พัฒนา IoT บนไมโครคอนโทรลเลอร์ ESP32 ด้วยภาษาไมโครไพทอน

สุดยอดการเรียนรู้การพัฒนา IoT

- ผ่านการสื่อสารแบบไวไฟ (Wi-Fi) บลูทูท (Bluetooth) และลอรา (LoRa)
- การใช้งานโพรโทคอล MQTT, CoAP และเว็บซ็อกเก็ต (Web Socket)
- การจัดเก็บข้อมูลไปยังฐานข้อมูล และแสดงผลบนโปรแกรมกราฟานา
- พัฒนาระบบจริงและการปรับปรุงการทำงานผ่าน OTA

รศ.ดร. ชัชชัย คุณบัว

ซีเอ็ด

พัฒนา IoT บนไมโครคอนโทรลเลอร์ ESP32 ด้วยภาษาไมโครไพทอน

โดย รศ.ดร. ชัชชัย คุณบัว

ข้อมูลทางบรรณานุกรมของหอสมุดแห่งชาติ

ชัชชัย คุณบัว.

พัฒนา IOT บนไมโครคอนโทรลเลอร์ ESP32 ด้วยภาษาไมโครไพทอน.-- กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2566.
352 หน้า.

1. อินเทอร์เน็ตในทุกสิ่ง. 2. ไมโครคอนโทรลเลอร์. 3. ไพธอน (ภาษาคอมพิวเตอร์).
I. ชื่อเรื่อง.

004.678

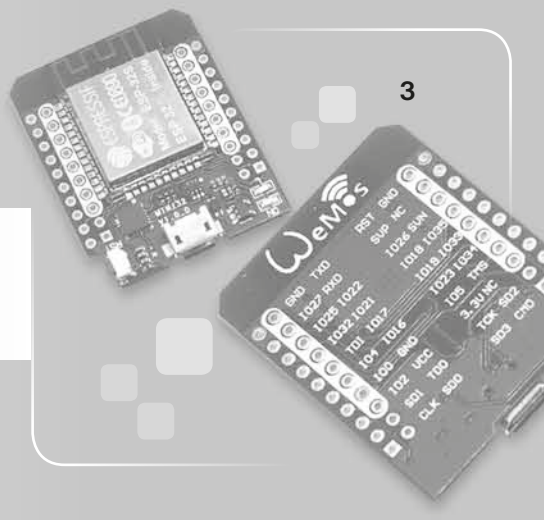
Barcode (e-book) : 978-616-08-5011-2

ผลิตและจัดจำหน่ายโดย

บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)
SE-EDUCATION PUBLIC COMPANY LIMITED

เลขที่ 1858/87-90 ถนนเทพรัตน แขวงบางนาใต้ เขตบางนา กรุงเทพฯ 10260
โทรศัพท์ 0-2826-8000

[หากมีคำแนะนำหรือติชม ติดต่อที่ comment@se-ed.com]



คำนำ

การพัฒนาอุปกรณ์บนเทคโนโลยี IoT ได้รับความนิยมเพิ่มขึ้นอย่างมาก เพื่อนำไปประยุกต์ใช้ในงานด้านต่าง ๆ และมีแนวโน้มว่าจะมีจำนวนที่สูงขึ้น ตามแนวโน้มของความต้องการและการพัฒนาของเทคโนโลยี ซึ่งปัจจุบันพบว่ามีโมดูลไมโครคอนโทรลเลอร์ เซนเซอร์ รวมไปถึงโปรแกรมแบบโอเพนซอร์ซจำนวนมากที่สามารถนำมาใช้เพื่อการพัฒนา

ภาษาไมโครไพทอนพัฒนาจากภาษาไพทอนที่ได้รับความนิยมอย่างสูง เนื่องจากสามารถทำความเข้าใจได้ง่ายและมีผู้ใช้งานอย่างแพร่หลาย การพัฒนาไมโครไพทอนมีจุดประสงค์เพื่อใช้งานบนไมโครคอนโทรลเลอร์ขนาดเล็ก รวมถึงไมโครคอนโทรลเลอร์ ESP32 ที่มีราคาไม่สูง สามารถจัดหาได้ง่าย

หนังสือเล่มนี้เขียนขึ้นเพื่อมุ่งเน้นการใช้โมดูลไมโครคอนโทรลเลอร์ ESP32 และการใช้งานพื้นฐานร่วมกับเซนเซอร์ต่าง ๆ โดยตั้งเป้าหมายเพื่อให้ผู้อ่านเข้าใจถึงการพัฒนา การเลือกใช้ และการประยุกต์ใช้เซนเซอร์รูปแบบต่าง ๆ ไปถึงจนการทำงานร่วมกับแอกทูเอเตอร์ และเข้าใจถึงการสื่อสารแบบไวไฟ (Wi-Fi) และบลูทูทแบบพลังงานต่ำ (BLE) บนโมดูล ESP32 ไปถึงการใช้งานแอปพลิเคชันโพรโทคอลสำหรับการสื่อสารผ่านคลาวด์ ได้แก่ โพรโทคอล MQTT และโพรโทคอล CoAP และการทำงานของเว็บแอปพลิเคชันผ่านเว็บซ็อกเก็ต (Web Socket) รองรับการสื่อสารแบบสองทาง (Bi-directional) ที่มีเวลาแฝงต่ำ (Low-latency) นอกจากการสื่อสารแบบไวไฟและบลูทูทแบบพลังงานต่ำ ภายในหนังสือเล่มนี้ ยังได้นำเสนอการเชื่อมต่อโมดูล ESP32 กับโมดูลลอรา (LoRa) สำหรับการสื่อสารไร้สายแบบพื้นที่กว้างพลังงานต่ำของเทคโนโลยี IoT

การพัฒนาเทคโนโลยี IoT นอกจากรูปแบบการสื่อสารที่มีประสิทธิภาพ การจัดเก็บข้อมูลเป็นอีกหนึ่งปัจจัยสำคัญเพื่อการวิเคราะห์ด้านต่าง ๆ ดังนั้นผู้เขียนยังได้อธิบายและยกตัวอย่างการจัดเก็บบนฐานข้อมูลแบบเรียลไทม์ของไฟร์เบส (Firebase Realtime Database) และ

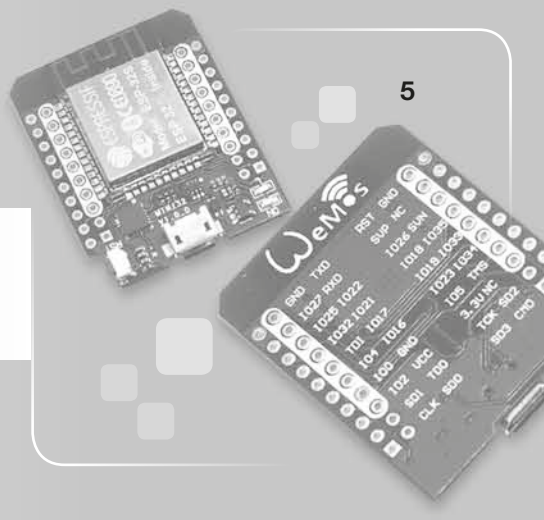
การพัฒนาส่วนต่อประสานโปรแกรมประยุกต์ (API) เพื่อจัดเก็บไปยังฐานข้อมูล MySQL พร้อมนำผลการแสดงผลแบบเรียลไทม์บนแดชบอร์ด (Dashboard) ด้วยโปรแกรมกราฟานา (Grafana) สุดท้ายการปรับปรุงโปรแกรมแบบออนไลน์ผ่านกระบวนการที่เรียกว่า โอเวอร์ดิแอร์ (Over the Air; OTA)

หนังสือเล่มนี้ เหมาะสำหรับนิสิต นักศึกษา และผู้สนใจ การพัฒนาด้านเทคโนโลยี IoT ผู้เขียนหวังว่าหนังสือเล่มนี้ จะเป็นจุดเริ่มต้นสำหรับผู้สนใจที่ต้องการเริ่มพัฒนาแอปพลิเคชันบนเทคโนโลยี IoT แบบต่างๆ ขึ้นเอง ตั้งแต่เริ่มต้นจนกระทั่งการนำไปใช้งานจริง โดยแม้ว่าการพัฒนาหลายส่วนที่เกิดขึ้นอาศัยข้อมูลจากกิตฮับ (Github) โดยผู้เขียนได้พยายามเพิ่มเติมด้านทฤษฎีที่เกี่ยวข้อง เพื่อให้เข้าใจถึงที่มาและการนำไปประยุกต์ใช้ เช่นเดียวกับหลายท่านได้กล่าวว่า กูเกิล (Google) เปรียบเหมือนพจนานุกรมสามารถเปิดหานิยามของคำและสิ่งต่างๆ ได้ แต่การจะเรียบเรียงประโยคที่สวยงามต้องอาศัยประสบการณ์และการเรียนรู้ การพัฒนาด้านเทคโนโลยีก็เช่นเดียวกันต้องอาศัยความรู้ ความเข้าใจขององค์ประกอบต่างๆ ที่จะนำมาใช้ เพื่อให้ได้การพัฒนาเทคโนโลยีที่ถูกต้องและมีประสิทธิภาพ หรือแม้แต่การใช้ปัญญาประดิษฐ์ (Artificial Intelligence) การนำมาใช้หรือต่อยอด ย่อมมาจากความรู้พื้นฐานที่ดี

