ARDUINO

ขั้นพื้นฐาน สำหรับผู้เริ่มต้น 1

โดยผู้เชี่ยวชาญทั้งด้านฮาร์ดแวร์และซอร์ฟแวร์

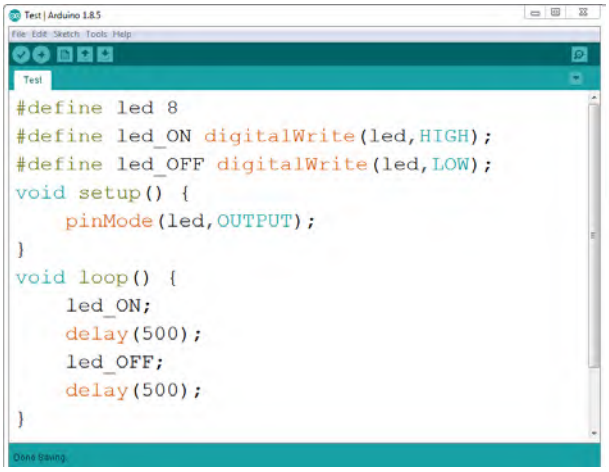
ARDUINO™

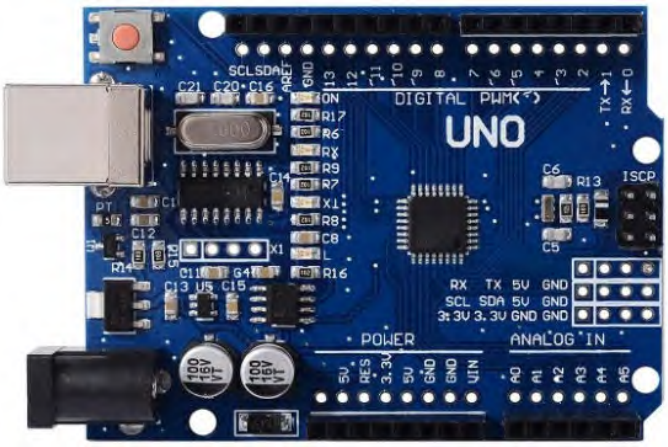
UNO R3

การเขียนโปรแกรม

ด้วยภาษา C++

บน Arduino IDE





เนื้อหาอ่านเข้าใจง่าย

พร้อมตัวอย่างให้ทดลองเพิ่มความเข้าใจ

เหมาะสำหรับผู้เริ่มต้น

MANOP LAB

และผู้ที่ต้องการศึกษาด้วยตนเอง

มานพ ปักษี

ISBN 978-616-485-872-5

ชื่อหนังสือ MicroPython ภาษาระดับสูง ที่ง่ายสำหรับทุกคน

ผู้แต่ง/ผู้เขียน มานพ ปักษี

ออกแบบปก มานพ ปักษี

จัดรูปเล่ม มานพ ปักษี

พิมพ์ครั้งที่ จัดพิมพ์ตามสั่ง หรือตามใจผู้เขียน

จำนวนหน้า 253 หน้า ไม่รวม สารบัญ และ คำนำ

เล่มนี้เป็นงานพิมพ์ด้วยเครื่องอิงค์เจท เข้าเล่มด้วยมือ (งานแฮนด์เมด)

ปกเคลือบสติ๊กเกอร์ใส ควรเก็บให้ห่างจากความร้อน และ ความชื้น

ข้อควรระวัง ไม่ควรให้โดนน้ำ

จัดพิมพ์โดย มานพ ปักษี

จัดจำหน่ายโดย มานพ ปักษี

ร้าน ManopLAB โทรศัพท์ 093-9749793

ที่อยู่ 179 หมู่ 13 ต.สารจิตร อ.ศรีสัชนาลัย จ.สุโขทัย รหัสไปรษณีย์ 64130

nopelectronic@gmail.com

ไฟล์โครงงานต่าง ๆ และเทคนิคพิเศษอีกมากมาย จะอัปเดตลงในเว็บไซต์

manoplab.com เร็ว ๆ นี้

เสมือนคำนำ (เรื่องเล่าก่อนเริ่มต้น)

หลายคนอาจเคยมีความฝันที่จะสร้างอุปกรณ์อิเล็กทรอนิกส์อัจฉริยะด้วยตนเอง ไม่ว่าจะเป็นระบบรดน้ำต้นไม้อัตโนมัติ, ไฟส่องสว่างที่เปิดปิดตามความเคลื่อนไหว, หรือสถานีตรวจวัดสภาพอากาศขนาดเล็กที่สามารถส่งข้อมูลขึ้นสู่อินเทอร์เน็ตได้ หากแนวคิดเหล่านี้สร้างความตื่นเต้น แต่ก็ตามมาด้วยความกังวลว่าการเริ่มต้นอาจเป็นเรื่องยากเกินไป หนังสือเล่มนี้ได้มอบแนวทางและคำตอบไว้ให้ทุกท่านแล้ว

ลองจินตนาการถึงการเขียนโปรแกรมควบคุมอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ ด้วยภาษา Python ซึ่งเป็นภาษาที่ขึ้นชื่อในด้านความเรียบง่าย โครงสร้างชัดเจน อ่านเข้าใจง่าย และได้รับความนิยมอย่างกว้างขวางในหมู่นักพัฒนาทั่วโลก นั่นคือสิ่งที่ MicroPython ได้เข้ามาสานต่อ โดยนำภาษา Python มาปรับแต่งให้สามารถทำงานได้บนไมโครคอนโทรลเลอร์ ซึ่งเป็นหัวใจสำคัญของระบบฝังตัว (Embedded Systems) มากมายในชีวิตประจำวันและเมื่อ MicroPython ถูกนำมาใช้งานร่วมกับ ESP32 ชิปไมโครคอนโทรลเลอร์ยอดนิยม ซึ่งมีทั้งประสิทธิภาพสูง มาพร้อมกับ Wi-Fi และ Bluetooth ที่อยู่ในชิพ ในราคาที่คนทั่วไปเข้าถึงได้ และมีผู้ใช้งานอย่างแพร่หลายทั่วโลก การผสมผสานนี้จึงก่อให้เกิดเป็นคู่หูที่ทรงพลังสำหรับการสร้างสรรค์โครงงานอิเล็กทรอนิกส์และ IoT (Internet of Things) ที่หลากหลาย จากผู้คนทั่วทุกมุมโลก

สำหรับผู้ที่คุ้นเคยกับการเขียนโปรแกรมบน Arduino ด้วยภาษา C/C++ มาก่อน อาจรู้สึกว่าการทดลองพัฒนาโค้ดแต่ละครั้งต้องผ่านกระบวนการคอมไพล์และอัปโหลดโปรแกรมไปยังบอร์ดใหม่ทุกครั้ง ซึ่งทำให้เสียเวลาในการทดสอบมาก โดยเฉพาะเมื่อทำงานกับโค้ดขนาดใหญ่หรือวงจรที่มีความซับซ้อน MicroPython เข้ามาเปลี่ยนประสบการณ์เหล่านี้ ด้วยความยืดหยุ่นของภาษา Python ผู้เรียนสามารถทดลองโค้ดได้ทันทีผ่านระบบ **REPL (Read-Evaluate-Print Loop)** ซึ่งช่วยให้มองเห็นผลลัพธ์ของการสั่งงานกับฮาร์ดแวร์แบบเรียลไทม์โดยไม่ต้องผ่านขั้นตอนการคอมไพล์และอัปโหลดซ้ำในทุกครั้ง โค้ดที่เขียนมีความกระชับ อ่านเข้าใจง่าย ลดภาระความซับซ้อนด้านไวยากรณ์ เหมาะอย่างยิ่งสำหรับผู้ที่ต้องการสำรวจเทคโนโลยีใหม่ ๆ หรือกำลังมองหาเครื่องมือที่ช่วยให้การพัฒนาโครงงานเป็นไปอย่างสนุกสนานและมีประสิทธิภาพมากยิ่งขึ้น

