: transformed society with the introduction of machines and mechanized production.
  - 2nd IR: introduced electricity, which led to mass production.
  - 3rd IR: has been called the dawn of the information age.
  - 4th IR: as "the fusion of technologies that is blurring the lines between the physical, digital, and biological spheres.“ (by *Klaus Schwab*)
- IoT is being called a major driver of the Fourth Industrial Revolution. Why?

1.2. Kiến trúc tổng quan hệ thống IoT

(1) (2) (3)

(1) Kiến trúc đơn giản: Các thiết bị kết nối trực tiếp đến server/cloud

(2) Kiến trúc phân cấp: Các thiết bị kết nối qua tầng trung gian (Fog/Edge node, gateway)

(3) Kiến trúc tương lai: “Things” kết nối trực tiếp “Things”

Kiến trúc phân tầng của hệ thống IoT (1)

Kiến trúc phân tầng của hệ thống IoT (2)

IoT Layered Architecture (Source: ITU-T)

Bốn "trụ cột" nền tảng của IoT

- **Connections**: Khả năng kết nối (mới) của các thiết bị và thông tin
- **Collection:** Khả năng thu thập dữ liệu (lớn) từ việc gia tăng kết nối các thiết bị và thông tin
- **Computation:** Khả năng tính toán cho phép chuyển đổi từ các dữ liệu đã thu thập vào các tính năng mới
- **Creation:** Khả năng sáng tạo độc đáo của các tương tác, các mô hình kinh doanh, và các giải pháp mới

New 5G & NB-IoT capabilities shift demand for celluar activity

Emerging biometrics sensing technologies provide more convenient and stronger security

Cloud & XaaS driving new business models and cost efficiencies

Artificial Intelligence drives Google dominance in digital assistant market

1.3. Các công nghệ IoT

- Phần cứng (Hardware)
- Truyền thông (Communication)
- Các giao thức (Protocols)
- Phân tích dữ liệu (Data Analytic)
- Nền tảng đám mây (Cloud Platforms)

1.3.1. Phần cứng IoT (Hardware)

Các máy tính nhúng (Embedded Computers):

- Vi điều khiển: 8-bit, 32-bit, không dùng hệ điều hành
- Vi điều khiển có dùng hệ điều hành đơn giản (ví dụ: FreeRTOS)
- Bộ xử lý 32-bit, 64-bit, hiệu năng cao, có hệ điều hành (Raspbian, Embedded Linux, Ubuntu, Embedded Windows, …)
- Các kiến trúc: AVR, Microchip, ARM, Intel, …

Arduino Uno

ESP32

Raspberry Pi

Intel Galileo

Các cảm biến (Sensors)

- Cảm biến có thể coi là thành phần quan trọng nhất trong thiết bị IoT
  - Đầu ra là tương tự hoặc đầu ra là số
- Các module cảm biến là các thiết bị thường bao gồm:
  - Thành phần cung cấp, quản lý năng lượng (energy/power modules)
  - Thành phần cảm biến (sensing modules)
  - Thành phần quản lý giao tiếp
- Thành phần quản lý giao tiếp thông qua xử lý tín hiệu (RF modules)
  - WiFi, ZigBee, Bluetooth, radio transceiver, …

Các cảm biến (Sensors)

- Có nhiều loại cảm biến:

Devices	
accelerometers	temperature sensors
magnetometers	proximity sensors
gyroscopes	image sensors
acoustic sensors	light sensors
pressure sensors	gas RFID sensors
humidity sensors	micro flow sensors

Các cảm biến (Sensors)

Các cảm biến (Laboratory type)

Wearables IoT

- **Head** – Helmets, glasses
- **Neck** – Jewelry, collars
- **Arm** – Watches, wristbands, rings
- **Torso** – Clothing, backpacks
- **Feet** – Socks, shoes

Wearables IoT

- https://www.sportswearable.net/fitness-and-sports-wearables-world-wide-market-analysis-forecasts-and-trends-through-2019-2025/

1.3.2. Truyền thông trong IoT (Communications)

Truyền thông trong IoT

- Một số chuẩn truyền thông phổ biến cho IoT:
  - NFC and RFID
  - Bluetooth
  - WiFi
  - GSM, 3G/4G/LTE, LTE-A

Truyền thông trong IoT

- Một số giao thức cho ứng dụng IoT:
  - HTTP (HyperText Transfer Protocol)
  - RESTful HTTP (Representational State Transfer)
  - MQTT (Message Queue Telemetry Transport)
  - AMQP (Advanced Message Queue Protocol)
  - CoAP (Constrained Application Protocol)
  - XMPP (Extensible Messaging and Presence Protocol)

https://www.postscapes.com/internet-of-things-technologies/

1.3.3. Phân tích dữ liệu IoT (Data Analytic)

1.3.4. IoT Technologies: Cloud Platforms

- Một số nền tảng dịch vụ đám mây trong IoT:
  - IBM BlueMix
  - AWS IoT
  - Google Cloud IoT
  - Azure IoT

https://www.postscapes.com/internet-of-things-technologies/

1.4. Các ứng dụng của IoT

Các ứng dụng của IoT

Top Industrial IoT Applications

- Healthcare
- Smart Retail
- Smart Building/Smart Home
- Smart Agriculture
- Smart Utilities (Power Energy, Water)

Healthcare

- One of the fastest sectors adopting the IoT
- A lot of sensors for tracking patients

Smart Retail

https://www.smartofthings.co.th/2018/08/27/smart-retail-solution/

Smart Retail

Amazon Go store

Smart Retail: Shopping

Scenario: shopping

(2) When shopping in the market, the goods will introduce themselves.

(1) When entering the doors, scanners will identify the tags on her clothing.

(4) When paying for the goods, the microchip of the credit card will communicate with checkout reader.

(3) When moving the goods, the reader will tell the staff to put a new one.

Smart Building

Smart Home

Source: Raymond James research.

Smart Agriculture

https://www.smartofthings.co.th/2018/08/27/smart-agriculture-solution/

Smart Utilities

Electrical power, Water Supplying

https://www.smartofthings.co.th/2018/08/27/smart-utilities-solution/

Transportation

Source: Raymond James research.

IoT – Manufacturing Applications

- Intelligent Product Enhancements
- Dynamic Response to Market Demands
- Lower Costs, Optimized Resource Use, Waste Reduction
- Improved Facility Safety
- Product Safety

Một số sản phẩm thương mại ứng dụng IoT

Một số sản phẩm thương mại ứng dụng IoT

Một số sản phẩm thương mại ứng dụng IoT

Thảo luận - Discussion

- What everyday activity can be changed by IoT?
- What existing process can be changed by IoT? Will this change lead to the greater good of society? Or to the individual?

1.5. Các thách thức của IoT

- Rất nhiều chuẩn công nghệ khác nhau:
  - Thuận lợi + Khó khăn ?

- Thách thức với các chính phủ trong vấn đề quản trị đổi mới quá nhanh của công nghệ:
  - Ví dụ Uber, Grab

- Vấn đề về quyền riêng tư (Privacy) và bảo mật (security)
  - Ví dụ: Facebook

- Thiếu vắng cơ quan quản lý, điều hành chung
  - Vấn đề chung của dịch vụ Internet

- Dễ bị tấn công trên Internet:
  - Ví dụ IP cameras

Các thách thức (Challenges) …

- **Connectivity - Vấn đề kết nối**:
  - Hiện tại, IoT dựa trên mô hình server/client để xác thực, kết nối các thiết bị trong mạng. Mô hình này đã có thể làm việc với hàng trăm, ngàn thiết bị. Vấn đề khó khăn khi số lượng thiết bị lên tới hàng triệu, tỷ trong mạng
  - Nếu không cân nhắc đến thiết kế thông lượng mạng thích hợp, vấn đề tắc nghẽn (bottlenecks) có thể xảy ra trong quá trình trao đổi dữ liệu tại server.
  - Các tác vụ có thể chuyển sang thực hiện trên các thiết bị (off-loading tasks to the edge)
  - Các mạng IoT sẽ cần các thiết bị có khả năng thực hiện phân tích dữ liệu, học máy, và thu thập dữ liệu

Các thách thức (Challenges) …

- **Brownfield deployment** (legacy infrastructure)
  – Vấn đề triển khai trên các hạ tầng cũ đã có:
  - Các thiết bị IoT, các hạ tầng mạng đã có, các công nghệ mới được kết hợp với nhau (brownfield deployment)
  - Các công ty đối mặt với vấn đề tích hợp các thiết bị công nghệ mới với hạ tầng mạng đã tồn tại

Các thách thức (Challenges) …

- **Dealing with non-standard communication protocol**:

- Vấn đề với các giao thức truyền thông phi chuẩn

  - Các mạng kết nối sẽ cần làm việc với số lượng gia tăng chưa từng có của dữ liệu từ các thiết bị và cảm biến
  - Việc kiểm soát, xử lý và lưu trữ dữ liệu sẽ gia tăng cùng với sự gia tăng của tải dữ liệu đầu vào, trong khi đó dữ liệu với sự gia tăng về kích thước, tần suất vẫn cần sẵn sàng cho việc phân tích dữ liệu

Các thách thức (Challenges) …

- **IT/OT convergence – Sự hội tụ của IT và OT**
  - Sự tích hợp/hội tụ của IT (Information Technology) và OT (Operational Technology – Công nghệ vận hành) trong các ứng dụng công nghiệp của Internet of Things (IIoT). Ví dụ về IT, OT trong nhà máy điện
  - IT: trung tâm dữ liệu (data-centric), OT: giám sát sự kiện (monitor events); IoT làm mờ sự phân biệt này thông qua việc giám sát các thiết bị đồng thời tạo ra một lượng lớn dữ liệu
  - Các doanh nghiệp vận hành công nghiệp sẽ cần điều chỉnh qui trình để thích ứng với các thiết bị IIoT và dữ liệu

Các thách thức (Challenges) …

- **Get actionable intelligence from data – Khai thác khả năng thông minh từ dữ liệu**:
  - Giá trị của dữ liệu gia tăng khi khả năng khai thác thông minh từ dữ liệu gia tăng
  - Phân tích IoT sẽ cần làm việc với các dữ liệu chưa được cấu trúc, lượng dữ liệu lớn theo thời gian thực và cả với các dữ liệu ngoại biệt

Thảo luận - Discussion

- Industry Transformations
  - IoT will cause the most transformation in non-technology-based industries.
- Give an example of a non-technology-based industry and explain how IoT will transform it.