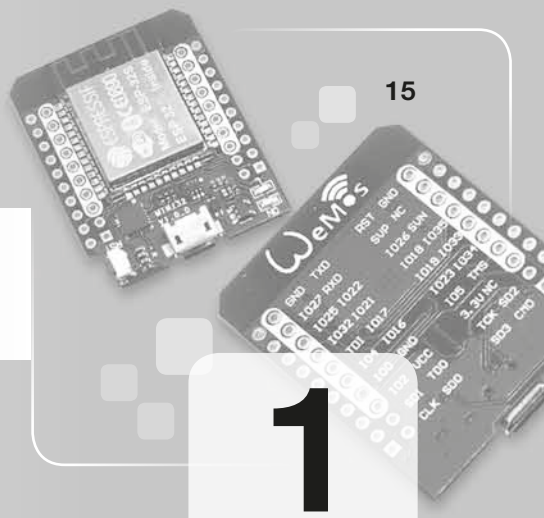
ขอขอบคุณรองศาสตราจารย์ ดร. ธนภัทร์ อนุศาสน์อมรกุลสำหรับคำแนะนำและแก้ไขหนังสือ คุณอรลักษณ์ บริมาสพร บริษัทไลท์ติ้ง แอนด์ อีควิปเมนท์ จำกัด (มหาชน) สำหรับโมดูลลอราเพื่อทดสอบ คุณกฤษฎากร หาดวรรณ์สำหรับคำแนะนำเว็บแอปพลิเคชัน คุณวิภาดา สีหพันธ์และคุณอิสยาภรณ์ ประสานกุลนันท์สำหรับภาพประกอบ คุณภาริดา สีหามาตย์สำหรับคลิปประกอบ นักศึกษาคณะวิศวกรรมศาสตร์ นายวรายุทธ ภูมิวัฒฑราคานนท์ นายธนชัย แก้วแสน นายปะริญัตติ์ ลินทะละ นายตรีเพชร ตรีจันทร์ นายกาล ซาเหลา นายเกียรติพันธ์ วารีรักษ์ นายกิตินันท์ กุณโฮง นายเสฏฐวุฒิ นัตธิลม นายกฤษณะ พุ่มพยอม Mr. Thavrak Chan และ Mr. Ruben James van Wyk สำหรับการทดสอบ ปรับปรุง และแก้ไขตัวอย่างโค้ดในหนังสือ และสุดท้ายคณาจารย์และเจ้าหน้าที่สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น ที่ให้การสนับสนุนจนหนังสือเล่มนี้สำเร็จได้ด้วยดี

หนังสือเล่มนี้ ตั้งใจเขียนขึ้นเพื่อให้ผู้อ่านใช้ประโยชน์สำหรับการพัฒนาเทคโนโลยี IoT โดยข้ามทฤษฎีเชิงลึกบางส่วน สำหรับผู้สนใจสามารถดูรายละเอียดเพิ่มเติมจากหนังสือ IoT สถาปัตยกรรมการสื่อสาร Internet of Things ทั้งนี้ผู้อ่านสามารถดาวน์โหลดโค้ดตัวอย่างในหนังสือได้ที่ https://github.com/ckboa/esp32_IoT หากมีข้อผิดพลาดประการใดผู้เขียนขออภัยมา ณ ที่นี้ และสามารถส่งแจ้งมายัง chatchai@kku.ac.th เพื่อปรับปรุงต่อไป ขอบคุณครับ

รองศาสตราจารย์ ดร. ชัชชัย คุณบัว



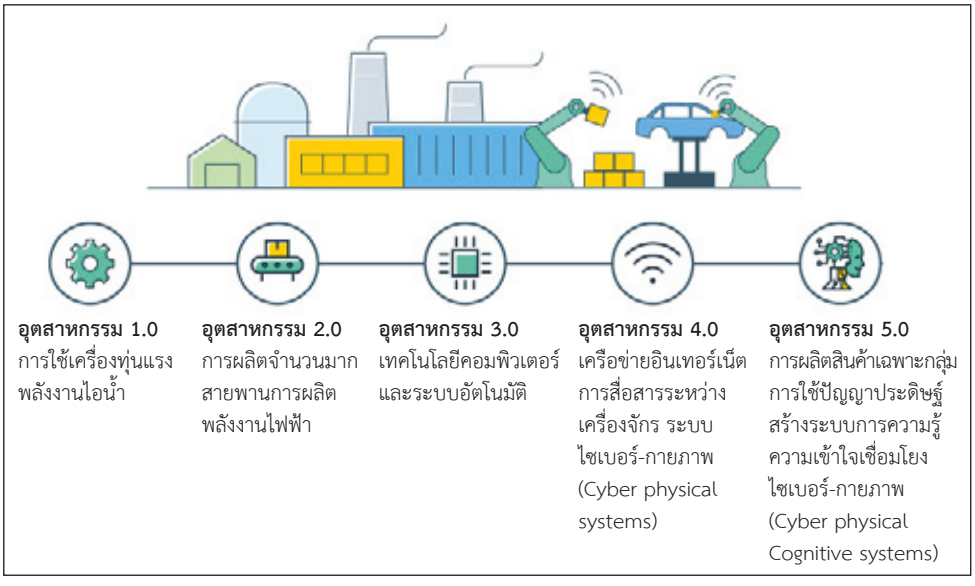
สารบัญ



บทนำ

1

อินเทอร์เน็ตของสรรพสิ่ง (Internet of Things; IoT) ถูกใช้งานอย่างแพร่หลายในปัจจุบัน ในยุคอุตสาหกรรม 4.0 โดยมีเป้าหมายสำคัญ ได้แก่ การปรับปรุงการเชื่อมต่อ (Improved Connectivity) การปรับกระบวนการให้เหมาะสม (Optimize Processes) การเพิ่มการประสานงาน (Increase Coordination) และสร้างธุรกิจใหม่ ทำให้การขับเคลื่อนด้วยเทคโนโลยี IoT ถือเป็นรากฐานสำคัญของการพัฒนาด้านต่าง ๆ ในยุคอุตสาหกรรม 4.0 เช่น การวิเคราะห์ข้อมูลตามเวลาจริง การควบคุมคุณภาพ และการคาดการณ์การบำรุงรักษาเครื่องจักร เป็นต้น และยังนำไปสู่การใช้งานในอุตสาหกรรมยุค 5.0 เพื่อให้เกิดปฏิสัมพันธ์ระหว่างมนุษย์และเครื่องจักรเพิ่มมากขึ้น ดังแสดงในรูปที่ 1.1

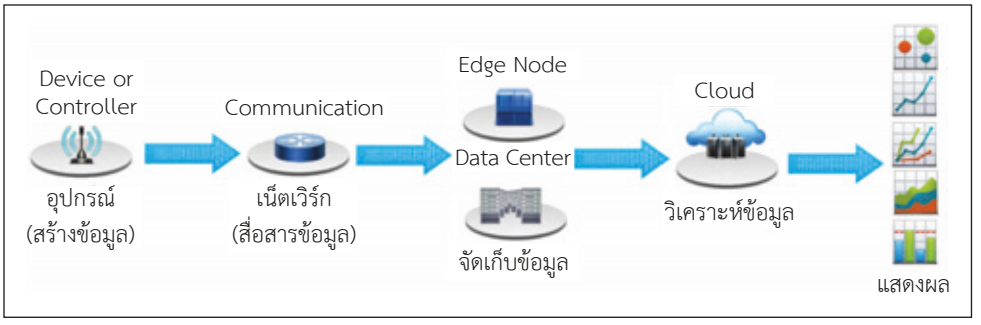


รูปที่ 1.1 การพัฒนาจากอุตสาหกรรมยุค 1.0 ไปสู่ 5.0 [3]

แม้ว่าการใช้งาน IoT จะได้รับการตอบรับในวงกว้าง แต่การพัฒนา IoT ยังถือว่ามีข้อจำกัด เนื่องจากผู้พัฒนาไม่เข้าใจภาพรวมของระบบ การเลือกใช้เซนเซอร์ ความเข้าใจถึงโพรโทคอลสำหรับการสื่อสาร ไปจนถึงการจัดเก็บและแสดงผล ส่งผลให้การพัฒนาที่เกิดขึ้นไม่เป็นไปตามจุดประสงค์ที่ต้องการ ดังนั้นเพื่อตอบสนองต่อความต้องการของ IoT ที่เพิ่มสูงขึ้นและการต่อยอดไปสู่การพัฒนาด้านต่าง ๆ ในอนาคต ผู้เขียนจึงได้เลือกไมโครคอนโทรลเลอร์ ESP32 ที่ได้รับความนิยมอย่างมาก มีราคาในช่วงที่สามารถยอมรับได้ สามารถรองรับการสื่อสาร IoT พื้นฐาน ได้แก่ ไวไฟ (Wi-Fi) และ บลูทูท (Bluetooth) ใช้ร่วมกับภาษาไมโครไพทอน ซึ่งมีพื้นฐานและรูปแบบการพัฒนาเช่นเดียวกับภาษาไพทอนที่ได้รับความนิยมอย่างมากสำหรับการพัฒนาการเรียนรู้ด้วยเครื่อง (Machine Learning; ML) และปัญญาประดิษฐ์ (Artificial Intelligence; AI) โดยมุ่งเน้นความเข้าใจพื้นฐาน ตลอดจนการใช้งานรูปแบบต่าง ๆ นอกจากนี้ผู้เขียนยังได้นำเสนอการใช้งาน ESP32 ร่วมกับโมดูล LoRa เพื่อรองรับการสื่อสาร IoT ที่สามารถครอบคลุมพื้นที่การสื่อสารที่กว้างขึ้น ตอบสนองความต้องการใช้งานในภาคเกษตรและอุตสาหกรรม