หนังสือเล่มนี้ไม่ได้มุ่งหวังเพียงการสอนวิธีทำให้ไฟ LED กะพริบ (แม้ว่านั่นจะเป็นจุดเริ่มต้นการเรียนรู้ที่ดีก็ตาม) แต่เนื้อหาจะนำพาผู้อ่านเดินทางไปไกลกว่านั้น โดยจะเริ่มปูพื้นฐานตั้งแต่การทำความเข้าใจ MicroPython และ ESP32, การติดตั้งเครื่องมือที่จำเป็น, การเรียนรู้คำสั่ง Python พื้นฐานในบริบทของการควบคุมฮาร์ดแวร์, ไปจนถึงการเชื่อมต่อ ESP32 เข้ากับเซ็นเซอร์ต่างๆ, การแสดงผล, การควบคุมมอเตอร์ และที่สำคัญคือการเชื่อมต่อกับเครือข่าย Wi-Fi เพื่อสร้างโครงงาน IoT ที่สามารถโต้ตอบกับโลกออนไลน์ได้จริง

เป้าหมายของหนังสือเล่มนี้

หนังสือเล่มนี้ถูกเรียบเรียงขึ้นด้วยความตั้งใจที่จะเป็น "คู่มือ" และ "แหล่งเรียนรู้" สำหรับ

- **ผู้เริ่มต้นใหม่ ที่อาจยังไม่มีประสบการณ์ด้านอิเล็กทรอนิกส์หรือการเขียนโปรแกรม**
- **ผู้ใช้งาน Arduino ที่ต้องการขยายขอบเขตความรู้และทดลองใช้ MicroPython**
- **นักเรียน นักศึกษา และผู้สนใจทั่วไป ที่มีความใฝ่ฝันจะเรียนรู้การสร้างโครงงาน IoT ด้วยตนเอง**
- **ครูอาจารย์ ที่กำลังมองหาแนวทางและสื่อการสอนสำหรับวิชาที่เกี่ยวข้อง**

เนื้อหาในหนังสือมุ่งเน้นการนำเสนอที่ เข้าใจง่าย เป็นขั้นตอน มีตัวอย่างที่ชัดเจน และ รวบรวมความรู้ที่จำเป็นทั้งภาคทฤษฎีและปฏิบัติ เพื่อให้ผู้เรียนสามารถศึกษาและลงมือทำตามได้จริง จนสามารถสร้างสรรค์โครงงานของตนเองได้อย่างมั่นใจ ทั้งหมดนี้ได้ถูกถ่ายทอดเป็นภาษาไทยที่กระชับและเข้าใจง่าย โดยคนไทย เพื่อคนไทย

การเดินทางสู่การสร้างสรรค์โครงงาน IoT ด้วย MicroPython และ ESP32:

ภายในหนังสือเล่มนี้ ผู้อ่านจะได้เรียนรู้ตามลำดับขั้นดังนี้ :

1. **การตั้งค่าสภาพแวดล้อม และเครื่องมือที่จำเป็นสำหรับการพัฒนา**
2. **พื้นฐานภาษา Python ที่สำคัญและประยุกต์ใช้บ่อยใน MicroPython**
3. **การควบคุมขา GPIO ของ ESP32 เพื่อสั่งงาน LED, อ่านค่าจากสวิตช์ และอื่นๆ**
4. **การทำงานกับ เซ็นเซอร์ ที่นิยมใช้ เช่น เซ็นเซอร์วัดอุณหภูมิ, ความชื้น, ระยะทาง**
5. **การ เชื่อมต่อ Wi-Fi และพื้นฐานการสื่อสารผ่านเครือข่าย**
6. **โครงงานตัวอย่าง ที่นำความรู้ทั้งหมดมาประยุกต์ใช้ เช่น สถานีตรวจอากาศส่วนบุคคล, ระบบควบคุมไฟอัจฉริยะ...**

ผู้เขียนเชื่อว่าเมื่อผู้อ่านศึกษาหนังสือเล่มนี้จบ จะไม่เพียงแต่เกิดความเข้าใจในหลักการทำงานของ MicroPython และ ESP32 แต่ยังจะได้รับแรงบันดาลใจและความพร้อมที่จะสร้างสรรค์สิ่งประดิษฐ์และนวัตกรรมใหม่ ๆ ออกมาอีกมากมาย ถึงเวลาแล้วที่จะปลดล็อกศักยภาพของไมโครคอนโทรลเลอร์ด้วยความมหัศจรรย์และความเรียบง่ายของ Python มาเริ่มต้นการเดินทางที่น่าตื่นเต้นนี้ไปด้วยกัน

มานพ ปักษี

สารบัญ

บทที่ 1: MicroPython และ ESP32 คืออะไร? ทำไมต้องใช้คู่นี้?

1.1 ยินดีต้อนรับสู่โลกแห่งไมโครคอนโทรลเลอร์ยุคใหม่

ในยุคปัจจุบัน เทคโนโลยีได้เข้ามาเป็นส่วนหนึ่งของชีวิตประจำวันอย่างแยกไม่ออก หนึ่งในเทคโนโลยีที่มีบทบาทสำคัญที่อยู่เบื้องหลังของนวัตกรรมและสิ่งอำนวยความสะดวกมากมายก็คือ **ไมโครคอนโทรลเลอร์ (Microcontroller)** หรือที่เราอาจคุ้นเคยในชื่อ **"สมองกลจิ๋ว","สมองกลฝังตัว"** อุปกรณ์ขนาดเล็กเหล่านี้เปรียบเสมือนหัวใจสำคัญที่ขับเคลื่อนอุปกรณ์อิเล็กทรอนิกส์หลากหลายชนิด ตั้งแต่ของเล่นเด็ก, เครื่องใช้ไฟฟ้าในบ้าน, ระบบควบคุมในรถยนต์, ไปจนถึงอุปกรณ์ทางการแพทย์ที่ซับซ้อน และที่กำลังได้รับความสนใจอย่างกว้างขวางคือการนำไปใช้ในโครงงาน **Internet of Things (IoT)** หรือ "อินเทอร์เน็ตของสรรพสิ่ง" ซึ่งเป็นการเชื่อมโยงอุปกรณ์ต่างๆ เข้ากับเครือข่ายอินเทอร์เน็ต ทำให้สามารถควบคุม สั่งการ หรือเก็บข้อมูลได้จากระยะไกล