Nội dung

- Chương 1. Tổng quan về IoT
- Chương 2. Các công nghệ IoT
- Chương 3. Lập trình ứng dụng IoT
- Chương 4. An toàn và Bảo mật IoT
- Chương 5. Thiết kế và xây dựng hệ thống IoT

Chương 2. Các công nghệ IoT

- 2.1. Cảm biến và thiết bị
- 2.2. Một số chuẩn truyền thông cho IoT
- 2.3. Các giao thức cho ứng dụng IoT
- 2.4. Nền tảng đám mây IoT

2.1. Cảm biến và thiết bị IoT

- Cảm biến và thiết bị:
  - Các cảm biến và thiết bị IoT được gọi đơn giản là thiết bị (devices) với ngầm định bao gồm cả các cảm biến được ghép nối
  - Trong nhiều tình huống, khi đề cập “thiết bị” có thể được hiểu là bao gồm cả thiết bị và cảm biến

Cảm biến và thiết bị IoT

- Cảm biến quan sát các thay đổi xung quanh và gửi thông tin về các thay đổi đó đến thiết bị
- Các thiết bị thu thập dữ liệu từ các cảm biến và gửi tới cloud/server

Cảm biến và thiết bị IoT

- Các thiết bị có thể hạn chế về tài nguyên tính toán (CPU, bộ nhớ hạn chế, …)
- Các thiết bị có thể giao tiếp truyền thông riêng mà không kết nối trực tiếp với cloud platform, ví dụ thông qua Bluetooth Low Energy (BLE)
- Các thiết bị có thể lưu trữ, xử lý và phân tích dữ liệu trước khi gửi lên cloud

Phân loại cảm biến (Types of Sensors)

- Theo yêu cầu về nguồn cấp

Type	Definition	Example
passive	Does not require external power to operate. They respond to input from their environment.	A temperature sensor that changes resistance in response to temperature changes
active	Requires external power to operate.	A camera

- Theo loại tín hiệu

Type	Definition	Example
analog	Outputs an analog continuous signal	Accelerometers, temperature sensors
digital	The output is converted to discrete values (digital 1s and 0s) before transmitting to a device	Digital pressure sensor, digital temperature sensor

- Theo loại thiết bị đo

Type	Definition	Example
chemical	Responds to chemical changes in its environment	Gas sensor
mechanical	Responds to physical changes in its environment	Microswitch
electrical	Responds to electrical changes in its environment	Optical sensor

Lựa chọn cảm biến

- Lựa chọn cảm biến cần cân nhắc đến tầm quan trọng của các yếu tố và mức độ ưu tiên trong thiết kế tổng thể.
- **Durability (Tính lâu bền):**
  - Tính lâu bên phải được cân nhắc đến dựa trên môi trường hoạt động của sensor.
  - Đảm bảo thiết bị hoạt động lâu bền trong khoảng thời gian đáng kể không phải tốn chi phí sửa chữa thay thế.
  - Ví dụ: cảm biến nhiệt kháng nước dùng cho trạm thời tiết

Lựa chọn cảm biến

- **Accuracy (Độ chính xác):**
  - Cần độ chính xác đáp ứng đủ nhu cầu (độ chính xác cao đi kèm giá thành đắt)
  - Ví dụ: Đo nhiệt độ nhà kho có thể chấp nhận độ chính xác +/-2 độ. Đo nhiệt độ hệ thống thiết bị y tế cần độ chính xác +/-0.2 độ

Lựa chọn cảm biến

- **Versatility (Tính linh hoạt):**
  - Cảm biến có thể vận hành với các biến động của môi trường xung quanh
  - Ví dụ: cảm biến cho trạm thời tiết có thể hoạt động trong điều kiện mùa hè, mùa đông. Không khả thi với loại cảm biến chỉ hoạt động trong môi trường trong nhà.

Lựa chọn cảm biến

- **Power Consumption (Tiêu thụ điện năng):**
  - Tùy theo tình huống, có thể đòi hỏi các thiết bị tiêu thụ ít điện năng, sử dụng các đặc tính tiết kiệm năng lượng
  - Ví dụ: cảm biến hoặc thiết bị sử dụng pin năng lượng mặt trời cần thiết kế chế độ ngủ (sleep mode) tiết kiệm năng lượng, thức dậy (wake-up) khi cần thu thập và truyền dữ liệu

Lựa chọn cảm biến

- Cân nhắc môi trường đặc biệt:
  - Ví dụ: Cảm biến trong giám sát chất lượng nước trong nuôi thủy sản đòi hỏi sensor đặt trong môi trường nước (pH, DO, TDS, …) khác với các cảm biến dựa trên lấy mẫu phân tích.

Lựa chọn cảm biến

- **Cost (Chi phí):**
  - Các mạng IoT thường bao gồm hàng trăm, ngàn thiết bị và cảm biến.
  - Thiết kế cần cân nhắc đến chi phí lắp đặt, bảo trì, thay thế, etc.

Devices – Thiết bị IoT

- Một “Thing” trong “Internet of Things” là một đơn vị xử lý mà có khả năng kết nối vào internet để trao đổi dữ liệu với cloud.
- Các thiết bị thường được gọi là "smart devices" or "connected devices."
- Thiết bị giao tiếp với 2 loại dữ liệu: telemetry (dữ liệu thu thập) và state (trạng thái của thiết bị).

Các loại thông tin

- Mỗi thiết bị có thể cung cấp hoặc sử dụng một vài loại thông tin phục vụ cho mỗi loại hệ thống backend khác nhau.
- **Device metadata:** Metadata chứa các thông tin mô tả về thiết bị (thường ít thay đổi). Ví dụ:
  - Identifier (ID) – Định danh thiết bị
  - Class or type: Lớp/loại thiết bị
  - Model
  - Revision
  - Date manufactured
  - Hardware serial number

Các loại thông tin

- **Telemetry**:
  - Dữ liệu được thu thập bởi thiết bị qua các cảm biến được gọi là telemetry.
  - Là dữ liệu chỉ đọc (từ môi trường xung quanh)

Các loại thông tin

- **State information**:
  - Thông tin trạng thái: Mô tả trạng thái hiện thời của thiết bị. Có thể đọc/ghi, cập nhật (nhưng ít thường xuyên)

Lệnh thực thi trên thiết bị (Device Commands)

- Commands: Là các hành động được được thực hiện bởi thiết bị.
- Ví dụ:
  - Quay 360 độ sang phải.
  - Tắt/bật đèn
  - Tăng tần suất truyền dữ liệu

Định nghĩa thiết bị

- Trong hệ thống IoT, định nghĩa thiết bị có thể thay đổi phụ thuộc vào nhu cầu của dự án.
- Mỗi thiết bị có thể được cân nhắc như một thực thể tách biệt, hoặc một thiết bị có thể chịu trách nhiệm với một nhóm cảm biến.

Định nghĩa Thiết bị – Ví dụ

- Giám sát nhiệt độ phòng khách sạn
- **Giải pháp 1:** Mỗi phòng có 3 cảm biến: one near the floor, one near the bed, and one near the ceiling.

Định nghĩa Thiết bị – Ví dụ

- **Giải pháp 1:** dữ liệu được gửi đến cloud như là 3 thiết bị khác nhau, mỗi cái gửi một thông tin nhiệt độ đến cloud.
  - {deviceID: "dh28dslkja", "location": "floor", "room": 128, "temp": 22 }
  - {deviceID: "8d3kiuhs8a", "location": "ceiling", "room": 128, "temp": 24 }
  - {deviceID: "kd8s8hh3o", "location": "bedside", "room": 128, "temp": 23 }

Định nghĩa Thiết bị – Ví dụ

- **Giải pháp 2:** mỗi phòng sử dụng 1 thiết bị chịu trách nhiệm gửi dữ liệu từ 3 cảm biến.

Định nghĩa Thiết bị – Ví dụ

- **Giải pháp 2:** dữ liệu được gửi đến cloud bằng 1 thiết bị với 3 cảm biến. Có thể bổ sung thêm xử lý tính nhiệt độ trung bình trên thiết bị.
  - {deviceID: "dh28dslkja", "room": 128, "temp_floor": 22, "temp_ceiling": 24, "temp_bedside": 23, "average_temp": 23 }

- Việc lựa chọn giải pháp nào phụ thuộc vào ý định sử dụng thông tin ở thời điểm hiện tại và cả tiềm năng trong tương lai.

2.2. Một số chuẩn truyền thông

- Communication standards:
  - NFC and RFID
  - Bluetooth
  - WiFi
  - GSM, GPRS, 2G/3G/4G, LTE
  - ZigBee
  - 6LoWPAN
  - Thread

Một số chuẩn truyền thông trong IoT

- RFID (Radio-frequency identification)

https://www.youtube.com/watch?v=MAA9JpGraoU

Frequency: 120–150 kHz (LF), 13.56 MHz (HF), 433 MHz (UHF), 865-868 MHz (Europe)902-928 MHz (North America) UHF, 2450-5800 MHz (microwave), 3.1–10 GHz (microwave)
Range: 10cm to 200m
Examples: Road tolls, Building Access, Inventory

Một số chuẩn truyền thông trong IoT

- NFC (Near-Field Communications)

Frequency: 13.56 MHz
Range: < 0.2 m
Examples: Smart Wallets/Cards, Action Tags, Access Control

Một số chuẩn truyền thông trong IoT

- Bluetooth

Frequency: 2.4GHz
Range: 1-100m
Examples: Hands-free headsets, key dongles, fitness trackers

Một số chuẩn truyền thông trong IoT

- WiFi

Frequency: 2.4 GHz, 3.6 GHz and 4.9/5.0 GHz bands.
Range: Common range is up to 100m but can be extended.
Applications: Routers, Tablets, etc

More: https://www.postscapes.com/internet-of-things-technologies/

Một số chuẩn truyền thông trong IoT

- GSM (Global System for Mobile communications)

Frequency: Europe: 900MHz & 1.8GHz, US: 1.9GHz & 850MHz, Full List can be found here.
Data Rate: 9.6 kbps
Examples: Cell phones, M2M, smart meter, asset tracking

More: https://www.postscapes.com/internet-of-things-technologies/

Một số chuẩn truyền thông trong IoT

- ZigBee

ZigBee®
Control your world

- Standard: ZigBee 3.0 based on IEEE802.15.4
  - Frequency: 2.4GHz
  - Range: 10-100m
  - Data Rates: 250kbps

Một số chuẩn truyền thông trong IoT

- **6LoWPAN**
- IPv6 protocol over low-power wireless PANs (sử dụng giao thức IPv6 trong các mạng PAN không dây công suất thấp)

6LoWPAN

IPv6-based Low-power
Wireless Personal Area Networks

Một số chuẩn truyền thông trong IoT

- Thread: Giao thức IP mới dựa trên nền tảng IPv6 được thiết kế riêng cho mảng tự động hóa trong nhà thông minh, các tòa nhà.
- Ra mắt vào giữa năm 2014 bởi Theard Group, giao thức Thread dựa trên các tiêu chuẩn khác nhau, bao gồm IEEE802.15.4, IPv6 và 6LoWPAN, và cung cấp một giải pháp dựa trên nền tảng IP cho các ứng dụng IoT
- Tiêu chuẩn: Theard, dựa trên IEEE802.15.4 và 6LowPAN.
- Tần số: 2.4GHz (ISM).
- Tham khảo: https://smarthomekit.vn/thread-protocol/

Một số chuẩn truyền thông trong IoT

Truyền thông trong IoT

Additional:

- 3G
- 4G LTE
- ANT
- Dash7
- Ethernet
- GPRS
- PLC / Powerline
- QR Codes, EPC
- WiMax
- X-10
- 802.15.4
- Z-Wave
- Zigbee

Backbone: IPv4, IPv6, TCP/UDP

More: https://www.postscapes.com/internet-of-things-technologies/

2.3. Một số giao thức cho ứng dụng IoT

Các giao thức ứng dụng IoT

- HTTP, HTTPS
- REST API HTTP (Representational State Transfer)
- MQTT (Message Queue Telemetry Transport)
- AMQP (Avanced Message Queue Protocol)

https://www.postscapes.com/internet-of-things-technologies/

2.3.1. Giao thức HTTP

HTTP = HyperText Transfer Procol

Ví dụ ứng dụng sử dụng HTTP

Hệ thống Smart Locker

Ví dụ thực hiện http request với Java (1)

- Sử dụng thư viện HttpURLConnection trên Java thực hiện một http request tới http REST api
- **Bước 1:** Tạo một kết nối http connection, thiết lập các thuộc tính cho request

```
public String requestUserInfor(String devId, String qrDevId, String
qrCode, int timeout){
    String result = "";
    try {
        String urlApi =
"https://203.171.20.94:8080/api/AccessControl/GetUserInfor";
        URL url = new URL(urlApi);
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setRequestMethod("POST");
        conn.setConnectTimeout(timeout);
        conn.setReadTimeout(timeout);
        conn.setRequestProperty("Content-Type", "application/json;
charset=utf-8");
```

Ví dụ thực hiện http request với Java (2)

- **Bước 2:** Ghi dữ liệu cần gửi vào luồng output stream của request. Dữ liệu có thể được đóng gói bằng JSON

```
        ...
        OutputStream os = conn.getOutputStream();
        BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(os, "UTF-8"));
        JSONObject jsonObj = new JSONObject();
        try {
            jsonObj.accumulate("deviceId", devId);
            jsonObj.accumulate("qrCodeId", qrDevId);
            jsonObj.accumulate("qrCodeValue", qrCode);
        } catch (JSONException e) {
            e.printStackTrace();
        }

        writer.write(jsonObj.toString());
        writer.flush();
        writer.close();
        os.close();
        conn.connect();
```

Ví dụ thực hiện http request với Java (3)

- **Bước 3:** Nhận phản hồi http response từ server. Kiểm tra mã trả về và đọc dữ liệu từ luồng input stream của http response

```
        ...
        StringBuffer sb = new StringBuffer();
        if (conn.getResponseCode() == HttpURLConnection.HTTP_OK) {
            InputStream in = conn.getInputStream();
            int chr;
            while ((chr = in.read()) != -1) {
                sb.append((char) chr);
            }
            in.close();
        } else {
            sb.append(conn.getResponseCode());
        }
        result = sb.toString();
        conn.disconnect();
    }
    catch(IOException e){
        e.printStackTrace();
    }
    return result;
}
```

Http Response code

- 200: OK
- 201: Created
- 204: No Content
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found
- 408: Request Timeout
- 500: Internal Server Error
- 502: Bad Gateway
- 503: Service Unavailable

Sử dụng công cụ Postman

Postman

File Edit View Help

Home Workspaces Reports Explore Search Postman Sign In Create Acc

Working locally in Scratch Pad. **Switch to a Workspace**

Overview | POST ht... | GET htt... | GET htt... | POST ht... | GET htt... | GET htt... | POST ht... | No Environment

https://203.171.20.94:8080/api/AccessControl/GetUserInfor Save

POST https://203.171.20.94:8080/api/AccessControl/GetUserInfor Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Response

Find and Replace Console Runner Trash

Ví dụ thực hiện http request với python

```
import requests
URL = "https://maps.googleapis.com/maps/api/geocode/json"
location = "Hanoi University of Science and Technology"
PARAMS = {'address':location}
# sending get request and saving the response as response
object
r = requests.get(url = URL, params = PARAMS)
# extracting data in json format
data = r.json()

# extracting latitude, longitude and formatted address
# of the first matching location
latitude = data['results'][0]['geometry']['location']['lat']
longitude = data['results'][0]['geometry']['location']['lng']
formatted_address = data['results'][0]['formatted_address']
# printing the output
print("Latitude:%s\nLongitude:%s\nFormatted Address:%s"
   %(latitude, longitude,formatted_address))
```

Bài tập 1

- HTTP request
  - Viết chương trình minh họa thực hiện HTTP request, lấy và hiển thị dữ liệu trả về
  - URL = http://203.171.20.94:8012/api/AccessControl/GetUserInfor
  - Request Body là json object


```
{"deviceId":"8a0fc66a61a959f6", "qrCodeId": "a652d57094b7590b0dea115b156c07098abdea87", "qrCodeValue":"P22498244182551944"}
```


  - Dùng Postman để kiểm thử

2.3.2. Giao thức MQTT

- **The Messaging and Data Exchange Protocol of the IoT**
- **MQTT** (Message Queuing Telemetry Transport):
  - Giao thức truyền thông điệp (message) theo mô hình publish/subscribe (xuất bản – theo dõi)
  - Sử dụng băng thông thấp, độ tin cậy cao và có khả năng hoạt động trong điều kiện đường truyền không ổn định.

Giao thức MQTT

- Kiến trúc mức cao của MQTT gồm 2 thành phần chính:
  - Broker
  - Clients

Giao thức MQTT

- **Publish/Subscribe:**
  - Trong một hệ thống sử dụng giao thức MQTT, nhiều node trạm (gọi là mqtt client – gọi tắt là client) kết nối tới một MQTT server (gọi là broker).
  - Mỗi client sẽ đăng ký một vài kênh (topic), ví dụ như "/client1/channel1", "/client1/channel2". Quá trình đăng ký này gọi là **"subscribe".** Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gửi dữ liệu vào kênh đã đăng ký.
  - Khi một client gửi dữ liệu tới kênh đó, gọi là **"publish"**.

Giao thức MQTT

https://www.hivemq.com/mqtt-essentials/

Giao thức MQTT

- **QoS (Quality of Service):**
  - Có 3 tùy chọn **QoS** khi “publish” và “subscribe”:
  - **QoS0** Broker/client sẽ gửi dữ liệu đúng 1 lần, quá trình gửi được xác nhận bởi chỉ giao thức TCP/IP, (“fire and forget”).
  - **QoS1** Broker/client sẽ gửi dữ liệu với ít nhất 1 lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
  - **QoS2** Broker/client đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng 1 lần, quá trình này phải trải qua 4 bước bắt tay

Giao thức MQTT

- Sử dụng công cụ client: MQTTBox
  - https://www.hivemq.com/mqtt-toolbox/
  - Publish/Subscribe (gửi/nhận) dữ liệu

Công cụ mqtt client: MQTTBox

Thư viện MQTT client

- Thư viện mqtt client cho các nền tảng khác nhau:
  - https://www.hivemq.com/mqtt-client-library-encyclopedia/

Ví dụ 1

- publish/subscribe (gửi/nhận) dữ liệu sử dụng thư viện Paho mqtt client cho Java
- Thư viện Paho mqtt client
  - https://www.eclipse.org/paho/index.php?page=clients/java/index.php
- Using the Paho Java Client:
  - Using Maven dependency
  - Hoặc add thư viện .jar (Build Path >> Add External Archives):

Download file thư viện: org.eclipse.paho.client.mqttv3-1.1.1.jar

Publish message

```
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
public class App {
public static void main(String[] args) {
    public static void main( String[] args ){
        System.out.println( "MQTT Demo!" );
        new App().doDemo();
    }
    public void doDemo() {
        String subTopic = "/ktmt/iot";
        String pubTopic = "/ktmt/iot";
        String content = "Hello World";
        int qos = 1;
        String broker = "tcp://broker.hivemq.com:1883";
        String clientId = "ID_OF_CLIENT";
        MemoryPersistence persistence = new MemoryPersistence();
        try {
            MqttClient client = new MqttClient(broker, clientId,
persistence);
```

```java
        // MQTT connection option
        MqttConnectOptions connOpts = new MqttConnectOptions();
        // retain session
        connOpts.setCleanSession(true);
        // set callback
        client.setCallback(new OnMessageCallback());
        // establish a connection
        System.out.println("Connecting to broker: " + broker);
        client.connect(connOpts);
        System.out.println("Connected");
        // Subscribe
        client.subscribe(subTopic);
        // Required parameters for message publishing
        MqttMessage message = new MqttMessage(content.getBytes());
        message.setQos(qos);
        System.out.println("Publishing message: " + content);
        client.publish(pubTopic, message);
        System.out.println("Message published");
        client.disconnect();
        System.out.println("Disconnected");
        client.close();
        System.exit(0);
    }
    catch (MqttException me) {
         me.printStackTrace();
    }
}
```

Subscribe - Callback

```
public class OnMessageCallback implements MqttCallback {
    public void connectionLost(Throwable cause) {
        // After the connection is lost, it usually reconnects here
        System.out.println("disconnect, you can reconnect");
    }
    public void messageArrived(String topic, MqttMessage message)
throws Exception {
        // The messages obtained after subscribe will be executed here
        System.out.println("Received message topic:" + topic);
        System.out.println("Received message Qos:" + message.getQos());
        System.out.println("Received message content:" + new
            String(message.getPayload()));
    }
    public void deliveryComplete(IMqttDeliveryToken token) {
        System.out.println("deliveryComplete---------" +
    token.isComplete());
    }
}
```

Ví dụ 2:

- Nhận dữ liệu thu thập cảm biến từ các thiết bị gửi lên
- Tham khảo mã nguồn:
  - https://github.com/hungpn37/iot_mqttdemo

Bài tập 2

- Viết một chương trình mqtt client thực hiện:
  - Gửi (publish) dữ liệu lên mqtt broker
  - Nhận (subscribe) dữ liệu từ mqtt broker
  - Đóng gói dữ liệu bằng JSON. Ví dụ:

    {"id":11, "packet_no":126, "temperature":30, "humidity":60, "tds":1100, "pH":5.0}
  - Dùng MQTTBox để minh họa gửi nhận
  - Ví dụ dùng mqtt broker: `tcp://broker.hivemq.com:1883`

2.3.3. Giao thức AMQP

- AMQP (Advanced Message Queue Protocol): Giao thức truyền nhận dùng hàng đợi thông điệp
- RabbitMQ: Một broker sử dụng giao thức AMQP
- RabbitMQ broker đóng vai trò trung gian lưu trữ cũng như điều phối thông điệp (message) giữa bên gửi (producer) và bên nhận (consumer)

Giao thức AMQP

- RabbitMQ:
  - Luồng gửi nhận message qua RabbitMQ

Giao thức AMQP

- RabbitMQ:
  - 4 loại Exchange: direct, topic, fanout, headers

Giao thức AMQP

- Cài đặt RabbitMQ broker (server)
  - https://www.rabbitmq.com/download.html
  - https://www.rabbitmq.com/install-windows.html
- Quản trị RabbitMQ broker bằng công cụ:
  - https://www.rabbitmq.com/management.html
- RabbitMQ tutorials:
  - https://www.rabbitmq.com/getstarted.html