1.1 ระบบนิเวศ (Ecosystem) ของเทคโนโลยี IoT

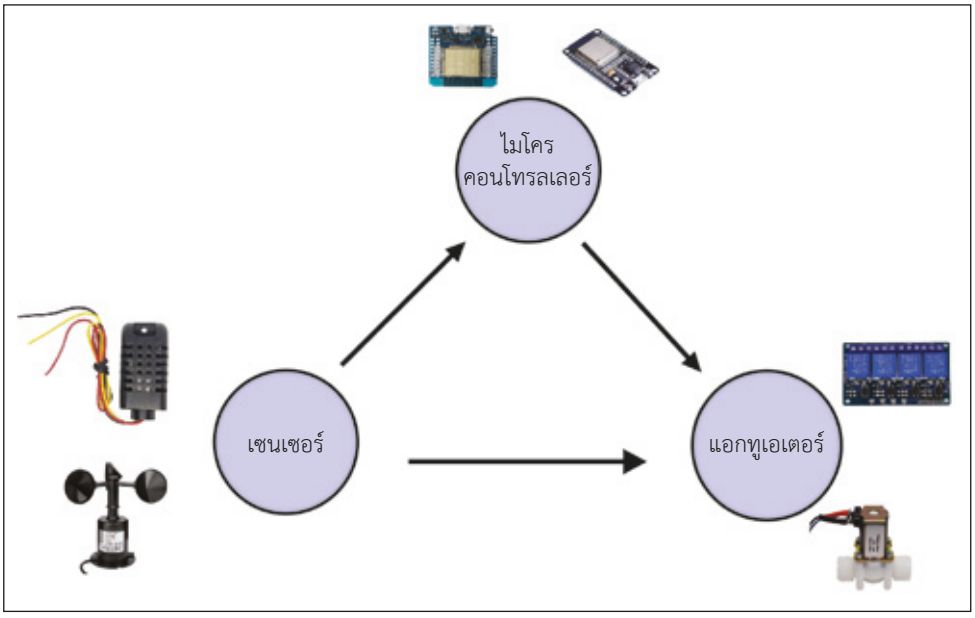
ตอบสนองความต้องการของเทคโนโลยี IoT ทำให้เกิดโมเดลการพัฒนาที่หลากหลาย เช่น โมเดล oneM2M เป็นความร่วมมือขององค์กรสื่อสารระดับโลก เช่น European Telecommunications Standards Institute (ETSI) และ Telecommunications Industry Association (TIA) เพื่อพัฒนาสถาปัตยกรรมร่วมกัน สำหรับรองรับเทคโนโลยีการทำงานระหว่างเครื่องจักรกับเครื่องจักร (Machine-to-Machine; M2M) ในปี พ.ศ. 2555 (ค.ศ. 2012) และโมเดล The IoT World Forum (IoTWF) [4] เกิดจากความร่วมมือของบริษัทซิสโก้ (Cisco) และบริษัทไอบีเอ็ม (IBM) และบริษัทอื่น ๆ ในปี พ.ศ. 2557 (ค.ศ. 2014) อย่างไรก็ตาม หากมองภาพรวมในเชิงระบบการพัฒนาระบบ IoT ประกอบด้วยโครงสร้างหลัก ได้แก่ อุปกรณ์ การสื่อสาร การจัดเก็บ วิเคราะห์ข้อมูล และการแสดงผล ดังแสดงในรูปที่ 1.2



รูปที่ 1.2 ระบบนิเวศของอินเทอร์เน็ตของสรรพสิ่ง (IoT) ดัดแปลงจาก [5]

1.1.1 อุปกรณ์ (Devices)

อุปกรณ์ถือเป็นโครงสร้างพื้นฐานสำคัญของระบบ IoT สามารถแบ่งออก 3 ส่วน ได้แก่ เซนเซอร์ แอกทูเอเตอร์ และไมโครคอนโทรลเลอร์ ดังแสดงในรูปที่ 1.3



รูปที่ 1.3 องค์ประกอบพื้นฐานของระบบ IoT

- **เซนเซอร์** เป็นส่วนสำคัญเพื่อใช้ตรวจสอบสภาวะการทำงานและสภาพแวดล้อมต่างๆ เช่น การใช้เซนเซอร์เพื่อตรวจวัดการสั่นสะเทือนของเครื่องจักร และเซนเซอร์วัดอุณหภูมิและความชื้นในโรงงาน เป็นต้น การเลือกชนิดและประเภทของเซนเซอร์ถือเป็นส่วนสำคัญของการพัฒนา ส่งผลต่อความแม่นยำและความถูกต้องโดยตรง
- **ไมโครคอนโทรลเลอร์** ถือเป็นส่วนสำคัญทำหน้าที่รับข้อมูลต่างๆ จากเซนเซอร์ แปลงค่าข้อมูลให้อยู่ในรูปแบบดิจิทัล การประมวลผลและการตัดสินใจในระบบ และการส่งสัญญาณควบคุมไปยังส่วนของแอกทูเอเตอร์ กล่าวโดยทั่วไปถือเป็นส่วนสมองของระบบ IoT
- **แอกทูเอเตอร์ (Actuator)** เป็นส่วนอุปกรณ์คอยรับคำสั่งจากไมโครคอนโทรลเลอร์ หรือเซนเซอร์โดยตรง เพื่อตอบสนองต่อสภาวะต่างๆ ที่เกิดขึ้น เช่น การใช้โมดูลรีเลย์เพื่อควบคุมการเปิดปิดไฟแสงสว่างภายในโรงงาน หรือการใช้โซลินอยด์วาล์วสำหรับการเปิดปิดน้ำ เป็นต้น

1.1.2 การสื่อสาร

เพื่อตอบสนองความต้องการของการสื่อสารที่หลากหลาย เช่น การสื่อสารเชิงข้อมูลขนาดใหญ่ การสื่อสารแบบประหยัดพลังงาน การสื่อสารที่ต้องการการครอบคลุมพื้นที่กว้าง ทำให้มีการพัฒนาเทคโนโลยีการสื่อสารที่แตกต่างกันตามจุดประสงค์การใช้งานในรูปแบบโพรโทคอลต่าง ๆ โดยสามารถแบ่งออกเป็น 2 กลุ่มหลัก ได้แก่ โพรโทคอลสำหรับการสื่อสารระหว่างอุปกรณ์และโพรโทคอลการสื่อสารร่วมกับระบบคลาวด์

1. โพรโทคอลสำหรับการสื่อสารระหว่างอุปกรณ์ ถือเป็นกลุ่มการสื่อสารพื้นฐานสำหรับอุปกรณ์ IoT โดยทั่วไป หากพิจารณาการสื่อสารที่เกิดขึ้นกับระยะทางเพื่อรองรับการทำงานของ IoT สามารถแบ่งออกเป็นสองกลุ่มหลัก [7] คือ กลุ่มที่สื่อสารระยะสั้นเพื่อใช้เชื่อมต่อระหว่างอุปกรณ์ภายในบริเวณหนึ่งระยะทางไม่เกิน 1,000 เมตร เช่น ไวไฟ บลูทูท และซิกบี (Zigbee) เป็นต้น และกลุ่มการสื่อสารระยะไกลที่มีระยะการสื่อสารมากกว่า 1,000 เมตรขึ้นไป เช่น Narrowband Internet of Things (NB-IoT) และ LoRa (ลอรา) เป็นต้น

- **ไวไฟ (Wi-Fi)** เทคโนโลยีพื้นฐานรองรับการทำงานของอุปกรณ์ IoT เป็นเทคโนโลยีที่ใกล้ตัวมากที่สุด และปัจจุบันคาดว่ามีการนำไปใช้เพื่อจัดเก็บและรวบรวมข้อมูลมากที่สุด เนื่องจากเป็นอุปกรณ์ที่สามารถจัดหาได้ง่าย
- **บลูทูท (Bluetooth)** บลูทูทถูกออกแบบมาเพื่อความสะดวกในการเชื่อมต่อในรูปแบบที่เรียกว่า Wireless Personal Area Network (WPAN) หรือเครือข่ายไร้สายส่วนบุคคล เชื่อมต่อการสื่อสารระหว่างคอมพิวเตอร์และอุปกรณ์รอบข้าง เช่น เมาส์ คีย์บอร์ด ปัจจุบันมีการพัฒนาให้ใช้พลังงานที่ต่ำลง เพิ่มระยะทางการสื่อสารมากขึ้น จนกระทั่งการพัฒนาเพื่อเชื่อมต่อในรูปแบบเมช
- **ซิกบี (Zigbee)** ถูกออกแบบมาเพื่อรองรับการสื่อสารที่ไม่ต้องการแบนด์วิดท์ (Bandwidth) ที่สูง ประหยัดพลังงาน สามารถสื่อสารในระยะที่ไกล เรียกว่า Low-rate Wireless Personal Area Network (LR-WPAN) ซิกบีได้รับความนิยมอย่างมากเพื่อการใช้งานเชื่อมต่อกับเซนเซอร์ประเภทต่าง ๆ สามารถทำงานในรูปแบบที่เป็นเครือข่าย เรียกว่า Wireless Sensor Network (WSN)
- **NB-IoT** เป็นการสื่อสารของข้อมูลผ่านโครงข่ายเซลลูลาร์เพื่อการใช้งานด้าน IoT ที่มีการสื่อสารข้อมูลขนาดเล็ก และการใช้พลังงานที่ต่ำ เพื่อให้รองรับการสื่อสารไร้สายแบบพื้นที่กว้างพลังงานต่ำ (Low Power Wide Area; LPWA)

- **ลอรา (LoRa)** เป็นอีกหนึ่งเทคโนโลยี LPWA เช่นเดียวกับ NB-IoT แต่ NB-IoT สามารถใช้งานบนโครงข่ายระบบเซลลูลาร์ที่ต้องได้รับอนุญาตเท่านั้น ส่วนการสื่อสารของลอรา ผู้ใช้สามารถติดตั้งและใช้งานได้ภายใต้คลื่นความถี่ที่ได้รับอนุญาต

2. โพรโทคอลสำหรับการสื่อสารผ่านคลาวด์ นอกจากโพรโทคอลสำหรับการสื่อสารพื้นฐาน การจัดเก็บรวบรวมข้อมูลพื้นฐานผ่านคลาวด์ถือเป็นสิ่งที่หลีกเลี่ยงไม่ได้ ประกอบไปด้วยโพรโทคอลที่สำคัญ ได้แก่ Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP) และ HyperText Transfer Protocol (HTTP)