การสร้างสรรค์สิ่งประดิษฐ์หรือระบบอัจฉริยะด้วยไมโครคอนโทรลเลอร์จึงไม่ใช่เรื่องไกลตัวอีกต่อไป เปิดโอกาสให้ทุกคน ไม่ว่าจะเป็นนักเรียน นักศึกษา นักประดิษฐ์ หรือผู้ที่สนใจทั่วไป สามารถเรียนรู้และพัฒนาโครงงานของตนเองขึ้นมาได้ อย่างไรก็ตาม ในอดีต การเริ่มต้นใช้งานไมโครคอนโทรลเลอร์อาจมีอุปสรรคอยู่บ้าง ไม่ว่าจะเป็นความซับซ้อนของภาษาโปรแกรมที่ใช้ควบคุม เครื่องมือพัฒนาที่อาจต้องตั้งค่าหลายขั้นตอน หรือข้อจำกัดทางด้านฮาร์ดแวร์ที่อาจไม่ยืดหยุ่นพอสำหรับโครงงานที่ต้องการความสามารถ หลากหลาย และรอบด้าน

แต่ในปัจจุบัน ด้วยความก้าวหน้าของเทคโนโลยี ทั้งด้านซอฟต์แวร์และฮาร์ดแวร์ ได้ทำให้การเข้าถึงและการเรียนรู้ไมโครคอนโทรลเลอร์เป็นเรื่องง่ายและสนุกสนานยิ่งขึ้นอย่างที่ไม่เคยเป็นมาก่อน หนังสือเล่มนี้จะนำพาผู้อ่านทุกท่านเข้าสู่โลกของไมโครคอนโทรลเลอร์ยุคใหม่ ผ่านการทำความรู้จักกับเครื่องมืออันทรงพลังสองสิ่งที่กำลังปฏิวัติวงการนี้ นั่นคือ **MicroPython** และชิป **ESP32** ซึ่งจะช่วยลดข้อจำกัดเดิม ๆ และเปิดประตูสู่ความเป็นไปได้ใหม่ ๆ ในการสร้างสรรค์โครงงานอิเล็กทรอนิกส์และ IoT ได้อย่างง่ายดายและมีประสิทธิภาพ

1.2 MicroPython: Python ในร่างจิ๋วแต่ทรงพลัง

MicroPython คืออะไร? MicroPython คือการนำภาษาโปรแกรม Python ที่ได้รับความนิยมอย่างสูงมาปรับปรุงและย่อส่วนเพื่อให้สามารถทำงานได้อย่างมีประสิทธิภาพบนไมโครคอนโทรลเลอร์ ซึ่งเป็นอุปกรณ์ที่มีทรัพยากรจำกัด เช่น หน่วยความจำ (RAM) และพื้นที่จัดเก็บข้อมูล (Flash memory) น้อยกว่าคอมพิวเตอร์ทั่วไป โดยยังคงรักษาจุดเด่นหลักของภาษา Python ไว้ นั่นคือไวยากรณ์ที่อ่านง่าย เข้าใจง่าย คล้ายคลึงกับภาษาอังกฤษ ทำให้ผู้เริ่มต้นสามารถเรียนรู้ได้อย่างรวดเร็ว ผู้สร้าง MicroPython คือ Damien George วิศวกรชาวออสเตรเลีย ผู้เริ่มต้นโครงการนี้ผ่านการระดมทุนบน Kickstarter ในปี 2013 โดยมีเป้าหมายเพื่อสร้างซอฟต์แวร์ที่ช่วยให้การ

เขียนโปรแกรมควบคุมฮาร์ดแวร์เป็นเรื่องง่ายขึ้น นับตั้งแต่นั้นมา MicroPython ก็ได้รับการพัฒนาอย่างต่อเนื่องและมีชุมชนผู้ใช้งานขยายใหญ่ขึ้นทั่วโลก รองรับไมโครคอนโทรลเลอร์หลากหลายรุ่น

ข้อดีของ MicroPython:

- **เรียนรู้ง่าย เขียนโค้ดสั้นกระชับ:** ด้วยไวยากรณ์ที่เรียบง่ายของ Python ผู้ที่ไม่มีพื้นฐานการเขียนโปรแกรมมาก่อนก็สามารถเริ่มต้นได้ไม่ยาก และสำหรับผู้ที่มีพื้นฐาน Python อยู่แล้ว ก็สามารถนำความรู้เดิมมาปรับใช้ได้ทันที นอกจากนี้ โค้ดที่เขียนด้วย MicroPython มักจะสั้นกว่าโค้ดที่เขียนด้วยภาษาโปรแกรมระดับต่ำอย่าง C/C++ สำหรับงานเดียวกัน ทำให้เข้าใจภาพรวมของโปรแกรมได้ง่ายขึ้น
- **พัฒนาและทดสอบได้รวดเร็ว (Rapid Prototyping):** MicroPython มาพร้อมกับเครื่องมือที่เรียกว่า REPL (Read-Evaluate-Print Loop) ซึ่งเป็น Command Line Interface ที่ช่วยให้ผู้ใช้สามารถพิมพ์คำสั่ง Python และเห็นผลลัพธ์การทำงานได้ทันทีบนตัวไมโครคอนโทรลเลอร์ ทำให้การทดลองโค้ดส่วนเล็กๆ หรือการตรวจสอบการทำงานของฮาร์ดแวร์เป็นไปอย่างรวดเร็ว ไม่จำเป็นต้องคอมไพล์และอัปโหลดโปรแกรมใหม่ทั้งหมดทุกครั้งที่แก้ไขโค้ดเพียงเล็กน้อย
- **มีไลบรารีพื้นฐานพร้อมใช้งาน:** MicroPython มีโมดูล (ไลบรารี) มาตรฐานที่จำเป็นสำหรับการควบคุมฮาร์ดแวร์ เช่น การจัดการขา GPIO (General Purpose Input/Output) การสื่อสารแบบ I2C, SPI, UART รวมถึงการเชื่อมต่อเครือข่าย ทำให้การเริ่มต้นพัฒนาโครงงานเป็นไปได้สะดวก
- **ขยายองค์ความรู้ Python:** การเรียนรู้ MicroPython เท่ากับเป็นการเรียนรู้ภาษา Python ไปในตัว ซึ่งเป็นภาษาที่มีความต้องการสูงในตลาดงานปัจจุบัน สามารถนำไปต่อยอดในการพัฒนาเว็บแอปพลิเคชัน, งานด้านข้อมูล (Data Science), ปัญญาประดิษฐ์ (AI) และอื่นๆ ได้อีกมากมาย

1.3 ESP32: ชิปมหัศจรรย์สำหรับโครงงาน IoT

ESP32 คืออะไร? ESP32 คือชื่อของซีรีส์ชิปไมโครคอนโทรลเลอร์ราคาประหยัดที่พัฒนาโดยบริษัท Espressif Systems จากประเทศจีน ซึ่งได้รับการออกแบบมาให้มีความสามารถรอบด้าน โดยเฉพาะอย่างยิ่งสำหรับงานที่เกี่ยวข้องกับ Internet of Things (IoT) และอุปกรณ์พกพา ESP32 เป็นรุ่นที่พัฒนาต่อยอดมาจาก ESP8266 ซึ่งเคยสร้างความนิยมอย่างสูงในหมู่นักพัฒนาและนักประดิษฐ์ ด้วยการเพิ่มคุณสมบัติและความสามารถที่ทรงพลังยิ่งขึ้น

โดยทั่วไป เมื่อพูดถึง ESP32 มักจะหมายถึงตัวชิป SoC (System on a Chip) แต่ในท้องตลาด บอร์ดพัฒนา (Development Board) ที่ใช้ชิป ESP32 ก็มักถูกเรียกว่า "บอร์ด ESP32" เช่นกัน บอร์ดเหล่านี้จะรวมเอา