Tutorial 1

- Viết chương trình gửi nhận dữ liệu qua RabbitMQ
  - Cài đặt, sử dụng một RabbitMQ broker
  - Chương trình Producer gửi dữ liệu (message)
  - Chương trình Consumer nhận dữ liệu
    - Sử dụng default exchange (type: direct)

https://www.rabbitmq.com/tutorials/tutorial-one-java.html

Tutorial 1. Producer

Sender.java

```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import java.nio.charset.StandardCharsets;

public class send {
    private final static String QUEUE_NAME = “ktmt";
    public static void main(String[] argv) throws Exception {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost(“localhost");
        factory.setUsername(“guest");
        factory.setPassword(“guest");
        try (Connection connection = factory.newConnection();
             Channel channel = connection.createChannel()) {
            channel.queueDeclare(QUEUE_NAME, false, false, false, null);
            String message = "Hello World! Publish message to broker";
            channel.basicPublish("", QUEUE_NAME, null,
                message.getBytes(StandardCharsets.UTF_8));
            System.out.println(" [x] Sent '" + message + "'");
        }
    }
}
```

Tutorial 1. Consumer

Recv.java

```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.DeliverCallback;
public class Recv {
    private final static String QUEUE_NAME = “ktmt";
    public static void main(String[] argv) throws Exception {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost(“localhost");
        factory.setUsername(“guest");
        factory.setPassword(“guest");
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();
        channel.queueDeclare(QUEUE_NAME, false, false, false, null);
        System.out.println(" [*] Waiting for messages. To exit press CTRL+C");
        DeliverCallback deliverCallback = (consumerTag, delivery) -> {
            String message = new String(delivery.getBody(), "UTF-8");
            System.out.println(" [x] Received '" + message + "'");
        };
        channel.basicConsume(QUEUE_NAME, true, deliverCallback, consumerTag -> {
});
    }
}
```

Tutorial 3

- Cải tiến Tutorial 1: một Producer gửi message đến tất cả các consumer
  - Sử dụng exchange type = “fanout” (broadcast)

```
channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
```

https://www.rabbitmq.com/tutorials/tutorial-three-java.html

Tutorial 4

- Routing (định tuyến) thông điệp
- Chọn thông điệp muốn nhận (Receiving messages selectively)

Producer

```
channel.exchangeDeclare(EXCHANGE_NAME, "direct");
channel.basicPublish(EXCHANGE_NAME, “black”, null, message.getBytes());
```

Consumer

```
channel.queueBind(queueName, EXCHANGE_NAME, “black”);
```

https://www.rabbitmq.com/tutorials/tutorial-four-java.html

Tutorial 5

- Sử dụng exchange type = “topic”
- `routing_key` must be a list of words, delimited by dots.
  - *(star) can substitute for exactly one word.
  - #(hash) can substitute for zero or more words.

Producer

```
channel.basicPublish(EXCHANGE_NAME, routingKey, null, message.getBytes("UTF-8"));
```

Consumer

```
channel.queueBind(queueName, EXCHANGE_NAME, bindingKey);
```

https://www.rabbitmq.com/tutorials/tutorial-five-java.html

Bài tập 3

- Viết một chương trình gửi/nhận dữ liệu qua RabbitMQ broker:
  - Gửi (publish) dữ liệu lên RabbitMQ broker
  - Nhận (subscribe) dữ liệu từ RabbitMQ broker
  - Đóng gói dữ liệu bằng JSON. Ví dụ:

    {"id":11, "packet_no":126, "temperature":30, "humidity":60, "tds":1100, "pH":5.0}

2.4. Nền tảng IoT Cloud Platform

- Các kiến trúc IoT phải có khả năng mở rộng (scaling) qui mô kết nối các thiết bị, thu thập dữ liệu (data gathering), xử lý (data processing) và lưu trữ dữ liệu (data storage)
- Khả năng thực hiện các công việc trên nhanh chóng trong khi vẫn thực hiện phân tích dữ liệu theo thời gian thực (real-time data insights)
- Gửi lượng lớn dữ liệu lên cloud làm chậm quá trình xử lý, đòi hỏi thêm băng thông cho truyền nhận dữ liệu

IoT Cloud Platform

- Giải pháp tính toán phân tán (distributed computing) ~ **fog or edge computing** trở nên phổ biến
- Edge computing (tính toán cận biên): việc xử lý tính toán thực hiện trên các nodes trong mạng là các thiết bị IoT (IoT devices) nằm ở biên “edge” của mạng
- Giải pháp dẫn đến nhu cầu cho thiết bị IoT có khả năng: xử lý, phân tích dữ liệu cục bộ (cleaning, processing, analyzing data locally). Chỉ gửi dữ liệu đã xử lý trước đến cloud

IoT Cloud Platform

IoT Cloud Platforms

- Một số nền tảng dịch vụ IoT Cloud:
  - IBM Watson IoT
  - AWS IoT
  - Google Cloud IoT
  - Microsoft Azure IoT

https://www.postscapes.com/internet-of-things-technologies/

Google Cloud IoT Architecture

Be capable of doing data import, process, storage, and analysis for hundreds of millions of events per hour from devices all over the world

Google Cloud IoT

- Kiến trúc Google Cloud IoT chia làm 4 thành phần:
  - data gathering (thu thập dữ liệu trên thiết bị)
  - data ingest (tiếp nhận dữ liệu trên Cloud)
  - data processing (xử lý dữ liệu)
  - data analysis (phân tích dữ liệu)

Google Cloud IoT

- Data Gathering (Thu thập dữ liệu trên thiết bị):
  - Xảy ra trên các thiết bị (devices) và cảm biến (sensors)
  - Cảm biến (sensors) thu thập (gather) dữ liệu từ môi trường, gửi lên cloud hoặc trực tiếp hoặc gián tiếp thông qua thiết bị trung gian
  - Thiết bị (devices) chuẩn bị dữ liệu truyền lên cloud. Tùy thuộc loại mạng kết nối, quá trình chuẩn bị bao gồm: làm sạch dữ liệu, tiền xử lý, phân tích hoặc có thể sử dụng cả machine learning

Google Cloud IoT

- Cloud IoT Edge:

Google Cloud IoT

- Cloud IoT Edge:
  - Trong kiến trúc Goolge Cloud IoT, giai đoạn thu thập dữ liệu có thể thực hiện trên Cloud IoT Edge.
  - Thành phần này là một nhóm các thiết bị có khả năng thực hiện các phân tích thời gian thực và ML (machine learning). Cho phép mở rộng các xử lý dữ liệu và ML thực hiện trên các thiết bị (edge devices).
  - Cloud IoT Edge có thể sử dụng Android Things OS, Linux-based OS.
  - Gồm 2 thành phần: Edge IoT Core và Edge ML

Google Cloud IoT

- Ingest and Process data (Tiếp nhận và xử lý dữ liệu):

Google Cloud IoT

- Data Analytics and ML (Phân tích dữ liệu và Học máy):
  - Có thể thực hiện trên Edge hoặc Cloud.
  - Google's Cloud IoT Core Data Analytics and ML are fully integrated with IoT data.

Nội dung

- Chương 1. Tổng quan về IoT
- Chương 2. Các công nghệ IoT
- Chương 3. Lập trình ứng dụng IoT
- Chương 4. An toàn và Bảo mật IoT
- Chương 5. Thiết kế và xây dựng hệ thống IoT

Chương 3. Lập trình ứng dụng IoT

- 3.1. Lập trình cho thiết bị IoT
- 3.2. Xây dựng IoT Server
- 3.3. Khai thác dịch vụ IoT Cloud Platform

3.1. Lập trình cho thiết bị IoT

- Các thiết bị dùng vi điều khiển
  - Kiến trúc 8 bit, hiệu năng thấp, không dùng hệ điều hành
  - Lập trình firmware, giao tiếp vào ra cơ bản GPIO, ghép nối các ngoại vi (sensor, actuator) theo các chuẩn truyền thông phổ biến UART, SPI, I2C
  - Ví dụ: ATmega, PIC, AVR, Arduino

3.1. Lập trình với thiết bị IoT

- Các thiết bị dùng Vi xử lý 32 bit, hiệu năng trung bình
  - Có thể dùng hệ điều hành đơn giản (FreeRTOS, TinyOS, uCLinux, …)
  - Giao tiếp vào ra cơ bản GPIO, ghép nối các ngoại vi theo các chuẩn truyền thông phổ biến UART, SPI, I2C
  - Có thêm khả năng kết nối Wi-Fi, Bluetooth, GSM module,
  - Ví dụ: ESP8266, ESP32, Arduino Uno WiFi

3.1. Lập trình với thiết bị IoT

- Các thiết bị dùng Vi xử lý 32/64 bit, hiệu năng cao
  - Dùng hệ điều hành (Embedded Linux, Raspbian, Windows Embedded, Android, Ubuntu, …)
  - Kết nối Wi-Fi, Ethernet, …
  - Các giao tiếp ngoại vi qua cơ chế driver trên hệ điều hành
  - Có thể sử dụng các thư viện trên hệ điều hành (Java, Python, .Net Framework, OpenCV, …)
  - Ví dụ: Raspberry Pi

3.1.1. Lập trình Arduino

- Arduino Uno: Microchip ATmega328 không dùng hệ điều hành

Lập trình Arduino

- Cài đặt Arduino IDE
  - https://www.arduino.cc/en/guide/windows
- Cấu trúc code

```
void setup()
{
        //Used to indicate the initial values of system on starting.
}

void loop()
{
        Contains the statements that will run whenever the system
is powered after setup.
}
```

Lập trình Arduino

- Program used to code and upload it to arduino boards (using PC)
- Editor (for code edit)
- Sketch (piece of program)

Lập trình Arduino

- Tài liệu: Arduino References
- http://arduino.cc/en/Reference/HomePage

Giao tiếp ngoại vi

- GPIO, UART, I2C, SPI, ADC, PWM

Ví dụ 1: Giao tiếp GPIO - Output

- LED on/off trên pin 13

```
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // turn the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off
  delay(1000);              // wait for a second
}
```

Ví dụ 1: Giao tiếp GPIO - Input

- Đọc trạng thái nút bấm trên pin 2, turn on/off trên pin 13

```
const int buttonPin = 2;  // the number of the pushbutton pin
const int ledPin =  13;   // the number of the LED pin
int buttonState = 0;
void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(100);
}
```

Ví dụ 2: Giao tiếp UART

- Sử dụng thư viện Serial của Arduino
  - https://www.arduino.cc/reference/en/language/functions/communication/serial/

```
int incomingByte = 0;
void setup() {
    // opens serial port, sets data rate to 9600 bps
    Serial.begin(9600);
}
void loop() {
    // send data only when you receive data:
    if (Serial.available() > 0) {
        // read the incoming byte:
        incomingByte = Serial.read();
        // say what you got:
        Serial.print("I received: ");
        Serial.println(incomingByte, DEC);
    }
}
```