- **Message Queue Telemetry Transport (MQTT)** อาศัยหลักการทำงานของพับบลิช/ซับสไครบ์ (Publish/Subscribe) เพื่อลดความยุ่งยากและเพิ่มความยืดหยุ่น (Flexibility) การพัฒนาระบบ การใช้งาน MQTT สามารถรองรับคุณภาพการให้บริการ (Quality of Service, QoS) ได้ถึง 3 ระดับ อันเป็นความโดดเด่นที่สำคัญ
- **Constrained Application Protocol (CoAP)** ถูกออกแบบมาสำหรับอุปกรณ์ IoT ที่มีทรัพยากรจำกัด ด้วยการสื่อสารในรูปแบบคล้ายกับการทำงานของโพรโทคอล HTTP แต่ทำงานอยู่บนโพรโทคอล UDP แทนการทำงานบนโพรโทคอล TCP แบบ HTTP
- **Hypertext Transfer Protocol (HTTP)** เป็นแอปพลิเคชันโพรโทคอลที่ได้รับความนิยมสำหรับการสื่อสารอินเทอร์เน็ต โดยการสื่อสารของโพรโทคอล HTTP เป็นการสื่อสารแบบร้องขอ/ตอบกลับ (Request/Response) ระหว่างไคลเอนต์และเซิร์ฟเวอร์

1.1.3 การจัดเก็บข้อมูล

"Data is the new oil." โดยคุณคลิฟ ฮัมบี (Clive Humby) [8] เป็นการแสดงถึงความสำคัญของข้อมูลที่เกิดขึ้น ส่งผลให้มีความจำเป็นอย่างมากที่การพัฒนา IoT ต้องมีการจัดเก็บข้อมูลอย่างเป็นระบบ เพื่อให้อยู่ในรูปแบบที่สามารถนำไปใช้งานภายหลัง ปัจจุบันกล่าวได้ว่าการจัดเก็บลงในฐานข้อมูล สามารถแบ่งออกเป็น 2 กลุ่มหลัก ได้แก่

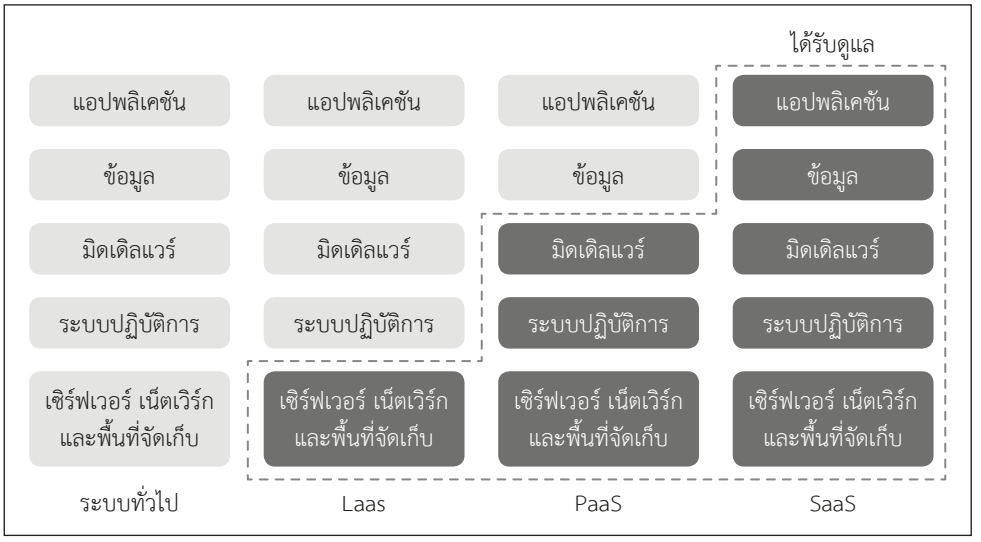
1. ฐานข้อมูลเชิงสัมพันธ์ (Relational Databases) เป็นการจัดเก็บข้อมูลรูปแบบเชิงสัมพันธ์ โดยข้อมูลที่เกิดขึ้นจะถูกจัดเก็บลงในตาราง (Table) ที่ได้รับการกำหนดรูปแบบโครงสร้าง (Schema) ของฐานข้อมูลที่ชัดเจน ความสัมพันธ์ของข้อมูลอาศัยคีย์หลัก (Primary Key) ไม่สามารถซ้ำกันได้ การเข้าถึงฐานข้อมูลอาศัยภาษา Structured Query Language (SQL) เช่น Insert (เพิ่มข้อมูล) Delete (ลบ) หรือ Update (แก้ไข) เป็นต้น ตัวอย่างฐานข้อมูลที่ได้รับความนิยม เช่น MySQL และ Microsoft SQL Server เป็นต้น

2. ฐานข้อมูลไม่เชิงสัมพันธ์ (Non-relational Databases) เป็นรูปแบบการจัดเก็บฐานข้อมูลที่เกิดขึ้นภายหลัง เนื่องจากความหลากหลายของรูปแบบข้อมูล และมีรูปแบบที่ไม่ชัดเจน ทำให้ต้องการรูปแบบการจัดเก็บข้อมูลที่รวดเร็ว สามารถค้นหา รวบรวม และนำไปวิเคราะห์ ตัวอย่างฐานข้อมูลประเภทนี้ที่สำคัญ ได้แก่ MongoDB และ Amazon DynamoDB เป็นต้น

จากการพัฒนาเทคโนโลยีไมโครคอนโทรลเลอร์ที่มีศักยภาพสูงขึ้น นอกเหนือจากการจัดเก็บลงฐานข้อมูลโดยตรงไปยังดาต้าเซ็นเตอร์ ด้วยการพัฒนาให้โหนดที่ใช้มีศักยภาพการประมวลผลที่สูงขึ้น การใช้งานรูปแบบประมวลผลแบบ Edge (Edge Computing) ทำให้ลดความต้องการทรัพยากรด้านต่าง ๆ จากคลาวด์ลง เช่น การใช้พลังงาน ขนาดแบนด์วิดท์ เป็นต้น

1.1.4 คลาวด์

คลาวด์เป็นอีกส่วนสำคัญของการพัฒนา IoT คลาวด์เป็นรูปแบบการให้บริการที่เกิดขึ้นโดยผู้ใช้ไม่จำเป็นต้องติดตั้งหรือดูแลระบบด้วยตนเอง แต่สามารถร้องขอการใช้บริการตามความเหมาะสมภายใต้ข้อตกลงการให้บริการ เช่น การให้บริการของ Amazon Web Services (AWS) หรือการให้บริการ Zoom เป็นต้น โดยทั่วไป การให้บริการคลาวด์ สามารถแบ่งได้เป็น 3 รูปแบบหลัก เปรียบเทียบกับระบบทั่วไปที่ผู้ใช้ติดตั้งระบบเอง ดังแสดงในรูปที่ 1.4 ได้แก่



รูปที่ 1.4 รูปแบบการให้บริการคลาวด์

- **Infrastructure as a Service (IaaS)** เป็นการให้บริการในระดับของอินฟราสตรักเจอร์ โดยที่ผู้ใช้สามารถที่จะติดตั้งระบบปฏิบัติการที่ต้องการและแอปพลิเคชันต่างๆ ตามความต้องการ ผู้ให้บริการทำหน้าที่จัดสรรทรัพยากรตามผู้ใช้ร้องขอ เช่น AWS หรือ DigitalOcean เป็นต้น
- **Platform as a Service (PaaS)** เป็นการให้บริการระดับแพลตฟอร์ม โดยผู้ใช้สามารถจะพัฒนาแอปพลิเคชันและติดตั้งแอปพลิเคชันของตนเอง เช่น Windows Azure และ Heroku เป็นต้น
- **Software as a Service (SaaS)** เป็นการให้บริการซอฟต์แวร์ โดยผู้ใช้สามารถเข้าไปใช้แอปพลิเคชันที่ได้รับการจัดเตรียมไว้จากผู้ให้บริการโดยไม่ต้องติดตั้ง ทำให้ประหยัดเวลาดูแลรักษาระบบ ตัวอย่างเช่น โปรแกรม Zoom และ Dropbox เป็นต้น

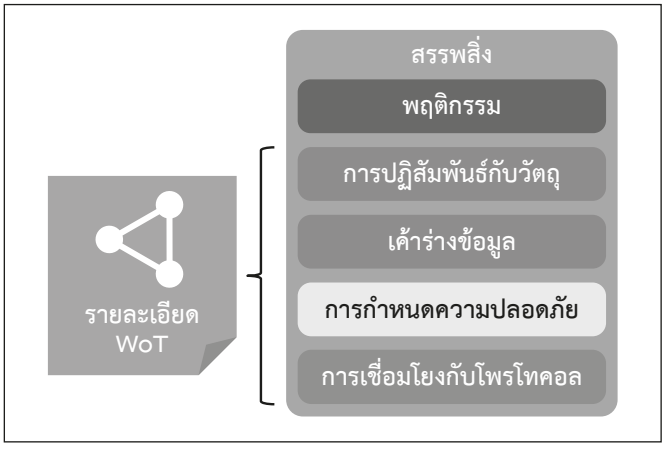
1.1.5 การแสดงผล

การแสดงผลที่เกิดขึ้นทำให้ผู้ใช้งาน IoT สามารถวิเคราะห์ เปรียบเทียบ และนำไปต่อยอดเพื่อเป็นแนวทางสำหรับการพัฒนาด้านอื่นๆ ต่อไป การแสดงผลผ่านแดชบอร์ด (Dashboard) เป็นการแสดงผลที่นิยมในปัจจุบัน สามารถนำผลจากรายงานสำหรับการวิเคราะห์ที่ง่าย และการแสดงผลจากอุปกรณ์ IoT แบบเรียลไทม์ (Real-time) โปรแกรมกราฟานา (Grafana) ถือเป็นโปรแกรมที่ได้รับความนิยมอย่างมาก การแสดงผลที่สวยงาม สามารถใช้งานร่วมกับข้อมูลหลากหลายรูปแบบ เช่น กราฟ (Graph) ชาร์ต (Chart) เป็นต้น ดังแสดงในรูปที่ 1.5