ชิป ESP32 พร้อมวงจรสนับสนุนที่จำเป็น เช่น วงจรแปลงแรงดันไฟฟ้า, พอร์ต USB สำหรับการโปรแกรมและจ่ายไฟ, และขาต่อต่างๆ ทำให้ง่ายต่อการนำไปใช้งาน

จุดเด่นของ ESP32:

- **การเชื่อมต่อไร้สายครบครัน:** ESP32 มาพร้อมกับ Wi-Fi (มาตรฐาน 802.11 b/g/n) และ Bluetooth (ทั้ง Classic และ Bluetooth Low Energy - BLE) ในตัว ทำให้สามารถเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตหรืออุปกรณ์อื่นๆ แบบไร้สายได้อย่างง่ายดาย ซึ่งเป็นหัวใจสำคัญของโครงงาน IoT
- **ประสิทธิภาพสูง:** ส่วนใหญ่มาพร้อมกับโปรเซสเซอร์ Xtensa LX6 แบบ Dual-core (สองแกนประมวลผล) ที่ความเร็วสัญญาณนาฬิกาสูงถึง 240 MHz (ขึ้นอยู่กับรุ่น) ทำให้สามารถประมวลผลงานที่ซับซ้อนได้ดีกว่าไมโครคอนโทรลเลอร์แกนเดียวหลายๆ รุ่น
- **หน่วยความจำเพียงพอ:** มีหน่วยความจำ SRAM และ Flash memory ภายในตัวในปริมาณที่ค่อนข้างมากสำหรับไมโครคอนโทรลเลอร์ (เช่น SRAM หลายร้อย KB และ Flash memory หลาย MB) เพียงพอสำหรับการรัน MicroPython และเก็บโปรแกรมขนาดกลางได้
- **ขาเชื่อมต่อ (GPIO) อเนกประสงค์:** มีขา General Purpose Input/Output (GPIO) จำนวนมาก ซึ่งสามารถกำหนดค่าให้ทำงานได้หลากหลายหน้าที่ เช่น เป็น Digital Input/Output, Analog Input (ADC), Digital to Analog Output (DAC), Pulse Width Modulation (PWM), รวมถึงรองรับโปรโตคอลการสื่อสารมาตรฐาน เช่น I2C, SPI, UART
- **เซ็นเซอร์ในตัว:** บางรุ่นของ ESP32 มีเซ็นเซอร์พื้นฐานในตัว เช่น เซ็นเซอร์สัมผัส (Touch Sensor) และเซ็นเซอร์อุณหภูมิ (Temperature Sensor)
- **ความปลอดภัย:** รองรับคุณสมบัติด้านความปลอดภัยของฮาร์ดแวร์ เช่น Secure Boot และ Flash Encryption
- **ราคาคุ้มค่า:** เมื่อเทียบกับคุณสมบัติและความสามารถที่ได้รับ บอร์ดพัฒนา ESP32 ถือว่ามีราคาที่เข้าถึงได้ง่าย ทำให้นักเรียน นักศึกษา และนักประดิษฐ์ทั่วไปสามารถเริ่มต้นได้โดยที่ไม่ต้องลงทุนสูงมากนัก

1.4 MicroPython + ESP32: คู่หูที่ลงตัวที่สุดสำหรับการเริ่มต้น

เมื่อความเรียบง่ายและยืดหยุ่นของ MicroPython ผสานเข้ากับพลังและความสามารถรอบด้านของชิป ESP32 จึงเกิดเป็นแพลตฟอร์มที่ลงตัวอย่างยิ่งสำหรับการเรียนรู้และพัฒนาโครงงานอิเล็กทรอนิกส์และ IoT โดยเฉพาะสำหรับผู้เริ่มต้นและผู้ที่ต้องการสร้างต้นแบบ (Prototype) อย่างรวดเร็ว

เหตุผลที่ MicroPython และ ESP32 เป็นคู่หูที่ยอดเยี่ยม:

- **ลดกำแพงการเริ่มต้น:** ผู้เริ่มต้นไม่จำเป็นต้องกังวลกับความซับซ้อนของภาษา C/C++ หรือการตั้งค่าเครื่องมือที่ยุ่งยาก เพียงแค่ติดตั้งเฟิร์มแวร์ MicroPython ลงบนบอร์ด ESP32 และใช้โปรแกรม Text Editor หรือ IDE ง่ายๆ อย่าง Thonny ก็สามารถเริ่มเขียนโค้ดควบคุมฮาร์ดแวร์ได้ทันที
- **ใช้ประโยชน์จากฮาร์ดแวร์ ESP32 ได้เต็มที่ (อย่างง่ายดาย):** MicroPython สำหรับ ESP32 มีโมดูลที่ช่วยให้การเข้าถึงคุณสมบัติหลักของ ESP32 เช่น Wi-Fi, Bluetooth, GPIO, ADC, PWM เป็นเรื่องง่าย เพียงเขียนโค้ด Python ไม่กี่บรรทัดก็สามารถสั่งงานฟังก์ชันเหล่านี้ได้
- **เหมาะสำหรับโครงงาน IoT:** การเชื่อมต่อ Wi-Fi ของ ESP32 เมื่อรวมกับความสามารถในการจัดการข้อความและข้อมูลของ Python ทำให้การพัฒนาอุปกรณ์ IoT ที่ส่งข้อมูลไปยังเซิร์ฟเวอร์ หรือรับคำสั่งผ่านเครือข่าย ทำได้ไม่ซับซ้อน
- **ชุมชนผู้ใช้งานขนาดใหญ่:** ทั้ง MicroPython และ ESP32 ต่างก็มีชุมชนผู้ใช้งานที่กระตือรือร้นและพร้อมให้ความช่วยเหลือ เมื่อพบปัญหาหรือมีข้อสงสัย สามารถค้นหาข้อมูล แนวทางการแก้ไข หรือตัวอย่างโค้ดได้ไม่ยาก ด้วยการผสมผสานนี้ ผู้อ่านสามารถจินตนาการถึงการสร้างสรรค์โครงงานที่หลากหลายได้อย่างง่ายดาย เช่น:
  - ระบบควบคุมไฟส่องสว่างในบ้านผ่านสมาร์ตโฟน
  - สถานีตรวจวัดสภาพอากาศขนาดเล็กที่ส่งข้อมูลอุณหภูมิและความชื้นขึ้นเว็บไซต์
  - หุ่นยนต์ขนาดเล็กที่ควบคุมการเคลื่อนที่ด้วยคำสั่ง Python
  - ระบบแจ้งเตือนเมื่อมีผู้บุกรุกผ่านเซ็นเซอร์ตรวจจับความเคลื่อนไหว
  - อุปกรณ์สวมใส่ที่เก็บข้อมูลสุขภาพเบื้องต้นและส่งไปยังแอปพลิเคชัน

1.5 เปรียบเทียบสำหรับผู้ใช้ Arduino: ก้าวใหม่ที่น่าลอง

สำหรับผู้ที่คุ้นเคยกับแพลตฟอร์ม Arduino และภาษา C/C++ การมาถึงของ MicroPython บน ESP32 อาจเป็นทั้งโอกาสและความท้าทายใหม่ที่น่าสนใจ Arduino ได้สร้างคุณูปการอย่างใหญ่หลวงในการทำให้การเรียนรู้ไมโครคอนโทรลเลอร์เป็นเรื่องง่ายสำหรับคนจำนวนมาก และ MicroPython ก็มีเป้าหมายเดียวกัน แต่มาพร้อมกับแนวทางที่แตกต่างออกไป