Ví dụ 3: Giao tiếp I2C

- Sử dụng thư viện Wire
  - https://www.arduino.cc/en/Reference/Wire
- Giao tiếp SRFxx Sonic Range Finder
  - https://www.arduino.cc/en/Tutorial/LibraryExamples/SFRRangerReader

3.1.2. Lập trình thiết bị ESP32/ESP8266

- Lập trình với ESP32/ESP8266
  - https://esp32.vn/index.html
  - https://randomnerdtutorials.com/projects-esp32/

Lập trình thiết bị ESP32

- A. Thiết lập Arduino IDE cho ESP32
- B. Kết nối WiFi cho ESP32
- C. Giao tiếp HTTP với ESP32
- D. Giao tiếp MQTT với ESP32
- E. Lập trình multi-tasks với FreeRTOS

A. Thiết lập Arduino IDE cho ESP32

- Cài đặt Add-on ESP32 Board trên Arduino IDE:
  - 1) Mở **File**> **Preferences**
  - 2) Nhập https://dl.espressif.com/dl/package_esp32_index.json vào trường “Additional Board Manager URLs”

Thiết lập Arduino IDE cho ESP32

3) Chọn**Tools** > **Board** > **Boards Manager…**

Thiết lập Arduino IDE cho ESP32

- Kết nối bo mạch ESP32 với máy tính
  - Chọn Board trong **Tools** > **Board** menu (ví dụ **DOIT ESP32 DEVKIT V1**)

Kết nối với thiết bị

Nạp chương trình cho thiết bị

- Upload the Blink Program. This program should blink the LED at an interval of 1 second.

```
int LED_BUILTIN = 2;
void setup() {
        pinMode (LED_BUILTIN, OUTPUT);
}
void loop() {
        digitalWrite(LED_BUILTIN,HIGH);
        delay(1000);
        digitalWrite(LED_BUILTIN,LOW);
        delay(1000);
}
```

B. Kết nối Wi-Fi cho ESP32

Ví dụ 1: File > Examples > WiFi (ESP32) > WiFiClientBasic

```
#include <WiFi.h>
const char* ssid = "yourNetworkName";
const char* password =  "yourNetworkPass";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println("Connected to the WiFi network");
}

void loop() {
}
```

Ví dụ 2: File > Examples > WiFi (ESP32) > WiFiClient

Ví dụ kết nối Wi-Fi

- Kết quả: Ghi log ra Serial port

C. Giao tiếp HTTP với ESP32

- HTTP request methods: GET vs. POST
- HTTP GET:
  - Sử dụng để request dữ liệu/tài nguyên (get values from APIs)
  - Dữ liệu được gửi trong URL của GET request
  - Ví dụ:

```
GET /update-sensor?temperature=value1
```

  - Hoặc get JSON object:

```
GET /get-sensor
```

Giao tiếp HTTP với ESP32

- HTTP POST:
  - Gửi dữ liệu đến server để create/update tài nguyên
  - Dữ liệu được gửi trong body của POST request
  - Ví dụ:

```
POST /update-sensor HTTP/1.1 Host: example.com
api_key=api&sensor_name=name&temperature=value1&humidity=value2&pressure=value3
Content-Type: application/x-www-form-urlencoded
```

  - Hoặc gửi JSON object:

```
POST /update-sensor HTTP/1.1 Host: example.com {api_key: "api",
sensor_name: "name", temperature: value1, humidity: value2, pressure: value3}
Content-Type: application/json
```

ESP32 HTTP GET

- ESP 32 gửi dữ liệu lên server dùng http get request

https://randomnerdtutorials.com/esp32-http-get-post-arduino/

ESP32 HTTP GET – Code minh họa

Part 1. Connect WiFi

```
#include <WiFi.h>
#include <HTTPClient.h>
const char* ssid = "HungiPhone";
const char* password = "hung1234";
String serverName = "https://postman-echo.com/get";
void setup()
{
    Serial.begin(115200);
    delay(10);
    // We start by connecting to a WiFi network
    WiFi.begin(ssid, password);
    Serial.print("Wait for WiFi... ");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.println("Connecting to WiFi ...");
        delay(1000);
    }
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    delay(500);
}
```

ESP32 HTTP GET – Code minh họa

Part 2. Send http GET request

```
void loop()
{
    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        String serverPath = serverName + "?temp1=24.7&temp2=30";
        // Your Domain name with URL path or IP address with path
        http.begin(serverPath.c_str()); // Send HTTP GET request
        int httpResponseCode = http.GET();
        if (httpResponseCode > 0) {
            Serial.print("HTTP Response code: ");
            Serial.println(httpResponseCode);
            String payload = http.getString();
            Serial.println(payload);
        }
        else {
            Serial.print("Error code: "); Serial.println(httpResponseCode);
        }
        http.end();
    }
    else {
        Serial.println("WiFi Disconnected");
    }
    delay(3000);
}
```

ESP32 HTTP POST

- Gửi dữ liệu dùng:
  - URL encoded (đóng gói trong URL)
  - JSON object (đóng gói trong request body)
  - Plain text

Tham khảo code: https://randomnerdtutorials.com/esp32-http-get-post-arduino/

ESP32 HTTP POST

- POST dữ liệu với URL encoded

POST /update-sensor HTTP/1.1 Host: 192.168.1.106:1880 api_key=tPmAT5Ab3j7F9&sensor=BME280&value1=24.25&value2=49.54 &value3=1005.14 Content-Type: application/x-www-form-urlencoded

```
// Your Domain name with URL path or IP address with path
http.begin(serverName);
// Specify content-type header
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
// Data to send with HTTP POST
String httpRequestData =
"api_key=tPmAT5Ab3j7F9&sensor=BME280&value1=24.25&value2=49.54&value3
=1005.14";
// Send HTTP POST request
int httpResponseCode = http.POST(httpRequestData);
```

ESP32 HTTP POST

- Post JSON object

POST /update-sensor HTTP/1.1 Host: example.com {api_key: "tPmAT5Ab3j7F9", sensor_name: "BME280", temperature: 24.25; humidity: 49.54; pressure: 1005.14} Content-Type: application/json

```
http.addHeader("Content-Type", "application/json");
int httpResponseCode =
http.POST("{\"api_key\":\"tPmAT5Ab3j7F9\",\"sensor\":\"BME280\",\"value1\"
:\"24.25\",\"value2\":\"49.54\",\"value3\":\"1005.14\"}");
```

- Post Plain Text

```
http.addHeader("Content-Type", "text/plain");
int httpResponseCode = http.POST("Hello, World!");
```

Sử dụng thư viện Arduino JSON

- Sketch > Include Library > Manage Libraries
- Add thư viện trên Arduino Library Manager

ESP32 Web server

- Xây dựng một web server đơn giản trên ESP32, điều khiển GPIO
- Ví dụ:
  - https://randomnerdtutorials.com/esp32-web-server-arduino-ide/

ESP32 Web server

D. Giao tiếp MQTT với ESP32

- Sử dụng thư viện mqtt PubSubClient cho Arduino
  - https://www.arduinolibraries.info/libraries/pub-sub-client
  - https://github.com/knolleary/pubsubclient

Giao tiếp MQTT với ESP32/ESP8266

- Tham khảo mã nguồn:
  - https://github.com/knolleary/pubsubclient/blob/master/examples/mqtt_esp8266/mqtt_esp8266.ino

```
#include <ESP8266WiFi.h>          //Hoặc #include <WiFi.h>
#include <PubSubClient.h>
WiFiClient espClient;
PubSubClient client(espClient);
const char* mqtt_server = "broker.mqtt-dashboard.com";
void setup_wifi() { …}
void callback(char* topic, byte* payload, unsigned int length) { …}
void reconnect() { …}
void setup() {
        Serial.begin(115200);
        setup_wifi();
        client.setServer(mqtt_server, 1883);
        client.setCallback(callback);
}
void loop() { … }
```

E. Lập trình multi-task với FreeRTOS

- RTOS = Real Time OS

E. Lập trình multi-task với FreeRTOS

- FreeRTOS multi-tasking

Lập trình multi-task với FreeRTOS

- Create a task

```
void toggleLED(void * parameter){
    for(;;){ // infinite loop
        // Turn the LED on
        digitalWrite(led1, HIGH);
        // Pause the task for 500ms
        vTaskDelay(500 / portTICK_PERIOD_MS);
        // Turn the LED off
        digitalWrite(led1, LOW);
        // Pause the task again for 500ms
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

Lập trình multi-task với FreeRTOS

- Create a task

```
void setup() {
xTaskCreate(
        toggleLED, // Function that should be called
        "Toggle LED", // Name of the task (for debugging)
        1000, // Stack size (bytes)
        NULL, // Parameter to pass
        1, // Task priority
        NULL// Task handle );
}
```

https://savjee.be/2020/01/multitasking-esp32-arduino-freertos/

https://savjee.be/2019/07/Home-Energy-Monitor-ESP32-CT-Sensor-Emonlib/

3.2. Xây dựng Web API cho IoT

3.2.1. Công nghệ .Net Core, Entity Framework

3.2.2. Công nghệ NodeJS, MongoDB

3.2.1. Xây dựng Web API với .NET Core

- Sử dụng công nghệ ASP.NET Core và Entity Framework để xây dựng Web API cho ứng dụng IoT

RESTful API

- REST = **RE**presentational **S**tate **T**ransfer: Các trạng thái tài nguyên được truyền tải qua HTTP
- Tiêu chuẩn dùng trong thiết kế Web API

RESTful API

- Dựa trên giao thức HTTP. Sử dụng các phương thức http
  - GET: Lấy về (READ) một tài nguyên cụ thể (by Id) hoặc một danh sách tài nguyên (Resource).
  - POST: Tạo mới (CREATE) một tài nguyên.
  - PUT: Cập nhật (UPDATE) thông tin cho 1 tài nguyên (by Id).
  - DELETE: Xóa (DELETE) 1 tài nguyên (by Id).

Tạo Web API project

- Open File/New/Project in Visual Studio

https://medium.com/net-core/how-to-build-a-restful-api-with-asp-net-core-fb7dd8d3e5e3

Tạo Web API project

- Chọn .NET Core, ASP.NET Core 2.2, API Project

Tạo Web API project

- Project WebAPIDemo

Tạo Web API project

- MVC (Model-View-Controller) được thêm trong Startup.cs

```
namespace WebAPIDemo
{
    2 references
    public class Startup
    {
        0 references | 0 exceptions
        public Startup(IConfiguration configuration)...
        1 reference | 0 exceptions
        public IConfiguration Configuration { get; }
        // This method gets called by the runtime. Use this method to add services to the container.
        0 references | 0 exceptions
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
        }
        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        0 references | 0 exceptions
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())...
            else...
            app.UseHttpsRedirection();
            app.UseMvc();
        }
    }
}
```

Routing and URL paths

- Run app (Ctrl + F5), sử dụng IIS Express (MS webserver)
- Kết quả demo trên trình duyệt

Routing and URL paths

- URL mặc định https://localhost:{port}/api/values được thiết lập trong thuộc tính launchUrl trong file ***Properties\launchSettings.json***
- Values nhận được trên trình duyệt được mặc định trong phương thức Get của ValuesController

launchSettings.json | Startup.cs | Program.cs

Schema: http://json.schemastore.org/launchsettings.json