รูปที่ 1.5 ตัวอย่างการแสดงผลของกราฟานา (https://grafana.com/grafana/)

1.2 เว็บของสรรพสิ่ง (Web of Things)

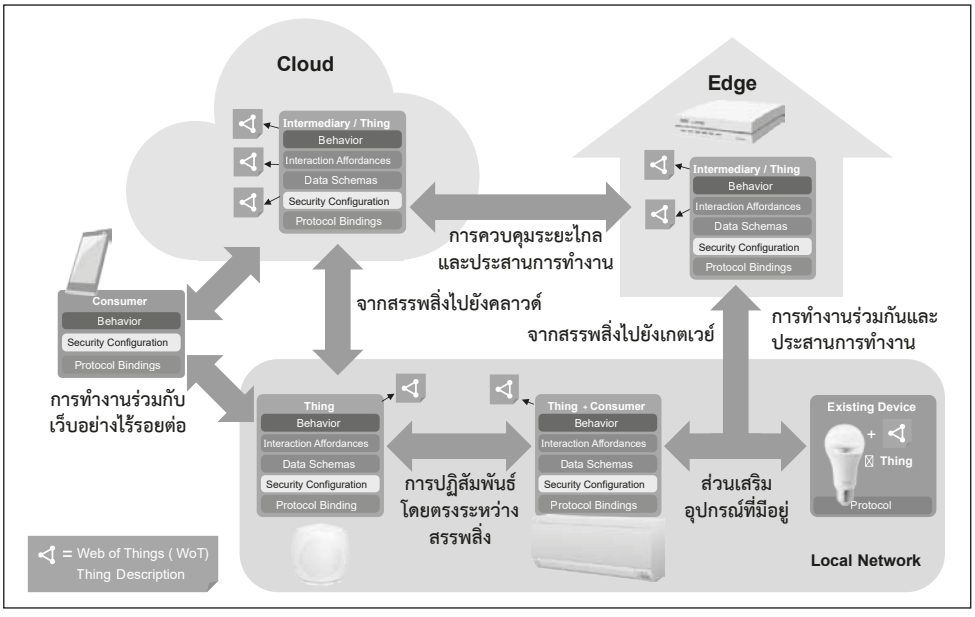
นอกจากการพัฒนาของ IoT ที่เพิ่มมากขึ้น การพัฒนาของเว็บแอปพลิเคชัน ผ่าน HTML ภาษาจาวาสคริปต์ (JavaScipt) หรือเว็บซ็อกเก็ตถูกพัฒนาอย่างต่อเนื่อง ทำให้เกิดการเชื่อมโยงกันระหว่างการใช้งานเว็บแอปพลิเคชันร่วมกับรูปแบบการสื่อสารของ IoT เป็นที่มาของเทคโนโลยี **เว็บของสรรพสิ่ง (Web of Things; WoT)** โดยองค์กรระหว่างประเทศที่ทำงานด้านการพัฒนาเทคโนโลยีเว็บภายใต้ชื่อ World Wide Web Consortium (W3C) โดย WoT มีจุดประสงค์เพื่อให้เกิดการใช้งานร่วมกันของแพลตฟอร์ม IoT และเว็บแอปพลิเคชันโดเมนภายใต้เทคโนโลยีที่มีอยู่ [9] โดยสถาปัตยกรรมของ WoT ประกอบด้วย 4 องค์ประกอบสำคัญ ดังแสดงในรูปที่ 1.6 ได้แก่



รูปที่ 1.6 สถาปัตยกรรมของ W3C WoT [9]

- **พฤติกรรม (Behavior)** เป็นกำหนดการทำงานของอุปกรณ์ โดยอาจอยู่ในรูปแบบอัตโนมัติ และการจัดการผ่านส่วนของ Interaction Affordances
- **การปฏิสัมพันธ์กับวัตถุ (Interaction Affordances)** กำหนดโมเดลที่ผู้ใช้สามารถเข้าถึงอุปกรณ์ได้ภายใต้การทำงานแบบเชิงนามธรรม (Abstract Operations) โดยไม่จำเพาะเน็ตเวิร์กโพรโทคอลหรือการเข้ารหัสข้อมูล (Data Encoding)
- **การกำหนดความปลอดภัย (Security Configuration)** กำหนดการเข้าถึงของการปฏิสัมพันธ์กับวัตถุและการจัดการที่เกี่ยวข้องกับความปลอดภัยของข้อมูล
- **การเชื่อมโยงกับโพรโทคอล (Protocol Bindings)** กำหนดรายละเอียดของการทำงานของส่วนของการปฏิสัมพันธ์กับวัตถุกับโพรโทคอลที่ใช้

โดยหลักการทำงานของ W3C WoT สามารถที่จะใช้ได้กับการทำงานของแอปพลิเคชัน IoT รูปแบบต่างๆ ตั้งแต่ส่วนของอุปกรณ์ ส่วน Edge และส่วนของคลาวด์ ดังแสดงในรูปที่ 1.7



รูปที่ 1.7 โครงสร้างระบบ WoT [9]

ดังนั้นโดยทั่วไปการเกิดขึ้นของ WoT เกิดจากการพัฒนาของเว็บเซิร์ฟเวอร์ขนาดเล็กภายใต้การทำงานของสถาปัตยกรรม (Architecture) อย่าง REST ทำให้สามารถใช้งานเว็บแอปพลิเคชันสะดวกขึ้นภายใต้ URIs และโพรโทคอล HTTP ผ่านการสื่อสารภายใต้ Application Programming Interface (API) เพื่อสื่อสารกับสมาร์ตออบเจ็กต์ต่างๆ การสื่อสารของเว็บที่เกิดขึ้นภายใต้รูปแบบที่เรียกว่า Extensible Markup Language (XML) และ JavaScript Object Notation (JSON) การใช้ XML และ JSON ไม่เพียงแต่สามารถสื่อสารระหว่างเครื่องได้แล้ว ผู้ใช้ยังสามารถเข้าใจทำให้การสื่อสารของสมาร์ตออบเจ็กต์ไม่เพียงแต่สามารถสื่อสารผ่านเว็บ ยังเป็นมิตรกับผู้ใช้ (User-friendly) อีกด้วย

1.3 ไมโครคอนโทรลเลอร์ ESP32

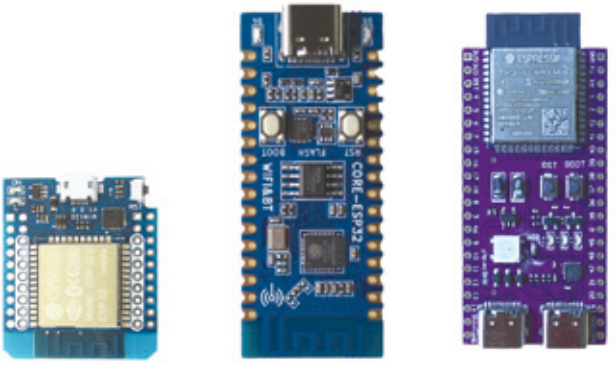
ไมโครคอนโทรลเลอร์ถือเป็นอุปกรณ์สำคัญสำหรับการพัฒนาระบบ IoT เนื่องจากเป็นส่วนที่จัดการประมวลผลข้อมูลต่างๆ ที่ได้รับ การสื่อสารกับอุปกรณ์ภายนอก ไปจนกระทั่งทำหน้าที่เป็น Edge Node/Fog Node ทำให้มีความจำเป็นต้องเข้าใจองค์ประกอบที่สำคัญ ได้แก่ ชนิดของไมโครคอนโทรลเลอร์ที่ต้องการเลือกใช้ การเชื่อมต่อของไมโครคอนโทรลเลอร์และโมดูลที่ใช้ และประสิทธิภาพด้านอื่นๆ ที่สำคัญ ไมโครคอนโทรลเลอร์ ESP32 ที่เลือกใช้ในที่นี้เป็นไมโครคอนโทรลเลอร์กลุ่มที่มีความสามารถประมวลผลในระดับปานกลาง ผู้สนใจสามารถที่จะจัดหาไมโครคอนโทรลเลอร์ประเภทอื่นมาใช้งานตามความเหมาะสม โดยทั่วไปสามารถแบ่งไมโครคอนโทรลเลอร์ออกได้เป็น 3 กลุ่มใหญ่ [10] ได้แก่ กลุ่มความสามารถระดับล่าง (Low-end)

ระดับกลาง (Middle-end) และระดับสูง (High-end)

- **กลุ่มความสามารถระดับล่าง (Low-end IoT Devices)** เป็นอุปกรณ์ที่มีทรัพยากรจำกัด จากคำจำกัดความของ IETF RFC 7228 [11] หมายถึง อุปกรณ์ที่มีความสามารถของซีพียู หน่วยความจำ และพลังงานที่จำกัด ทำให้ไม่สามารถรองรับระบบปฏิบัติการต่างๆ ได้ มีหน่วยความจำหลัก 10-100 กิโลไบต์ (Kilobytes) สถาปัตยกรรมของซีพียูแบบ 8 บิต หรือ 16 บิต อาจมีบางรุ่นที่สามารถทำงานแบบ 32 บิตได้ นิยมใช้เพื่อเชื่อมต่อเซนเซอร์หรือแอกทูเอเตอร์ เช่น บอร์ด Arduino, Tmote Sky เป็นต้น

รูปที่ 1.8 บอร์ด Tmote Sky [12]