ความแตกต่างหลักระหว่าง Arduino (C/C++) และ MicroPython บน ESP32

คุณสมบัติ	Arduino (C/C++)	MicroPython บน ESP32
ภาษาโปรแกรม	C/C++ (Static Typing)	Python (Dynamic Typing)
การทำงาน	คอมไพล์เป็น Machine Code แล้วอัปโหลด	Interpreter ทำงานบนเฟิร์มแวร์ MicroPython
การพัฒนา	เขียนโค้ด -> คอมไพล์ -> อัปโหลด -> ทดสอบ	เขียนโค้ด -> (อัปโหลดไฟล์ .py) -> ทดสอบผ่าน REPL/รันไฟล์
ความเร็วในการทดลอง	ช้ากว่า เนื่องจากต้องคอมไพล์ใหม่ทุกครั้ง	เร็วกว่ามาก สามารถทดลองโค้ดทีละบรรทัดได้
ความซับซ้อนของภาษา	สูงกว่า โดยเฉพาะเรื่อง Pointer, Memory Management	ต่ำกว่า ไวยากรณ์อ่านง่าย จัดการหน่วยความจำอัตโนมัติ
การจัดการทรัพยากร	ผู้เขียนโค้ดต้องจัดการอย่างระมัดระวัง	มี Garbage Collector ช่วยจัดการหน่วยความจำ

ข้อดีที่ผู้ใช้ Arduino อาจได้รับจาก MicroPython:

- **ลดเวลาในการพัฒนา:** วงจรการแก้ไขโค้ดและทดสอบที่สั้นลงอย่างเห็นได้ชัด
- **ภาษาที่ยืดหยุ่นกว่า:** การจัดการข้อความ (String), รายการข้อมูล (List), และโครงสร้างข้อมูลแบบ Dictionary ใน Python ทำได้ง่ายและมีประสิทธิภาพกว่าใน C/C++ พื้นฐาน
- **การเขียนโปรแกรมเชิงวัตถุ (OOP):** แม้ C++ จะรองรับ OOP แต่ Python ก็มีโครงสร้าง OOP ที่เข้าใจง่ายและเป็นธรรมชาติ
- **ลดความกังวลเรื่อง Memory Management พื้นฐาน:** ช่วยให้ผู้เริ่มต้นโฟกัสไปที่ตรรกะของโปรแกรมได้มากขึ้น

อย่างไรก็ตาม ไม่ได้หมายความว่า MicroPython ดีกว่า Arduino C/C++ ในทุกกรณี การเขียนโปรแกรมด้วย C/C++ โดยตรงยังคงให้ประสิทธิภาพการทำงานที่เร็วกว่าและใช้ทรัพยากรน้อยกว่า เหมาะสำหรับงานที่ต้องการความเร็วสูงมากๆ หรือมีข้อจำกัดด้านหน่วยความจำที่เข้มงวดสุดๆ MicroPython จึงเป็นอีก **ทางเลือกหนึ่ง** ที่น่าสนใจสำหรับโครงงานหลายประเภท โดยเฉพาะโครงงาน IoT ที่เน้นการเชื่อมต่อเครือข่ายและการจัดการข้อมูล ซึ่งความเร็วในการพัฒนาอาจมีความสำคัญมากกว่าประสิทธิภาพระดับไมโครวินาที

1.6 MicroPython กับ ESP32 ในมุมมองของครูอาจารย์

สำหรับครูอาจารย์ผู้สอนวิชาวิทยาการคำนวณ, การออกแบบและเทคโนโลยี, หรือวิชาที่เกี่ยวข้องกับ STEM และ IoT นั้น MicroPython และ ESP32 ถือเป็นเครื่องมือการสอนที่มีศักยภาพสูงมาก

- **ลดอุปสรรคการเรียนรู้โค้ดดิ้ง:** ไวยากรณ์ที่ง่ายของ Python ช่วยให้นักเรียนที่ไม่เคยมีประสบการณ์มาก่อน สามารถเข้าใจแนวคิดการเขียนโปรแกรมได้เร็วขึ้น ลดความกลัวและความรู้สึกว่าโค้ดดิ้งเป็นเรื่องยาก
- **ส่งเสริมการเรียนรู้แบบโครงงานเป็นฐาน (Project-Based Learning):** นักเรียนสามารถนำความรู้ไปสร้างสรรค์โครงงานที่จับต้องได้จริงอย่างรวดเร็ว ตั้งแต่การควบคุม LED ง่ายๆ ไปจนถึงการสร้างอุปกรณ์ IoT ที่เชื่อมต่ออินเทอร์เน็ต ซึ่งช่วยเพิ่มแรงจูงใจและความสนุกสนานในการเรียนรู้
- **เชื่อมโยงกับชีวิตจริง:** โครงงาน IoT ที่สร้างจาก ESP32 และ MicroPython สามารถออกแบบให้แก้ปัญหาในชีวิตประจำวันหรือในโรงเรียนได้ เช่น ระบบตรวจวัดคุณภาพอากาศ, ระบบรดน้ำต้นไม้อัตโนมัติสำหรับแปลงผักของโรงเรียน ทำให้นักเรียนเห็นคุณค่าของสิ่งที่เรียน
- **พัฒนาทักษะการแก้ปัญหาและคิดเชิงคำนวณ:** การออกแบบและแก้ไขโปรแกรมสำหรับควบคุมฮาร์ดแวร์ เป็นการฝึกฝนทักษะการคิดวิเคราะห์ การวางแผน และการแก้ปัญหาอย่างเป็นระบบ
- **เครื่องมือที่เข้าถึงง่ายและราคาไม่แพง:** บอร์ด ESP32 มีราคาที่ไม่สูงนัก ทำให้โรงเรียนหรือสถาบันการศึกษาสามารถจัดหาให้นักเรียนใช้งานได้อย่างทั่วถึง

การใช้ MicroPython และ ESP32 ในห้องเรียน จึงไม่เพียงแต่สอนทักษะการเขียนโปรแกรม แต่ยังเป็นการเปิดโลกทัศน์ให้นักเรียนเห็นถึงพลังของเทคโนโลยีในการสร้างสรรค์นวัตกรรมและแก้ปัญหาต่างๆ รอบตัว

1.7 บทสรุป: เตรียมพร้อมสำหรับการเดินทาง

จากที่กล่าวมาทั้งหมด จะเห็นได้ว่า MicroPython และ ESP32 คือการผสมผสานที่ลงตัวระหว่างความง่ายในการใช้งานของซอฟต์แวร์และความสามารถอันทรงพลังของฮาร์ดแวร์ นับเป็นเครื่องมือที่เปิดโอกาสให้ทุกคนสามารถก้าวเข้าสู่โลกแห่งการสร้างสรรค์โครงงานอิเล็กทรอนิกส์และ Internet of Things ได้อย่างมั่นใจ ไม่ว่าจะมีพื้นฐานมาก่อนหรือไม่ก็ตาม

เสน่ห์ของ MicroPython อยู่ที่ความสามารถในการปลดปล่อยจินตนาการของผู้พัฒนาให้เป็นอิสระจากความซับซ้อนของโค้ด ในขณะที่ ESP32 ก็มอบขุมพลังและช่องทางการเชื่อมต่อที่หลากหลายรอให้ถูกนำไปใช้งาน หนังสือเล่มนี้จะเป็นเสมือนแผนที่นำทาง พาผู้อ่านไปสำรวจดินแดนอันน่าตื่นเต้นนี้ ตั้งแต่การทำความรู้จัก การติดตั้งเครื่องมือ จนถึงการลงมือสร้างโครงงานจริง

ในบทต่อไป เราจะเริ่มต้นการเดินทางภาคปฏิบัติ ด้วยการเตรียมความพร้อมเครื่องมือและอุปกรณ์ที่จำเป็น รวมถึงการติดตั้ง MicroPython ลงบนบอร์ด ESP32 เพื่อให้พร้อมสำหรับการเขียนโปรแกรมแรก ขอเชิญผู้อ่านทุกท่านเตรียมตัวให้พร้อม แล้วมาเริ่มต้นสร้างสรรค์สิ่งประดิษฐ์ด้วยกัน!