```
1  {
2    "$schema": "http://json.schemastore.org/launchsettings.json",
3    "iisSettings": {
4      "windowsAuthentication": false,
5      "anonymousAuthentication": true,
6      "iisExpress": {
7        "applicationUrl": "http://localhost:54911",
8        "sslPort": 44382
9      }
10   },
11   "profiles": {
12     "IIS Express": {
13       "commandName": "IISExpress",
14       "launchBrowser": true,
15       "launchUrl": "api/values",
16       "environmentVariables": {
17         "ASPNETCORE_ENVIRONMENT": "Development"
18       }
19     },
20     "MovieAPI": {
21       "commandName": "Project",
22       "launchBrowser": true,
23       "launchUrl": "api/values",
24       "applicationUrl": "https://localhost:5001;http://localhost:5000",
25       "environmentVariables": {
26         "ASPNETCORE_ENVIRONMENT": "Development"
27       }
28     }
29   }
30 }
```

```
[Route("api/[controller]")]
[ApiController]
public class ValuesController : ControllerBase
{
    // GET api/values
    [HttpGet]
    public ActionResult<IEnumerable<string>> Get()
    {
        return new string[] { "value1", "value2" };
    }
```

Tạo một model

- Xây dựng data model class (**Model** trong kiến trúc MVC)
- In ***Solution Explorer***, right-click the project.
  Select ***Add*** > ***New Folder*** and name the folder ***Models***.
- Then right-click the ***Models*** folder and select ***Add->Class***. Name the class ***SensorData.cs*** and click Add.

Tạo một model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace WebAPIDemo.Models
{
    public class SensorData
    {
        public int Id { get; set; }
        [Required]  //Data Annotation
        [StringLength(60, MinimumLength = 3)]
        public string Name { get; set; }
        public float Value { get; set; }
        [DataType(DataType.Date)]
        public DateTime ReceiveTime { get; set; }
    }
}
```

Tạo một controller

- Tạo một controller (thành phần Controller trong mô hình MVC), gắn với SensorData model
- Sử dụng Entity Framework Core (EF Core) để làm việc với database

Tạo một controller

Tạo một controller

- Công cụ ***scaffolding*** của .NET Core hỗ trợ tự động sinh code
- The automatic creation of the database context and **CRUD** (**C**reate, **R**ead, **U**pdate, and **D**elete) action methods is known as scaffolding.

Controller Class

- Ví dụ SensorDatasController class với các methods:
  - GetSensorData (GET method)
  - PostSensorData (POST method)
  - PutSensorData (PUT method)
  - DeleteSensorData (DELETE method)
- Controller class kế thừa từ ControllerBase

```
public class SensorDatasController : ControllerBase
```

SensorDatasController.cs

```
namespace WebAPIDemo.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference | 0 requests
    public class SensorDatasController : ControllerBase
    {
        private readonly WebAPIDemoContext _context;
        0 references | 0 exceptions
        public SensorDatasController(WebAPIDemoContext context)...
        // GET: api/SensorDatas
        [HttpGet]
        0 references | 0 requests | 0 exceptions
        public async Task<ActionResult<IEnumerable<SensorData>>> GetSensorData()...
        // GET: api/SensorDatas/5
        [HttpGet("{id}")]
        0 references | 0 requests | 0 exceptions
        public async Task<ActionResult<SensorData>> GetSensorData(int id)...
        // PUT: api/SensorDatas/5
        [HttpPut("{id}")]
        0 references | 0 requests | 0 exceptions
        public async Task<IActionResult> PutSensorData(int id, SensorData sensorData)...
        // POST: api/SensorDatas
        [HttpPost]
        0 references | 0 requests | 0 exceptions
        public async Task<ActionResult<SensorData>> PostSensorData(SensorData sensorData)...
        // DELETE: api/SensorDatas/5
        [HttpDelete("{id}")]
        0 references | 0 requests | 0 exceptions
        public async Task<ActionResult<SensorData>> DeleteSensorData(int id)...
        1 reference | 0 exceptions
        private bool SensorDataExists(int id)...
    }
}
```

EF Core Database Context

- ***Data\WebAPIDemoContext.cs***
- ***WebAPIDemoContext*** kết nối với các chức năng của EF Core (Create, Read, Update, Delete, etc.) cho **SensorData** model
- ***WebAPIDemoContext*** kế thừa từ Microsoft.EntityFrameworkCore.DbContext.

```
using Microsoft.EntityFrameworkCore;
namespace WebAPIDemo.Models
{
    5 references
    public class WebAPIDemoContext : DbContext
    {
        0 references | 0 exceptions
        public WebAPIDemoContext (DbContextOptions<WebAPIDemoContext> options)...
        6 references | 0 exceptions
        public DbSet<WebAPIDemo.Models.SensorData> SensorData { get; set; }
    }
}
```

EF Core Database Context

- In Entity Framework terminology, an entity set typically corresponds to a database table and an entity corresponds to a row in the table.
- ASP.NET Core configuration system reads the connection string from the ***appsettings.json*** file:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "WebAPIDemoContext": "Server=(localdb)\\mssqllocaldb;Database=WebAPIDemoContext-56113fd8-
  }
}
```

EF Core Database Context

- Đăng ký DB Context (WebAPIDemoContext) trong Startup.cs

```
// This method gets called by the runtime. Use this method to add services to the container.
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);

    services.AddDbContext<WebAPIDemoContext>(options =>
            options.UseSqlServer(Configuration.GetConnectionString("WebAPIDemoContext")));
}
```

Inject the database context (WebAPIDemoContext) into the controller:

```
namespace WebAPIDemo.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference | 0 requests
    public class SensorDatasController : ControllerBase
    {
        private readonly WebAPIDemoContext _context;
        0 references | 0 exceptions
        public SensorDatasController(WebAPIDemoContext context)
        {
            _context = context;
        }
```

Tạo database sử dụng Migrations

- Sử dụng Entity Framework code first
- Công cụ Migrations tạo (create) database tương ứng với data model được xây dựng (code first) và cập nhật (update) database schema khi có thay đổi của data model
- Chạy các lệnh của công cụ Migrations:
  - Mở ***Tools -> NuGet Package Manager > Package Manager Console (PMC), thực hiện command:***
  - **Add-Migration Initial (Tạo database)**
  - **Update-Database (Cập nhật database schema)**

Tạo database sử dụng Migrations

- Migration files được sinh tự động trong Migration folder

```
namespace WebAPIDemo.Migrations
{
    1 reference
    public partial class Initial : Migration
    {
        0 references | 0 exceptions
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "SensorData",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn),
                    Name = table.Column<string>(maxLength: 60, nullable: false),
                    Value = table.Column<float>(nullable: false),
                    ReceiveTime = table.Column<DateTime>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_SensorData", x => x.Id);
                });
        }
        0 references | 0 exceptions
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "SensorData");
        }
    }
}
```

Tạo database sử dụng Migrations

- Chạy lệnh **Update-Database** (trong PMC)
- Phương thức Up trong file ***Migrations/{time-stamp}_Initial.cs*** sẽ được thực thi và tạo database
- Mở SQL Server Object Explorer

Tạo database

- Thêm dữ liệu vào bảng SensorData
- **Mở *SQL Server Object Explorer***, chuột phải vào bảng SensorData, chọn ***View Data*** để thêm bản ghi

dbo.SensorData [Data]

Max Rows: 1000

Id	Name	Value	ReceiveTime
1	Temperature	37	4/8/2021 11:1...
2	Humidity	66	4/8/2021 10:3...
4	PM2.5	123	4/9/2021 12:0...
6	PM10	345	4/9/2021 1:20...
NULL	NULL	NULL	NULL

GET method

- GetSensorData(): trả về tất cả bản ghi trong SensorData database
- GetSensorData(int id): trả về bản ghi Id tương ứng
- [HttpGet] phương thức xử lý khi có HTTP GET request

```
// GET: api/SensorDatas
[HttpGet]
0 references | 0 requests | 0 exceptions
public async Task<ActionResult<IEnumerable<SensorData>>> GetSensorData()
{
    return await _context.SensorData.ToListAsync();
}
// GET: api/SensorDatas/5
[HttpGet("{id}")]
0 references | 0 requests | 0 exceptions
public async Task<ActionResult<SensorData>> GetSensorData(int id)
{
    var sensorData = await _context.SensorData.FindAsync(id);

    if (sensorData == null)
    {
        return NotFound();
    }

    return sensorData;
}
```

GET method

- GET endpoints:
  - GET /api/SensorData
  - GET /api/SensorData/{id}
- Gọi HTTP GET request từ trình duyệt:
  - https://localhost:{port}/api/SensorDatas
  - https://localhost:{port}/api/SensorDatas/2
- Kiểu giá trị trả về **ActionResult<T> type**
  - .NET Core tự động serializes object thành JSON và ghi vào body của gói tin response
  - Respone code: **200** (nếu không có unhandled exceptions), nếu có sẽ trả về mã lỗi **5xx**

GET method

- Kiểm thử trên trình duyệt

localhost:44364/api/SensorDatas

```
// 20210409113750
// https://localhost:44364/api/SensorDatas

[
  {
    "id": 1,
    "name": "Temperature",
    "value": 37.0,
    "receiveTime": "2021-04-08T11:16:20"
  },
  {
    "id": 2,
    "name": "Humidity",
    "value": 66.0,
    "receiveTime": "2021-04-08T10:30:01"
  },
```

localhost:44364/api/SensorDatas/2

```
// 20210409113838
// https://localhost:44364/api/SensorDatas/2

{
  "id": 2,
  "name": "Humidity",
  "value": 66.0,
  "receiveTime": "2021-04-08T10:30:01"
}
```

GET method

- URL mặc định: Cấu hình launchUrl trong file launchsettings.json

launchSettings.json

Schema: http://json.schemastore.org/launchsettings.json

```
7          "applicationUrl": "http://localhost:54420",
8          "sslPort": 44364
9        }
10     },
11     "profiles": {
12       "IIS Express": {
13         "commandName": "IISExpress",
14         "launchBrowser": true,
15         "launchUrl": "api/SensorDatas",
16         "environmentVariables": {
17           "ASPNETCORE_ENVIRONMENT": "Development"
18         }
19       },
20       "WebAPIDemo": {
21         "commandName": "Project",
22         "launchBrowser": true,
23         "launchUrl": "api/values",
24         "applicationUrl": "https://localhost:5001;http://localhost:5000",
25         "environmentVariables": {
26           "ASPNETCORE_ENVIRONMENT": "Development"
27         }
28       }
29     }
30   }
```

POST method

- Hàm PostSensorData tạo một bản ghi mới trong database
- Dữ liệu (bản ghi) được gửi trong body của HTTP POST request
- Hàm CreatedAtAction:
  - Trả về mã HTTP 201 nếu thành công