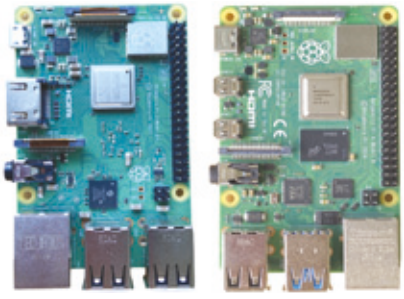
- **กลุ่มความสามารถระดับกลาง (Middle-end IoT Devices)** เป็นอุปกรณ์ที่มีทรัพยากรที่จำกัดระดับหนึ่ง มีความสามารถประมวลผลที่สูง บางโมดูลสามารถทำงานด้านการประมวลผลด้วยภาพได้บางส่วน รวมทั้งรองรับการสื่อสารได้มากกว่าหนึ่งรูปแบบ โดยทั่วไปจะมีความเร็วที่ระดับมากกว่า 100 เมกะเฮิรตซ์ และมีหน่วยความจำระดับกิโลไบต์เทียบกับส่วนของระดับล่างที่ทำงานอยู่ระดับสิบเมกะเฮิรตซ์ ตัวอย่างโมดูลในกลุ่มนี้ ได้แก่ Arduino Yun, ESP8266 และ ESP32 เป็นต้น รูปที่ 1.9 แสดงตัวอย่างโมดูลในกลุ่มนี้



(ก) ESP32-S2 (ข) ESP32-C3 (ค) ESP32-S3

รูปที่ 1.9 บอร์ด ESP32

- **กลุ่มความสามารถระดับสูง (High-end IoT Devices)** เป็นโมดูลที่มีศักยภาพสูง สามารถรองรับระบบปฏิบัติการวินโดวส์หรือลินุกซ์ได้ บางครั้งมีส่วนของหน่วยประมวลผลกราฟิกเฉพาะ (Graphical Processing Unit; GPU) รวมถึงรองรับการประมวลผลด้านการเรียนรู้ด้วยเครื่อง (Machine Learning) ได้ บอร์ด Raspberry Pi เป็นตัวอย่างบอร์ดที่นิยมในกลุ่มนี้ นอกจากนี้กลุ่มนี้ยังนิยมใช้เป็นเกตเวย์เพื่อเชื่อมต่อกับโมดูลระดับล่างและระดับกลาง รูปที่ 1.10 แสดงตัวอย่างโมดูล Raspberry Pi



รูปที่ 1.10 บอร์ด Raspberry Pi 3 โมเดล B+ และ Raspberry Pi 4 โมเดล B

1.3.1 ประสิทธิภาพ ESP32

แม้ว่าในที่นี้จะเลือกใช้ ESP32 เพื่อเป็นองค์ประกอบหลักของระบบ หากพิจารณาจากจำนวนพิน (Pin) ที่มีอยู่ อาจมีเกินความจำเป็น โดยอีกหนึ่งไมโครคอนโทรลเลอร์ที่มีราคาย่อมเยากว่า ได้แก่ ESP 8266 ที่สถาปัตยกรรมใกล้เคียงกัน แต่เนื่องจากจำนวนพินเชื่อมต่อมีอยู่ค่อนข้างจำกัด และการเชื่อมต่ออุปกรณ์ต่าง ๆ ในอนาคตอาจมีปัญหาได้ จึงเลือกใช้ ESP32 ตารางที่ 1.1 เปรียบเทียบประสิทธิภาพและคุณลักษณะของ ESP8266 กับ ESP32

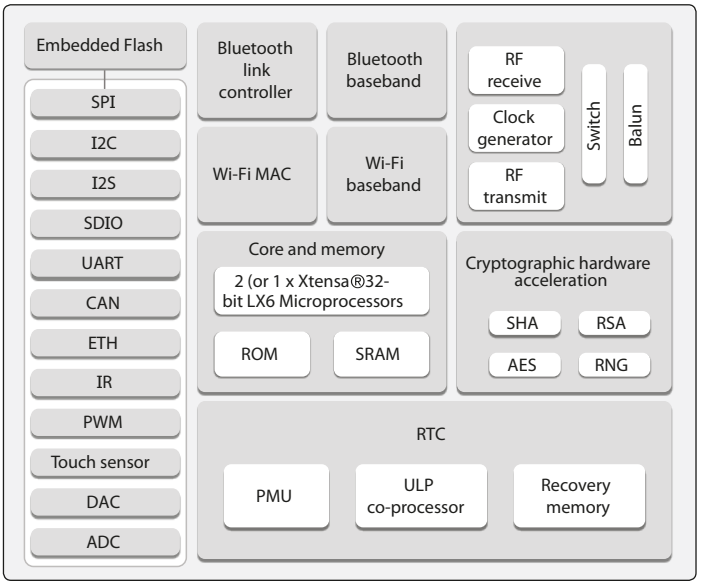
ตารางที่ 1.1 เปรียบเทียบ ESP8266 กับ ESP32

คุณลักษณะ	ESP8266 (ESP8266-12E)	ESP32 (ESP-WROOM-32)
หน่วยประมวลผล	Tensilica LX106 32-bit ที่ 80 MHz (สูงสุดที่ 160 MHz)	Tensilica LX6 32-bit Dual-Core ที่ 160/240 MHz
หน่วยความจำ (SRAM)	32 KB	520 KB
หน่วยความจำ Flash	4 MB (สูงสุดที่ 16 MB)	2 MB (สูงสุดที่ 64 MB)
การสื่อสารไวไฟ (Wi-Fi)	802.11 b/g/n	802.11 b/g/n
การสื่อสารบลูทูท (Bluetooth)	-	Bluetooth 4.2 and BLE
ความถี่	80 MHz	160 MHz
จำนวน GPIO	17	34
Hardware	-	8 channels
จำนวน SPI/I2C/I2S/UART	2/1/2/2	4/2/2/3
ความละเอียดของ ADC	10 - bit	12 - bit
ราคา	100 - 150 บาท	150 - 250 บาท

จากตารางข้างต้น จะเห็นว่าประสิทธิภาพของ ESP32 สูงกว่า ESP8266 ค่อนข้างมาก ตั้งแต่ความเร็ว หน่วยความจำ จำนวนขาเพื่อการเชื่อมต่อ และด้วยราคาที่ต่างกันไม่มาก ทำให้การเลือก ESP32 ถูกมองว่าเป็นทางเลือกที่เหมาะสมกว่า

1.3.2 โครงสร้างทั่วไปของ ESP32

ปัจจุบันไมโครคอนโทรลเลอร์ถือเป็นหัวใจสำคัญของการทำงานด้านระบบสมองกลฝังตัว (Embedded) และ IoT ทำให้มีรูปแบบต่างๆ เกิดขึ้นเป็นจำนวนมาก เพื่อรองรับงานหลากหลายมากขึ้น โดยทั่วไปประกอบด้วยส่วนต่างๆ ได้แก่ หน่วยประมวลผลซีพียู (CPU) หน่วยความจำ (Memory) พอร์ตอินพุตเอาต์พุต (Input/Output Port) และการสื่อสารรูปแบบต่างๆ เช่น ไวไฟ และบลูทูท เป็นต้น โดยรูปที่ 1.11 แสดงสถาปัตยกรรมภายในของ ESP32 โดยมีองค์ประกอบสำคัญ ได้แก่



รูปที่ 1.11 สถาปัตยกรรมภายในของ ESP32 [14]

- **ซีพียู (Central Processing Unit; CPU)** ได้แก่ ส่วนประมวลผลกลาง ทำหน้าที่ประมวลข้อมูลที่ได้รับ ประกอบด้วยองค์ประกอบย่อย ได้แก่ Arithmetic and Logic Unit (ALU) สำหรับการประมวลผลทางคณิตศาสตร์และตรรกศาสตร์ ส่วน Control Unit (CU) เพื่อส่งและรับคำสั่งจาก ALU ส่วนหน่วยความจำเพื่อจัดเก็บและประมวลผล และส่วนรีจิสเตอร์ (Register) ชนิดต่างๆ เพื่อความรวดเร็วของการประมวลผล และสุดท้ายบัส (Bus) ต่างๆ ได้แก่ ดาต้าบัส แอดเดรสบัส และคอนโทรลบัสเพื่อสื่อสารกับส่วนของโพรเซสเซอร์ หน่วยความจำ และอุปกรณ์รอบข้าง

- **หน่วยความจำ (Memory)** โดยทั่วไปไมโครคอนโทรลเลอร์ประกอบด้วยหน่วยความจำ 3 ประเภท ได้แก่ (1) แรม (Ramdom Access Memory; RAM) เป็นหน่วยความจำชั่วคราวเพื่อจัดเก็บข้อมูลระหว่างการประมวลผล ข้อมูลที่จัดเก็บจะหายไปเมื่อปิดเครื่อง (2) รอม (Read Only Memory; ROM) เป็นหน่วยความจำไม่ลบเลือน (Nonvolatile Memory) ที่สามารถอ่านได้เท่านั้น ข้อมูลที่จัดเก็บไม่หายแม้ว่าจะปิดเครื่อง (3) Electrically Erasable Programmable Read-Only Memory (EEPROM) เป็นหนึ่งประเภทของหน่วยความจำแบบไม่ลบเลือน สามารถอ่านเขียนได้ด้วยการจ่ายกระแสไฟฟ้า
- **ไทเมอร์ (Timers) และเคาน์เตอร์ (Counters)** ไมโครคอนโทรลเลอร์จะประกอบด้วยส่วนไทเมอร์และเคาน์เตอร์ เพื่อกำหนดเวลาหน่วงต่าง ๆ และกำหนดการทำงานที่ต้องการ เช่น การทำงานของอินเทอร์รัปต์ (Interrupt) เพื่อแทรกงานที่มีความสำคัญสูงกว่างานที่ทำปกติ และวอตช์ด็อกไทเมอร์ (Watchdog Timer) เพื่อป้องกันไม่ให้โปรแกรมทำงานในสถานะที่ไม่เสถียร (Inconsistent State)
- **อินเทอร์เฟซเพื่อการดีบัก (Debugging Interface)** เนื่องจากอุปกรณ์ฝังตัวไมโครคอนโทรลเลอร์ทั่วไป จะไม่มาพร้อมกับจอแสดงผล คีย์บอร์ด เช่นเดียวกับระบบคอมพิวเตอร์ ทำให้ต้องมีการจัดเตรียมช่องทางการเชื่อมต่อ โดยทั่วไปในรูปแบบฮาร์ดแวร์อินเทอร์เฟซ เพื่อให้ดีบักส่วนของฮาร์ดแวร์และซอฟต์แวร์ของไมโครคอนโทรลเลอร์ได้ เช่น การเชื่อมต่อผ่านคอมพิวเตอร์ผ่านทาง Universal Asynchronous Receiver/Transmitter (UART) หรือการใช้งาน Joint Test Action Group (JTAG) เป็นต้น
- **การเชื่อมต่อสื่อสาร** โมดูล ESP32 รองรับการสื่อสารแบบไวไฟมาตรฐาน 802.11 b/g/n ทำงานในโหมดสเตชัน (การเชื่อมต่อรูปแบบไวไฟที่ใช้ปกติ) และทำหน้าที่เป็นแอ็กเซสพอยต์ เพื่อรองรับการเชื่อมต่อจากอุปกรณ์อื่น นอกจากนี้ยังรองรับการสื่อสารบลูทูทเวอร์ชัน 4.2 และเวอร์ชัน 5.0 (รุ่นใหม่) สำหรับการสื่อสารแบบบลูทูทพลังงานต่ำ