บทที่ 2: ติดตั้งและเริ่มต้นใช้งาน MicroPython บน ESP32

หลังจากที่เราได้ทำความรู้จักกับ MicroPython และ ESP32 รวมถึงเหตุผลที่ทำให้ทั้งสองเป็นคู่หูที่ยอดเยี่ยมสำหรับการเริ่มต้นในบทที่ 1 แล้ว ในบทนี้ จะเข้าสู่ภาคปฏิบัติกันอย่างเต็มตัว โดยจะแนะนำขั้นตอนที่จำเป็นทั้งหมด ตั้งแต่การเลือกบอร์ด ESP32, การเตรียมอุปกรณ์, การติดตั้งไดรเวอร์, การติดตั้งเฟิร์มแวร์ MicroPython ลงบนบอร์ด ESP32, ไปจนถึงการตั้งค่าโปรแกรม Thonny IDE และการทดลองเขียนโปรแกรมแรกเพื่อควบคุมฮาร์ดแวร์เบื้องต้น การทำตามขั้นตอนในบทนี้อย่างละเอียดจะช่วยให้ผู้อ่านมีพื้นฐานที่มั่นคงและพร้อมสำหรับการเรียนรู้การเขียนโปรแกรม MicroPython เพื่อสร้างสรรค์โครงงานต่างๆ ในบทต่อๆ ไป

2.1 การเลือกบอร์ด ESP32 สำหรับมือใหม่

ชิป ESP32 ถูกนำไปใช้งานในบอร์ดพัฒนา (Development Board) หลากหลายรูปแบบและหลายผู้ผลิต สำหรับผู้เริ่มต้น การเลือกบอร์ดที่เหมาะสมจะช่วยให้การเรียนรู้ง่ายขึ้น บอร์ดที่แนะนำควรมีลักษณะดังนี้:

- **มีพอร์ต Micro-USB หรือ USB-C:** สำหรับการจ่ายไฟและอัปโหลดโปรแกรมได้สะดวก
- **มีวงจร USB to UART ในตัว:** บอร์ดส่วนใหญ่มักใช้ชิป เช่น CP2102, CP2104 หรือ CH340/CH9102F เพื่อทำหน้าที่นี้ ทำให้สามารถเชื่อมต่อกับคอมพิวเตอร์ผ่านสาย USB ได้โดยตรง
- **มีปุ่ม BOOT (หรือ FLASH) และ EN (หรือ RESET/RST):** ปุ่มเหล่านี้จำเป็นสำหรับการเข้าโหมดดาวน์โหลดเฟิร์มแวร์
- **มี LED ออนบอร์ด (Onboard LED):** อย่างน้อยหนึ่งดวง เพื่อใช้ทดสอบการเขียนโปรแกรมเบื้องต้นได้ง่าย
- **ขา GPIO ถูกต่อออกมาให้ใช้งานง่าย:** ควรมี Pinout Diagram (แผนผังขา) ที่ชัดเจน และขาต่างๆ ควรถูกพิมพ์ชื่อกำกับไว้บนบอร์ดหรือหาข้อมูลได้ง่าย

บอร์ด ESP32 ที่เป็นที่นิยมและเหมาะสำหรับผู้เริ่มต้น ได้แก่:

- **ESP32-DevKitC:** เป็นบอร์ดมาตรฐานจาก Espressif เอง มีหลายเวอร์ชันย่อย (เช่น ใช้ชิป ESP32-WROOM-32, ESP32-WROVER) หาซื้อง่าย มีข้อมูลสนับสนุนเยอะ
- **NodeMCU-32S:** เป็นอีกหนึ่งบอร์ดที่ได้รับความนิยมสูง มีลักษณะคล้ายกับ NodeMCU ที่ใช้ ESP8266 ทำให้ผู้ที่เคยใช้ ESP8266 มาก่อนคุ้นเคยได้ง่าย

- **บอร์ดจากผู้ผลิตอื่นๆ:** เช่น Wemos Lolin D32, Adafruit HUZZAH32, SparkFun ESP32 Thing ซึ่งมักจะมีคุณสมบัติเพิ่มเติมหรือการออกแบบที่เป็นเอกลักษณ์

คำแนะนำ: ก่อนซื้อ ควรตรวจสอบข้อมูลจำเพาะของบอร์ด รีวิวจากผู้ใช้งาน และแหล่งข้อมูลสนับสนุน เพื่อให้ได้บอร์ดที่ตรงกับความต้องการและง่ายต่อการเริ่มต้น

2.2 เตรียมความพร้อมก่อนเริ่มต้น

ก่อนจะลงมือติดตั้งและเขียนโปรแกรม มีสิ่งจำเป็นที่ต้องเตรียมให้พร้อมดังนี้:

1. **บอร์ด ESP32:** ตามที่ได้เลือกไว้ในหัวข้อ 2.1
2. **สาย Micro-USB หรือ USB-C Data Cable:**
    - *ข้อควรระวัง:* ตรวจสอบให้แน่ใจว่าเป็นสายข้อมูล (Data Cable) ไม่ใช่สายชาร์จไฟอย่างเดียว (Charge-only Cable) สายข้อมูลจะสามารถทั้งจ่ายไฟและรับส่งข้อมูลได้ สังเกตได้จากเมื่อเสียบกับคอมพิวเตอร์แล้วควรมีอุปกรณ์ใหม่ถูกตรวจพบ (แม้จะยังไม่ได้ติดตั้งไดรเวอร์ก็ตาม)
3. **คอมพิวเตอร์:** ระบบปฏิบัติการ Windows, macOS, หรือ Linux ก็ได้
4. **การเชื่อมต่ออินเทอร์เน็ต:** สำหรับดาวน์โหลดซอฟต์แวร์และไดรเวอร์ที่จำเป็น

2.3 การติดตั้งไดรเวอร์ USB to UART (ถ้าจำเป็น)

บอร์ด ESP32 ส่วนใหญ่จะใช้ชิปแปลงสัญญาณ USB เป็น UART (Universal Asynchronous Receiver/Transmitter) เพื่อให้คอมพิวเตอร์สามารถสื่อสารกับชิป ESP32 ผ่านทางพอร์ต USB ได้ ชิปที่นิยมใช้คือ CP210x (จาก Silicon Labs) หรือ CH340/CH9102F (จาก WCH)