```
// POST: api/SensorDatas
[HttpPost]
0 references | 0 requests | 0 exceptions
public async Task<ActionResult<SensorData>> PostSensorData(SensorData sensorData)
{
    _context.SensorData.Add(sensorData);
    await _context.SaveChangesAsync();

    return CreatedAtAction("GetSensorData", new { id = sensorData.Id }, sensorData);
}
```

Kiểm thử dùng Postman

- GET method: get all resource

GET https://localhost:44364/api/SensorDatas Send Save

Params

Query Params

KEY	VALUE	DESCRIPTIC
Key	Value	Description

Bulk Edit

Body 200 OK 188 ms 692 B Save Response

Pretty JSON


```
[
    {
        "id": 1,
        "name": "Temperature",
        "value": 37.0,
        "receiveTime": "2021-04-08T11:16:20"
    },
    {
        "id": 2,
        "name": "Humidity",
        "value": 66.0,
        "receiveTime": "2021-04-08T10:30:01"
    },
```

Kiểm thử dùng Postman

- GET method: get 1 resource by Id

GET https://localhost:44364/api/SensorDatas/2 Send Save

Params

Query Params

	KEY	VALUE	DESCRIPTIC
	Key	Value	Description

Bulk Edit

Body 200 OK 114 ms 394 B Save Response

Pretty JSON


```
{
    "id": 2,
    "name": "Humidity",
    "value": 66.0,
    "receiveTime": "2021-04-08T10:30:01"
}
```

Kiểm thử dùng Postman

- POST method

POST https://localhost:44364/api/SensorDatas Send Save

Body raw JSON Beautify

```
1 {
2     "name": "Light",
3     "value": 203,
4     "receiveTime":
          "2021-04-10T12:17:20"
5 }
```

Body 201 Created 1339 ms 446 B Save Response

Pretty JSON

```
1 {
2     "id": 7,
3     "name": "Light",
4     "value": 203.0,
5     "receiveTime": "2021-04-10T12:17:20"
6 }
```

PUT method

- Phương thức PutSensorData() cập nhật một bản ghi

```
// PUT: api/SensorDatas/5
[HttpPut("{id}")]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> PutSensorData(int id, SensorData sensorData)
{
    if (id != sensorData.Id)
    {
        return BadRequest();
    }
    _context.Entry(sensorData).State = EntityState.Modified;
    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!SensorDataExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return NoContent();
}
```

PUT method

- Mã trả về: HTTP 204 (No Content)

PUT https://localhost:44364/api/SensorDatas/7 Send Save

Body raw JSON Beautify

```
1 {
2     "id":7,
3     "name": "Light",
4     "value": 111,
5     "receiveTime":
          "2021-04-10T12:17:20"
6 }
```

Body 204 No Content 1137 ms 252 B Save Response

Pretty Text

```
1
```

DELETE method

- Xóa một bản ghi (by Id)

```
// DELETE: api/SensorDatas/5
[HttpDelete("{id}")]
0 references | 0 requests | 0 exceptions
public async Task<ActionResult<SensorData>> DeleteSensorData(int id)
{
    var sensorData = await _context.SensorData.FindAsync(id);
    if (sensorData == null)
    {
        return NotFound();
    }

    _context.SensorData.Remove(sensorData);
    await _context.SaveChangesAsync();

    return sensorData;
}
```

DELETE method

- Xóa một bản ghi (by Id)
- Mã trả về HTTP OK 200 (xóa thành công)

3.2.2. Công nghệ NodeJS, MongoDB

- MEAN stack (mongoDB, ExpressJS, AngularJS, NodeJS)

https://viblo.asia/p/xay-dung-restful-api-don-gian-voi-nodejs-1Je5EdewlnL

https://viblo.asia/p/tao-rest-api-voi-spring-boot-mysql-jpa-repository-Eb85oJykl2G

3.3. Sử dụng các dịch vụ IoT Cloud

- Các dịch vụ:
  - Tiếp nhận dữ liệu (Collecting, Ingesting data)
  - Lưu trữ dữ liệu (Cloud Storage, Database)
  - Trình bày dữ liệu (Data Representation),
  - Theo dõi giám sát, điều khiển (Dashboard)
- Một số nền tảng dịch vụ IoT Cloud:
  - IBM BlueMix
  - AWS IoT
  - Google Cloud IoT
  - Azure IoT

3.3.1. Sử dụng IBM IoT Cloud

https://cloud.ibm.com/catalog/services/internet-of-things-platform#about

IBM Cloud | Search resources and offerings... | Catalog | Docs | Support | Manage | Pham Ngoc Hu...

Catalog / Services /

Internet of Things Platform

IBM • **Date of last update:** 09/22/2020 • Docs

Summary

This service is the hub for IBM Watson IoT and lets you communicate with and consume data from connected devices and gateways. Use the built-in web console dashboards to monitor your IoT data and analyze it in real time. Then, enhance and customize your IBM Watson IoT Platform experience by building and connecting your own apps by using messaging and REST APIs.

- Giao tiếp, thu thập dữ liệu từ các thiết bị (connected devices, gateways)
- Giám sát, phân tích dữ liệu IoT theo thời gian thực (Sử dụng built-in web console dashboards)
- Hỗ trợ phát triển apps sử dụng cơ chế messaging và REST APIs

IBM IoT Platform

- Các tính năng (Features):
  - Connect: Đăng ký, kết nối đến các devices, sensors, gateways
  - Information Management: Quản lý lữu trữ dữ liệu (data storage), chuyển đổi dữ liệu, tích hợp với các dịch vụ dữ liệu khác
  - Analyze in real time: Giám sát, phân tích dữ liệu theo thời gian thực
  - Risk and Security Management

IBM IoT Platform

- Kiến trúc các dịch vụ IBM IoT Platform

Tutorials: Sử dụng IBM IoT platform

- 1. Đăng ký dịch vụ
  - Create an IBM Internet of Things Platform service

1. Open a browser and sign in to one of your team's IBM Cloud accounts at https://cloud.ibm.com.
   Your Apps dashboard opens.
2. Click the **Create resource** button.

3. Click the Internet of Things Platform service.

Tip: To quickly find the service, use the search bar's type-ahead feature. Be sure to click the IoT Platform service, not the IoT Platform boilerplate.

4. Type a name for your instance of the IoT Platform service. A default name is suggested, but you can replace it. Although you can name the service anything, in the examples in this course, the service name is *myWorkshop-IoT*.
5. Click **Create**. After a moment, the Service Details page of your new IoT Platform service is shown.

Tutorials: Sử dụng IBM IoT platform

- 2. Đăng ký, kết nối thiết bị, gửi dữ liệu đến IBM IoT platform

Tutorials: Sử dụng IBM IoT platform

- Đăng ký thiết bị

Tutorials: Sử dụng IBM IoT platform

Lập trình thiết bị ESP32 gửi dữ liệu tới IBM IoT platform (1)

https://developer.ibm.com/recipes/tutorials/connect-an-esp32-to-the-watson-iot-platform/

```
#include <WiFi.h>
#include <PubSubClient.h>
//-------- Customise these values -----------
const char* ssid = ”DESKTOP-JD1VOFG 3221”
const char* password = “hung1234";

#define ORG "quickstart" // your organization or "quickstart"
#define DEVICE_TYPE “Collect_Device"
#define DEVICE_ID “esp32id1"
#define TOKEN "ckN_nGfj9jEzOpLOCB" //device token registered to IBM
//-------- Customise the above values --------

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[] = "iot-2/evt/status/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
```

Tutorials: Sử dụng IBM IoT platform

Lập trình thiết bị ESP32 gửi dữ liệu tới IBM IoT platform (2)

```
WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);
void setup() {
 Serial.begin(115200); delay(1); Serial.println();
 initWiFi();
}
void loop() {
 if (!!!client.connected()) {
   Serial.print("Reconnecting client to "); Serial.println(server);
   while (!!!client.connect(clientId, authMethod, token)) {
     Serial.print(".");  delay(500);
   }
   Serial.println();
 }
 String payload = "{ \"data\" : {\"counter\":";
 payload += millis()/1000;
 payload += "}}";
 Serial.print("Sending payload: "); Serial.println(payload);
 if (client.publish(topic, (char*) payload.c_str())) {
   Serial.println("Publish ok");
 } else { Serial.println("Publish failed"); } delay(3000);
}
```

Tutorials: Sử dụng IBM IoT platform

Lập trình thiết bị ESP32 gửi dữ liệu tới IBM IoT platform (3)