โดยสรุปประสิทธิภาพของ ESP32 ที่สำคัญ ได้แก่

- ซีพียูใช้สถาปัตยกรรม Tensilica LX6 แบบ 2 แกนสมอง สัญญาณนาฬิกา 240 MHz
- มีแรมในตัว 512 KB
- รองรับการเชื่อมต่อรอมภายนอกสูงสุด 16 MB

- มาพร้อมกับไวไฟ มาตรฐาน 802.11 b/g/n รองรับการใช้งานทั้งในโหมดสเตชัน (การเชื่อมต่อรูปแบบไวไฟที่ใช้ปกติ) ทำหน้าที่เป็นแอ็กเซสพอยต์ รองรับการเชื่อมต่อจากอุปกรณ์อื่น
- มีบลูทูทในตัว เวอร์ชัน BLE 4.2

นอกจากรายละเอียดข้างต้นของ ESP32 ทางบริษัทผู้ผลิต ได้แก่ บริษัท Espressif ยังมีพัฒนา ESP32 อย่างต่อเนื่อง โดยทุกรุ่นจะเป็นแบบ System-on-Chip (SoC) มีหน่วยประมวลผลขนาด 32 บิต มีรองรับไวไฟความถี่ 2.4 GHz และบลูทูท โดยแต่ละรุ่นมีความแตกต่างกันดังแสดงในตารางที่ 1.2 [18]

ตารางที่ 1.2 โมดูลตระกูล ESP32

รุ่น	ESP32 Series	ESP32-S2 Series	ESP32-C3 Series	ESP32-S3 Series
ปีที่ออก	2016	2020	2020	2020
Core	Xtensa®dual-/ singlecore 32-bit LX6	Xtensa®single-core 32-bit LX7	32-bit single-core RISC-V	Xtensa®dual-core 32-bit LX7
ไวไฟ	802.11 b/g/n, 2.4 GHz	802.11 b/g/n, 2.4 GHz	802.11 b/g/n 2.4 GHz	802.11 b/g/n, 2.4 GHz
บลูทูท	Bluetooth v4.2 BR/EDR and BLE	X X	Bluetooth 5.0	Bluetooth 5.0
สัญญาณนาฬิกา	240 MHz (160 MHz for ESP32-S0WD)	240 MHz	160 MHz	240 MHz
SRAM	520 KB	320 KB	400 KB	512 KB
ROM	448 KB	128 KB	384 KB	384 KB

1.4 สรุป

การพัฒนา IoT ถือว่ามีความสำคัญอย่างมากสำหรับการพัฒนาเทคโนโลยีอุตสาหกรรม 4.0 เพื่อนำไปสู่อุตสาหกรรม 5.0 ต่อไปในอนาคต โดยในบทนี้ได้นำเสนอระบบนิเวศที่สำคัญตั้งแต่ส่วนของอุปกรณ์ การสื่อสาร การจัดเก็บข้อมูล คลาวด์ และการแสดงผล เพื่อให้ผู้อ่านได้เห็นภาพรวมของระบบ IoT ที่เกิดขึ้น รวมไปถึงการความสัมพันธ์ของ IoT กับสถาปัตยกรรม WoT ภายใต้องค์กร W3C พร้อมองค์ประกอบที่สำคัญของ WoT นอกจากนี้ยังได้กล่าวถึงไมโครคอนโทรลเลอร์ ESP32 ซึ่งเป็นโมดูลหลักสำหรับการใช้งานและพัฒนา IoT ในหนังสือเล่มนี้

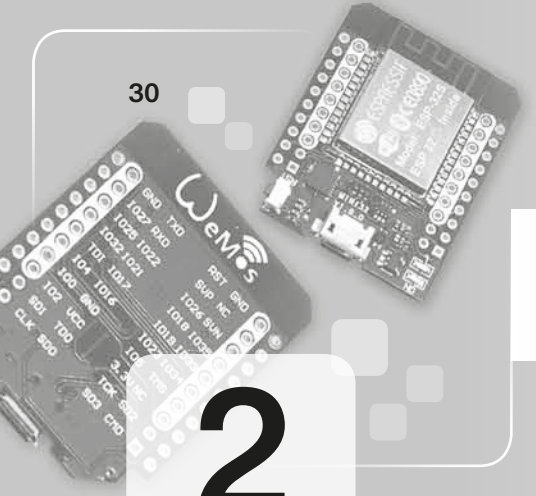
1.5 คำถาม

1. ระบบนิเวศของ IoT ประกอบด้วยอะไรบ้าง

2. WoT มีความสำคัญสำหรับการสื่อสารในปัจจุบันอย่างไร

3. โมดูล ESP32 ประกอบด้วยการสื่อสารแบบใดบ้าง

4. จงเปรียบเทียบประสิทธิภาพของ ESP32 แต่ละรุ่นมีจุดเด่นที่แตกต่างกันอย่างไร



2 ไพทอน (Python)

ไพทอนเป็นหนึ่งภาษาที่ได้รับความนิยมอย่างมากในปัจจุบัน ตั้งแต่การใช้งานพื้นฐานต่างๆ ด้านวิทยาศาสตร์ข้อมูล (Data Science) ไปจนถึงการใช้งานเพื่อพัฒนาเว็บไซต์ ทำให้ภาษาไพทอนมีโมดูลจำนวนมากให้ผู้ที่สนใจสามารถนำมาประยุกต์ใช้และง่ายต่อการเรียนรู้ โดยไพทอนเป็นภาษาที่มีรูปแบบการแปลภาษาแบบอินเทอร์พรีเตอร์ (Interpreter) หมายถึงการประมวลผลไปทีละบรรทัด ทำให้ผู้ใช้สามารถทดสอบคำสั่งบรรทัดต่อบรรทัด แตกต่างจากภาษาแบบคอมไพเลอร์ (Compiler) ทั่วไป เช่น ภาษาซี (C Programming) ที่ต้องแปลงเป็นภาษาเครื่องก่อน ทำให้ภาษาไพทอนสามารถทดสอบคำสั่งได้ง่ายกว่า นอกจากนี้ภาษาไพทอนยังรองรับการกำจัดข้อมูลในหน่วยความจำที่ไม่ได้ใช้อัตโนมัติ (Garbage Collection)

การจัดกลุ่มคำสั่งถือเป็นส่วนสำคัญสำหรับพัฒนาโปรแกรมให้เป็นไปตามวัตถุประสงค์ของการใช้งาน การเขียนภาษาไพทอนอาศัยการจัดกลุ่มคำสั่งของโปรแกรมด้วยการย่อหน้าคล้ายการเขียนไฟล์เอกสาร ทำให้การอ่านโปรแกรมที่เขียนได้ง่าย แต่ก็แลกกับความผิดพลาดที่อาจเกิดขึ้นได้ง่ายเช่นกัน หากการย่อหน้ากลุ่มคำสั่งของโปรแกรมไม่ถูกต้อง ดังนั้นการใช้งานภาษาไพทอนผู้อ่านควรใช้โปรแกรมเอดิเตอร์ (Editor) ที่รองรับการจัดย่อหน้าของภาษา เพื่อลดความผิดพลาดที่อาจเกิดขึ้น เช่น การใช้โปรแกรมทอนนี (Thonny, https://thonny.org/) และไมโครซอฟท์วิชวลโค้ด (VS Code, https://code.visualstudio.com/) เป็นต้น โดยในบทนี้จะได้สรุปพื้นฐานภาษาไพทอน ตั้งแต่ไวยากรณ์ การกำหนดการทำงานแบบมีเงื่อนไข การกำหนดการทำงานซ้ำหลายครั้ง ไปจนกระทั่งพื้นฐานการเขียนฟังก์ชัน และอื่นๆ ที่สำคัญ เพื่อนำไปใช้สำหรับการพัฒนาภาษาไมโครไพทอนต่อไป

2.1 ไวยากรณ์สำคัญ

ไวยากรณ์สำคัญ มีดังนี้

- **การตั้งชื่อตัวแปรภาษาไพทอน** ชื่อตัวแปรตัวใหญ่และตัวเล็กถือว่าเป็นคนละตัวแปร เช่น abc, aBC และ ABC เป็นต้น อย่างไรก็ตาม การตั้งชื่อตัวแปรควรสื่อถึงการใช้งาน เพื่อให้สามารถเข้าใจได้เมื่อกลับมาอ่านอีกครั้ง และห้ามใช้คีย์เวิร์ด (Keyword) เป็นตัวแปร ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 คีย์เวิร์ดในภาษาไพทอน

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	Import	pass	
break	except	in	raise	

- **การเยื้องย่อหน้า (Indentation)** ภาษาไพทอนอาศัยการเยื้องย่อหน้าเพื่อแสดงขอบเขตกลุ่มคำสั่ง เช่น กรณีการเขียนฟังก์ชันใหม่ของผู้ใช้ หรือการใช้ฟังก์ชันต่าง ๆ ของภาษาไพทอน เช่น if, for และ while เป็นต้น ผู้เขียนควรใช้ช่องว่าง (Space) หรือแท็บ (Tabs) อย่างใดอย่างหนึ่งเท่านั้นตลอดโปรแกรม

```
def on_relay(relay):
    relay.value(0)
def off_relay(relay):
    relay.value(1)
```