- **สำหรับ Windows:**
    - โดยทั่วไป Windows 10/11 อาจรู้จักชิปเหล่านี้และติดตั้งไดรเวอร์ให้โดยอัตโนมัติเมื่อเสียบบอร์ด ESP32 ครั้งแรก
    - หากไม่รู้จัก (เช่น ใน Device Manager แสดงเป็น Unknown Device หรือมีเครื่องหมายตกใจสีเหลือง) จำเป็นต้องดาวน์โหลดและติดตั้งไดรเวอร์ด้วยตนเอง:
        - **CP210x:** ดาวน์โหลดจากเว็บไซต์ Silicon Labs (ค้นหา "CP210x USB to UART Bridge VCP Drivers")
        - **CH340/CH9102F:** ดาวน์โหลดจากเว็บไซต์ผู้ผลิต WCH (ค้นหา "CH341SER.EXE" หรือ "CH340 driver")
    - หลังติดตั้งไดรเวอร์และเสียบบอร์ด ESP32 ควรจะเห็นพอร์ต COM ใหม่ปรากฏขึ้นใน Device Manager (อยู่ใต้ Ports (COM & LPT)) เช่น Silicon Labs CP210x USB to UART Bridge

(COM3) หรือ USB-SERIAL CH340 (COM4) หรือ COM เลขอื่น ๆ แล้วแต่เครื่องนั้น ๆ ให้จดจำหมายเลขพอร์ต COM นี้ไว้

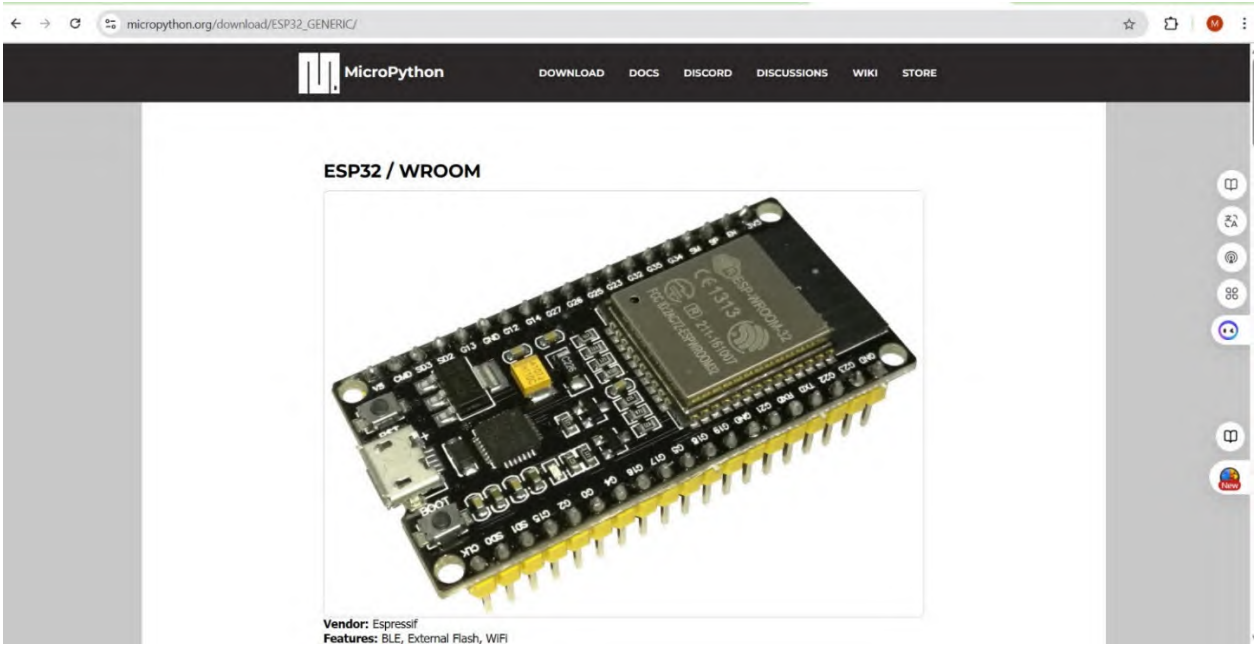
- **สำหรับ macOS:**
    - macOS ส่วนใหญ่มักจะรู้จักชิป CP210x โดยไม่ต้องติดตั้งไดรเวอร์เพิ่มเติม
    - สำหรับชิป CH340/CH9102F อาจจำเป็นต้องติดตั้งไดรเวอร์จากผู้ผลิต WCH (ค้นหา "CH34x_Install_V1.x.pkg" หรือไดรเวอร์สำหรับ Mac รุ่นล่าสุด)
    - หลังจากเสียบบอร์ด ESP32 พอร์ตจะปรากฏในรูปแบบ /dev/cu.usbserial-xxxx หรือ /dev/cu.SLAB_USBtoUART (สำหรับ CP210x) หรือ /dev/cu.wchusbserialxxxx (สำหรับ CH340) สามารถตรวจสอบได้โดยเปิด Terminal แล้วพิมพ์คำสั่ง ls /dev/cu.*
- **สำหรับ Linux:**
    - Linux Kernel ส่วนใหญ่รองรับชิปทั้ง CP210x และ CH340/CH9102F โดยไม่ต้องติดตั้งไดรเวอร์เพิ่มเติม
    - เมื่อเสียบบอร์ด ESP32 พอร์ตมักจะปรากฏในรูปแบบ /dev/ttyUSB0, /dev/ttyUSB1 หรือคล้ายกัน สามารถตรวจสอบได้โดยเปิด Terminal แล้วพิมพ์คำสั่ง ls /dev/ttyUSB* หรือ dmesg | grep tty หลังจากเสียบบอร์ด
    - *ข้อควรทราบสำหรับ Linux:* ผู้ใช้อาจต้องเพิ่มสิทธิ์ในการเข้าถึงพอร์ตอนุกรม โดยการเพิ่ม user เข้าไปในกลุ่ม dialout (หรือ tty ในบาง Distribution) ด้วยคำสั่ง sudo usermod -a -G dialout $USER (แล้ว logout และ login ใหม่ หรือ reboot)

2.4 การติดตั้งเฟิร์มแวร์ MicroPython ลงบน ESP32

เฟิร์มแวร์ (Firmware) คือซอฟต์แวร์ที่ทำงานอยู่บนตัวฮาร์ดแวร์โดยตรง ในที่นี้คือระบบ MicroPython ที่จะทำให้ ESP32 สามารถเข้าใจและรันโค้ด Python ได้ โดยปกติบอร์ด ESP32 ที่ซื้อมาใหม่จะยังไม่มีเฟิร์มแวร์ MicroPython ติดตั้งอยู่ หรืออาจจะมีเฟิร์มแวร์อื่น (เช่น AT Command firmware หรือ Arduino bootloader) ดังนั้น จึงจำเป็นต้องติดตั้งเฟิร์มแวร์ MicroPython ก่อน

2.4.1 ดาวน์โหลดเฟิร์มแวร์ MicroPython

1. ไปที่เว็บไซต์ทางการของ MicroPython: https://micropython.org/download/esp32/



2. มองหาส่วน "Firmware for ESP32 boards"
3. ดาวน์โหลดไฟล์เฟิร์มแวร์เวอร์ชันล่าสุดที่เป็น **Stable** (แนะนำ) หรือ Daily builds (หากต้องการลองฟีเจอร์ใหม่ล่าสุด แต่อาจไม่เสถียร) ไฟล์เฟิร์มแวร์สำหรับ ESP32 ทั่วไปจะมีนามสกุล .bin (เช่น ESP32_GENERIC-YYYYMMDD-vx.xx.x.bin)

คำแนะนำ: สำหรับบอร์ด ESP32 ทั่วไปที่ใช้ชิป ESP32-WROOM หรือ ESP32-SOLO ให้เลือกเฟิร์มแวร์ GENERIC หากใช้บอร์ดที่มีคุณสมบัติพิเศษ เช่น SPIRAM (PSRAM) อาจมีเฟิร์มแวร์เฉพาะ เช่น GENERIC_SPIRAM ควรตรวจสอบสเปกของบอร์ดที่ใช้

- On Windows, the port name is usually similar to COM4.