```
void initWiFi() {
 Serial.print("Connecting to "); Serial.print(ssid);
 if (strcmp (WiFi.SSID().c_str(), ssid) != 0) {
   WiFi.begin(ssid, password);
 }
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 Serial.println("");
 Serial.print("WiFi connected, IP address: ");
 Serial.println(WiFi.localIP());
}
```

Chú ý

- Chọn Security level trên IBM IoT platform; sử dụng TLS Optional

Default Rule

Define the default connection security level to use for all device types that do not have custom rules defined.

Scope	Security Level	# of Devices
Default	TLS Optional	0 devices

Custom Rules

Tutorials: Sử dụng IBM IoT platform

- Dữ liệu nhận từ thiết bị theo thời gian thực trên IBM IoT platform

Device ID	Status	Device Type
esp32id1	Connected	Collect_Device

Identity | Device Information | Recent Events | State | Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
status	{"data":{"counter":35,"temperature":25,"humidit...	json	a few seconds ago
status	{"data":{"counter":30,"temperature":25,"humidit...	json	a few seconds ago
status	{"data":{"counter":25,"temperature":25,"humidit...	json	a few seconds ago

Tutorials: Sử dụng IBM IoT platform

- Xây dựng ứng dụng bằng Node-RED
  - https://developer.ibm.com/recipes/tutorials/getting-started-with-watson-iot-platform-using-node-red/

Tutorials: Sử dụng IoT platform services

- Tạo cơ sở dữ liệu dùng dịch vụ Db2 Warehouse
  - Task 3: Create a Db2 Warehouse on Cloud service
  - Task 4: Create a Db2 Warehouse table for environment data

Tutorials: Sử dụng IoT platform services

- Tạo một application sử dụng Node-RED
  - Task 5: Create a IBM Cloud Node-RED application space
  - Task 6: Create the Raspberry Pi Node-RED flows
  - Task 7: Create the IBM Cloud Node-RED flows
  - Task 8: Deploy and validate

3.3.2. Sử dụng Google Cloud IoT

- A tutorial: https://medium.com/google-cloud/build-a-weather-station-using-google-cloud-iot-core-and-mongooseos-7a78b69822c5

Nội dung

- Chương 1. Tổng quan về IoT
- Chương 2. Các công nghệ IoT
- Chương 3. Lập trình ứng dụng IoT
- Chương 4. An toàn và Bảo mật IoT
- Chương 5. Thiết kế và xây dựng hệ thống IoT

Chương 4. An toàn và Bảo mật IoT

- 4.1. Tổng quan về bảo mật IoT
- 4.2. Các dạng tấn công vào hạ tầng IoT
- 4.3. Các điểm yếu bảo mật trong IoT

4.1. Tổng quan về bảo mật IoT

- Các vấn đề trong bảo mật IoT:
  - Thiết kế ban đầu cho mạng truyền thông riêng sau đó được chuyển sang mạng IP và Internet
  - Cập nhật firmware cho thiết bị IoT khó khăn
  - Xuất phát từ những yêu cầu bảo mật cơ bản, sau đó xuất hiện các lỗi bảo mật kèm theo các yêu cầu bảo mật phức tạp hơn.
  - Các thiết bị bảo mật kém từ các thiết kế ban đầu đã được sử dụng trên thực tế

Tổng quan về bảo mật IoT

- Phân loại nguy cơ trong bảo mật IoT:
  - **Capture:** Các nguy cơ liên quan đến “bắt” (thu thập, ăn cắp) thông tin dữ liệu từ hệ thống
  - **Disrupt:** Các nguy cơ liên quan đến tấn công từ chối dịch vụ (denying), phá hủy (destroying), ngắt/dừng (interrupting) hệ thống
  - **Manipulate:** Các nguy cơ liên quan đến can thiệp/thay đổi (manipulating) dữ liệu, định danh.

Tổng quan về bảo mật IoT

- Các yêu cầu bảo mật IoT:
  - Confidentiality – Tính tin cẩn:
    - Dữ liệu truyền chỉ có thể được đọc bởi bên nhận
  - Availability – Tính sẵn dùng:
    - Việc truyền thông giữa các thiết bị truyền và nhận luôn luôn sẵn sàng
  - Integrity – Tính toàn vẹn
    - Dữ liệu nhận không bị nhiễu trong quá trình truyền, đảm bảo tính chính xác, vẹn toàn của dữ liệu
  - Authenticity – Tính xác thực
    - Bên gửi luôn luôn có thể xác thực dữ liệu được gửi
    - Dữ liệu chỉ có thể được truy cập bởi bên nhận được cho phép

4.2. Các dạng tấn công hạ tầng IoT

- Hạ tầng IoT thông dụng

Một số dạng tấn công

- Fake Control Server

Một số dạng tấn công

- Attack on device open ports

Một số dạng tấn công

- Attack on server open ports

Một số dạng tấn công

- Steal Credential

Một số dạng tấn công

- Inject bad configuration or firmware

Một số dạng tấn công

- Sniff data on private network

4.3. Các điểm yếu bảo mật IoT

- I1. Insecure Web Interface (Giao diện quản trị không an toàn)
- I2. Insufficient Authentication/Authorization (Xác thực không đủ an toàn)
- I3. Insecure Network Services (Các dịch vụ mạng thiếu bảo mật)
- I4. Lack of Transport Encryption/Integrity Verifcation (Thiếu mã hóa tầng giao vận)
- I5. Privacy Concerns (Các vấn đề liên quan quyền riêng tư)
- I6. Insecure Cloud Interface (Giao diện Cloud thiếu bảo mật)
- I7. Insecure Mobile Interface (Giao diện mobile thiếu bảo mật)
- I8. Insufficient Security Configurability (Cấu hình bảo mật không an toàn)
- I9. Insecure Software/Firmware (Phần mềm không an toàn)
- I10. Poor Physical Security (Thiếu bảo mật tầng vật lý)

1. Giao diện quản trị không an toàn

2. Cơ chế xác thực không an toàn

3. Các dịch vụ mạng không an toàn

4. Thiếu sử dụng mã hóa tầng giao vận

5. Các vấn đề về quyền riêng tư

6. Giao diện Cloud không an toàn

7. Giao diện Mobile không an toàn

8. Thiếu cấu hình bảo mật

9. Phần mềm không an toàn

10. Thiếu bảo mật tầng vật lý

Nội dung

- Chương 1. Tổng quan về IoT
- Chương 2. Các công nghệ IoT
- Chương 3. Lập trình ứng dụng IoT
- Chương 4. An toàn và Bảo mật IoT
- Chương 5. Thiết kế và xây dựng hệ thống IoT

Chương 5. Thiết kế và xây dựng hệ thống IoT

- Project-Based Learning:
  - Học dựa trên giải quyết một bài toán/dự án IoT cụ thể
- Kỹ năng:
  - Tìm vấn đề cần giải quyết (Finding the problems)
  - Phân tích thiết kế (System design and Analysis)
  - Xây dựng hệ thống (Implementing, Programming)
  - Trình bày (Presentation, Report)
  - Làm việc nhóm (Teamwork)

Cách thức thực hiện

- Lập nhóm dự án: 3-4 thành viên
- Xác định bài toán ứng dụng IoT
- Phân tích và thiết kế hệ thống IoT:
  - Thiết kế kiến trúc tổng quan của hệ thống
  - Thiết kế chi tiết các thành phần của hệ thống
  - Lựa chọn thiết bị phần cứng: máy tính nhúng, sensors
  - Lựa chọn giải pháp phần mềm/công nghệ IoT
- Xây dựng hệ thống:
  - Lập trình triển khai các thành phần của hệ thống
  - Vận dụng các giải pháp, dịch vụ, công nghệ IoT
- Thuyết trình và báo cáo

Các chủ đề ứng dụng IoT

- P1. Smart Agriculture:
  - Ứng dụng IoT trong nông nghiệp thông minh

P1. Smart Agriculture

https://www.smartofthings.co.th/2018/08/27/smart-agriculture-solution/

P2. Smart Home

- Ứng dụng IoT trong nhà thông minh:
  - Điều khiển các thiết bị từ xa (mobile app)
  - Điều khiển bằng giọng nói (Google assistant)
  - Giám sát từ xa

P2. Smart Home

- Ví dụ:

Các chủ đề ứng dụng IoT

- P3. Smart Water Supply Monitoring:
  - Ứng dụng IoT trong cấp nước thông minh
- P4. Smart Electrical Power Monitoring:
  - Ứng dụng IoT trong giám sát phân phối điện năng thông minh

Smart Utitlies (Electrical power, Water)

https://www.smartofthings.co.th/2018/08/27/smart-utilities-solution/

IoT for Smart Grids

The paradigm of IoT in smart electrical grids

https://www.researchgate.net/publication/331103655_IoT_Architecture_for_Smart_Grids

Các chủ đề ứng dụng IoT

- P5. Smart Transportation:
  - Ứng dụng IoT trong giám sát vận tải thông minh

P5. Smart Transportation

Các chủ đề ứng dụng IoT

- 6. Smart Air Quality Monitoring:
  - Ứng dụng IoT trong giám sát chất lượng không khí

P6. Smart Air Quality Monitoring

Các chủ đề ứng dụng IoT

- 7. Smart Gabaging:
  - Ứng dụng IoT trong thu gom rác thông minh
  - Tham khảo: https://www.instructables.com/id/Smart-Garbage-Monitoring-System-Using-Internet-of-/
- 8. Smart Parking:
  - Ứng dụng IoT trong đỗ xe thông minh
- 9. Smart Health Monitoring:
  - Ứng dụng IoT trong giám sát sức khỏe thông minh
- 10. Smart Robotic Warehouse:
  - Ứng dụng IoT trong nhà kho thông minh

Project 1: Smart Gabage

- Tổng quan hệ thống

Project 1: Smart Gabage

- Web app quản lý: thêm/sửa thùng rác

Project 1: Smart Gabage

- Mobile app

Project 1: Smart Gabage

- Mobile app

Project 2. Smart Home

- Sơ đồ thiết kế tổng quan

Project 2. Smart home

- Mobile app:
  - Quản lý home/rooms (add/edit)

Project 2. Smart home

- Mobile app:
  - Quản lý home/rooms (add/edit)

Project 2. Smart home

- Mobile app:
  - Add thiết bị, ghép nối thiết bị vật lý

Project 2. Smart home

- Mobile app:
  - Automation

Project 2. Smart home

- Xây dựng server:

Project 2. Smart home

- Xây dựng server:
  - Luồng nghiệp vụ thêm/xóa người dùng

Thêm/xóa người dùng

Project 2. Smart home

- Xây dựng server:
  - Luồng nghiệp vụ thêm nhà

Thêm một ngôi nhà mới

Project 3. Smart home

- Thiết kế hệ thống

Project 3. Smart home

- Mobile app

Project 4. Smart garden

- Kiến trúc hệ thống

Project 4. Smart garden

- Công nghệ sử dụng:
  - Sử dụng dữ liệu ảo cho các thiết bị cảm biến.
  - Server: NodeJS, ExpressJS.
  - Database: MongoDB.
  - Website: React JS.
  - Mobile App: React Native.

Project 4. Smart garden

- Thiết kế CSDL

Project 4. Smart garden

- Xây dựng các API
  - Đăng ký: POST /v1/auth/register
  - Đăng nhập: POST /v1/auth/login
  - Lấy danh sách các thiết bị: GET /v1/device/{houseId}
  - Sửa thông tin thiết bị:PUT /v1/device/{deviceId}
  - Lấy dữ liệu thống kê: GET /v1/data/{houseId}
  - Lấy danh sách các cảnh báo: GET /v1/house/events/{houseId}
  - Lấy thông tin nhà nấm: GET /v1/house
  - Thêm nhà nấm mới: POST /v1/house
  - Sửa thông tin nhà nấm: PUT /v1/house/update-house/{houseId}
  - Thêm thành viên: PUT /v1/house/update-member/{houseId}
  - Xóa thành viên: PUT /v1/house/delete-member/{houseId}/{userId}

Project 4. Smart garden

- Ví dụ API lấy ds thiết bị

GET http://localhost:9000/v1/device/63e8a7ab1d624a76b5d91255 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (8) Test Results Status: 200 OK Time: 233 ms Size: 81.27 KB Save Response

Pretty Raw Preview Visualize JSON


```
1   {
2       "success": true,
3       "devices": [
4           {
5               "_id": "63de1380dd8742bc1ba5390a",
6               "name": "ND-01",
7               "active": true,
8               "spaceX": 100,
9               "spaceY": 0,
10              "spaceZ": 0,
11              "data": 21.8,
12              "created_at": "2023-02-04T08:12:48.776Z",
13              "__v": 0,
14              "delay": 2,
15              "status": {…
18              },
19              "updatedAt": "2023-02-19T03:53:20.523Z",
20              "history": […
1113            ],
1114            "max": 30,
1115            "min": 15,
1116            "type_name": "Temp"
1117        },
1118        {
1119            "_id": "63de139ddd8742bc1ba5390f",
1120            "name": "ND-02",
1121            "active": true,
```

Project 4. Smart garden

- Web app quản trị:

Project 4. Smart garden

- Web app quản trị:

Project 4. Smart garden

- Web app quản trị:

Project 4. Smart garden

- Mobile app