- **การเขียนคำสั่งเกิน 1 บรรทัด** เมื่อจำเป็นต้องเขียนคำสั่งที่มีความยาวมาก ๆ ไม่สามารถจบใน 1 บรรทัดได้ ให้ใช้เครื่องหมาย \ ตามด้วยการกดขึ้นบรรทัดใหม่ (Enter)
- **เครื่องหมายอัญประกาศ** ไพทอนใช้เครื่องหมายอัญประกาศเดี่ยว (' , Single Quote) และอัญประกาศ" (", Double Quote) เป็นการกำหนดค่าของอักขระหรือสตริง (String) เช่น 'khon Kean' หรือ "khon Kaen" เป็นต้น สำหรับการต่อเชื่อมสตริงแบบหลายบรรทัดจะใช้เครื่องหมายอัญประกาศ 3 ตัวติดกัน (""", Triple Quote)

- **คอมเมนต์ (Comment)** สำหรับบรรทัดที่ไม่ต้องการให้ทำงาน หรือสำหรับบันทึกข้อมูลต่าง ๆ เพื่อกลับมาดูย้อนหลัง ให้ใส่เครื่องหมายสี่เหลี่ยมหน้าบรรทัดนั้น ๆ

```
# ... partial code
i2c = SoftI2C(scl=Pin(22), sda=Pin(21))
display = ssd1306.SSD1306_I2C(128, 64, i2c)
# ...
```

2.2 การใช้งานโอเปอเรเตอร์ (Operator)

การทำงานของภาษาไพทอนรองรับการใช้งานโอเปอเรเตอร์ต่าง ๆ ได้แก่ ด้านคณิตศาสตร์ ด้านตรรกศาสตร์ รวมถึงการทำงานระดับบิต โดยส่วนนี้ขอเน้นเฉพาะ บางส่วนที่จำเป็นสำหรับการพัฒนาระบบเท่านั้น ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 ตัวอย่างการทำงานของภาษาไพทอน

ประเภท	โอเปอเรเตอร์	หน้าที่	ตัวอย่าง	ผลลัพธ์
คณิตศาสตร์	+	บวก	15+10	25
	-	ลบ	15-10	5
	*	คูณ	15*10	150
	/	หาร	15.0/10.0	1.5
	%	เศษ	15%2	1
เปรียบเทียบ	==	เท่ากับ	2 == 3	False
	!=	ไม่เท่ากับ	2 != 3	True
	>	มากกว่า	2 > 3	False
	<	น้อยกว่า	2 < 3	True
	>=	มากกว่าหรือเท่ากับ	2 >= 3	False
	<=	น้อยกว่าหรือเท่ากับ	2 <= 3	True
ตรรกศาสตร์	and	และ	2 < 3 and 3 < 0	False
	or	หรือ	2 < 3 or 3 < 0	True

2.3 ชนิดข้อมูล (Data Types)

ชนิดข้อมูลถือเป็นส่วนสำคัญสำหรับการเขียนโปรแกรม ผู้ใช้ภาษาไพทอนสามารถกำหนดตัวแปรแบบอัตโนมัติ โดยไม่จำเป็นต้องกำหนดชนิดตัวแปรล่วงหน้า ภาษาไพทอนมีการกำหนดชนิดของข้อมูลประเภทต่าง ๆ ดังนี้

- **ประเภทตัวเลขจำนวนเต็ม (Integers)** เป็นข้อมูลประเภทตัวเลขที่ไม่มีจุดทศนิยม สามารถเป็นค่าบวกหรือลบได้

```
>>> x = 10
>>> y = 20
>>> x+y
30
```

- **ประเภทตัวเลขจำนวนจริง (Float)** เป็นตัวเลขที่มีจุดทศนิยม สามารถเป็นค่าบวกหรือลบได้

```
>>> a = 3.5
>>> b = 1.5
>>> a*b
5.25
```

- **ประเภทบูลีน (Boolean)** เป็นชนิดของข้อมูลที่มีได้เพียง 2 ค่าเท่านั้น คือ จริง (True) หรือเท็จ (False)

```
>>> x = 10
>>> y = 20
>>> x == y
False
>>> Z = True
>>> print(Z)
True
```

- **ประเภทสตริง (String)** ข้อมูลชนิดตัวอักษร (String) หรือข้อความ การใช้งานสตริงจะต้องอยู่ภายเครื่องหมายอัญประกาศ (" ") หรืออัญประกาศเดี่ยว (' ')

```
>>> myname = "CK"
>>> print(myname)
CK
```

- **ประเภทลิสต์ (List)** ใช้เก็บข้อมูลได้หลายจำนวนภายในตัวแปรเดียว สามารถเก็บข้อมูลประเภทเดียวกันและต่างประเภทกัน การกำหนดข้อมูลประเภทนี้จะใช้เครื่องหมายวงเล็บเหลี่ยม

```
>>> relay_name = ['V2', 'V3', 'V4', 'V5']
>>> print(relay_name)
['V2', 'V3', 'V4', 'V5']
>>> print(relay_name[0])
V2
>>> relay_name[1] = 'V8'
>>> print(relay_name)
['V2', 'V8', 'V4', 'V5']
```

สำหรับการส่งผ่านค่าลิสต์ ผู้ใช้สามารถใช้เครื่องหมายดอกจัน (*) ตามตัวอย่าง ดังนี้

```
relay_name = ['V2', 'V3', 'V4', 'V5']
def func(*value_list):
    sum = 0
    for val in value_list:
        print('name: '.format(val))
func(*relay_name)
```

- **ประเภททูเพิล (Tuple)** เป็นการเก็บข้อมูลลักษณะคล้ายกับลิสต์ แต่ไม่สามารถแก้ไขได้ การกำหนดข้อมูลประเภทนี้จะใช้เครื่องหมายวงเล็บ

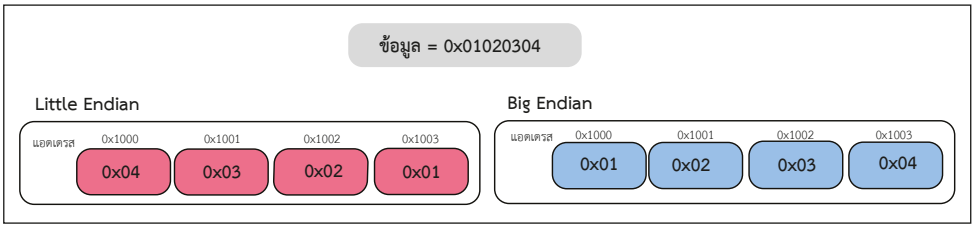
```
>>> relay_name = ('V2', 'V3', 'V4', 'V5')
>>> print(relay_name)
('V2', 'V3', 'V4', 'V5')
>>> print(relay_name[0])
V2
>>>relay_name[1] = 'V8'
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

- **ประเภทดิกชันนารี (Dictionary)** เป็นการเก็บข้อมูลแบบคู่ระหว่างคีย์ (Key) และข้อมูล (Value) โดยที่การกำหนดค่าคีย์ต้องไม่ซ้ำกัน การกำหนดข้อมูลประเภทนี้จะใช้เครื่องหมายวงเล็บปีกกา

```
>>> relay_name = {'relay1':'V2', 'relay2':'V3'}
>>> print(relay_name)
{ 'relay1': 'V2', 'relay2': 'V3' }
>>> print(relay_name['relay1'])
V2
>>> relay_name['relay2'] = 'V8'
>>> print(relay_name)
{'relay1': 'V2', 'relay2': 'V8'}
```

2.4 การแปลงข้อมูลไบต์เป็นข้อมูลแบบไบนารี (Struct)

เนื่องจากการพัฒนาที่แตกต่างกันของการจัดเก็บข้อมูลภายในหน่วยความจำ เช่น การจัดเก็บแบบเรียงจากไบต์หลังสุดของข้อมูลในตำแหน่งหน่วยความจำจากที่แอดเดรสต่ำไปหาแอดเดรสที่สูงกว่า หรือการจัดเก็บข้อมูลจากไบต์แรกของข้อมูลไปตามลำดับของแอดเดรสจากต่ำไปหาแอดเดรสที่สูงกว่า เรียกว่า Little-endian และ Big-endian ตามลำดับ ดังแสดงในรูปที่ 2.1 โดยสถาปัตยกรรมของ Intel x86 และ AMD64 (x86-64) จะใช้รูปแบบการจัดเก็บแบบ Little-endian ในขณะที่สถาปัตยกรรมของ IBM z จะเป็นแบบ Big-endian ทั้งนี้ สถาปัตยกรรมของ ARM, RISC-V และ IBM สามารถใช้ได้ทั้ง 2 แบบ โดยสามารถใช้คำสั่ง sys.byteorder สำหรับการตรวจสอบ



รูปที่ 2.1 ความแตกต่างของ Little Endian และ Big Endian [19]

จากความแตกต่างที่เกิดขึ้น การเก็บข้อมูลหรือการสื่อสารผ่านเน็ตเวิร์กที่อาจมีรูปแบบที่แตกต่างกัน จำเป็นต้องแปลงให้อยู่รูปแบบของออบเจ็กต์ อาศัยการจัดรูปแบบของภาษา C ตามตารางที่ 2.3

ตารางที่ 2.3 Tab:ตัวอย่างการแปลงค่า

รูปแบบ	ประเภท C	ประเภทของไพทอน	ขนาดมาตรฐานการจัดเก็บ
x	Pad Byte	No Value	
c	Char	Bytes of Length 1	1
b	Signed Char	Integer	1
B	Unsigned Char	Integer	1
?	_Bool	Bool	1
h	Short	Integer	2
H	Unsigned Short	Integer	2
i	Int	Integer	4