Flashing

Then deploy the firmware to the board, starting at address 0x1000:

```
esptool.py --baud 460800 write_flash 0x1000 ESP32_BOARD_NAME-DATE-VE
RSION.bin
```

Replace ESP32_BOARD_NAME-DATE-VERSION.bin with the .bin file downloaded from this page.

As above, if esptool.py can't automatically detect the serial port then you can pass it explicitly on the command line instead. For example:

```
esptool.py --port PORTNAME --baud 460800 write_flash 0x1000 ESP32_BO
ARD_NAME-DATE-VERSION.bin
```

Troubleshooting

If flashing starts and then fails partway through, try removing the --baud 460800 option to flash at the slower default speed.

If these steps don't work, consult the MicroPython ESP32 Troubleshooting steps and the esptool documentation.

Important: From the options below, download the .bin file for your board.

Firmware

Releases

v1.25.0 (2025-04-15) .bin / [.app-bin] / [.elf] / [.map] / [Release notes] (latest)
v1.24.1 (2024-11-29) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.24.0 (2024-10-25) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.23.0 (2024-06-02) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.2 (2024-02-22) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.1 (2024-01-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.0 (2023-12-27) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.21.0 (2023-10-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.20.0 (2023-04-26) .bin / [.elf] / [.map] / [Release notes]
v1.19.1 (2022-06-18) .bin / [.elf] / [.map] / [Release notes]
v1.18 (2022-01-17) .bin / [.elf] / [.map] / [Release notes]
v1.17 (2021-09-02) .bin / [.elf] / [.map] / [Release notes]
v1.16 (2021-06-23) .bin / [.elf] / [.map] / [Release notes]
v1.15 (2021-04-18) .bin / [.elf] / [.map] / [Release notes]
v1.14 (2021-02-02) .bin / [.elf] / [.map] / [Release notes]
v1.13 (2020-09-02) .bin / [.elf] / [.map] / [Release notes]
v1.12 (2019-12-20) .bin / [.elf] / [.map] / [Release notes]

Preview builds

v1.26.0-preview.158.g5cfafb73d (2025-05-28) .bin / [.app-bin] / [.elf] / [.map]
v1.26.0-preview.148.g49f81d504 (2025-05-22) .bin / [.app-bin] / [.elf] / [.map]
v1.26.0-preview.146.g7f6fedef2 (2025-05-21) .bin / [.app-bin] / [.elf] / [.map]
v1.26.0-preview.142.ga1ee42cd3 (2025-05-21) .bin / [.app-bin] / [.elf] / [.map]
(These are automatic builds of the development branch for the next release)

Firmware (Support for OTA)

Releases

v1.25.0 (2025-04-15) .bin / [.app-bin] / [.elf] / [.map] / [Release notes] (latest)
v1.24.1 (2024-11-29) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.24.0 (2024-10-25) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.23.0 (2024-06-02) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.2 (2024-02-22) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.1 (2024-01-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.0 (2023-12-27) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.21.0 (2023-10-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.20.0 (2023-04-26) .bin / [.elf] / [.map] / [Release notes]
v1.19.1 (2022-06-18) .bin / [.elf] / [.map] / [Release notes]
v1.18 (2022-01-17) .bin / [.elf] / [.map] / [Release notes]

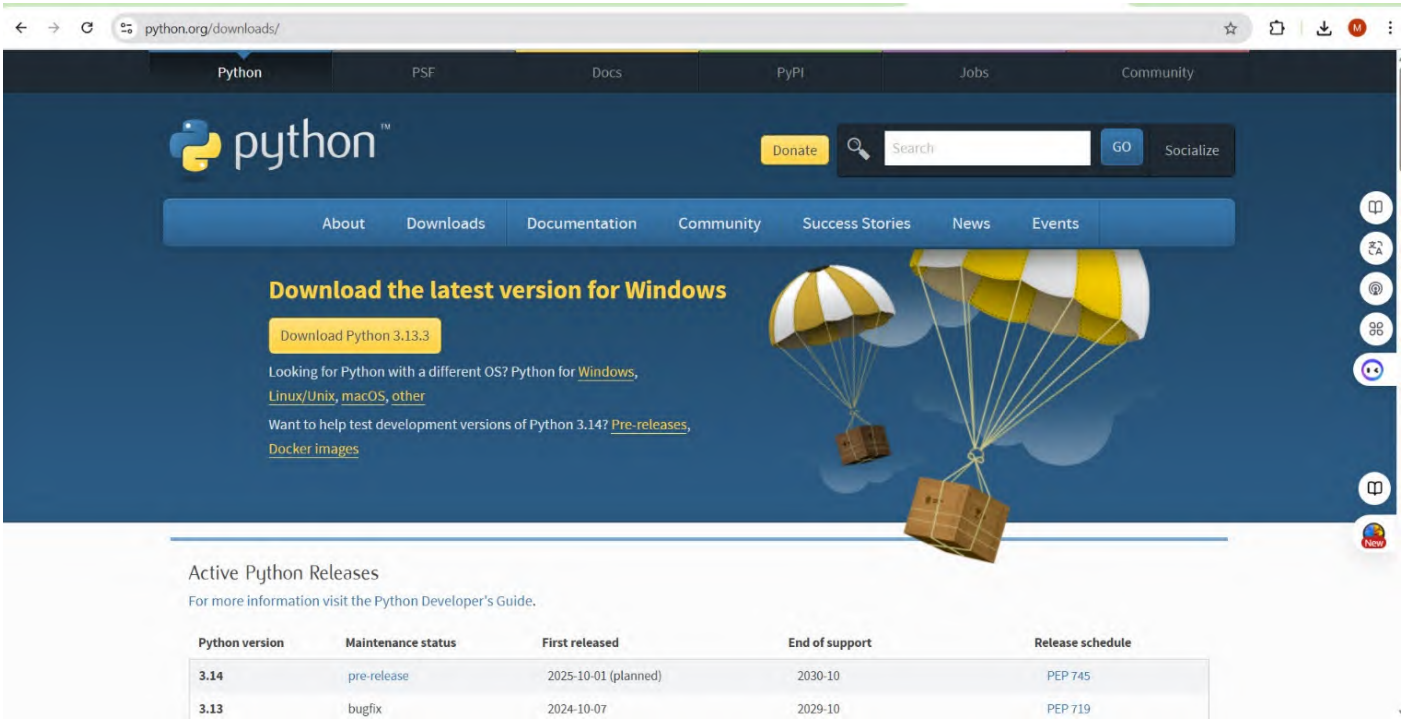
เครื่องมือที่ใช้ในการอัปโหลดเฟิร์มแวร์ มีทั้งแบบพิมพ์คำสั่ง Command-Line และ GUI เลือกใช้งานได้ตามถนัดหากถนัดแบบพิมพ์คำสั่งก็ต้องติดตั้ง esptool.py แต่ถ้าหากว่าการพิมพ์คำสั่งมันดูยุ่งยากซับซ้อนเกินไป ก็ให้ข้ามขั้นตอนนี้ไป ดาวน์โหลดโปรแกรมแบบ GUI มาติดตั้งซึ่งใช้งานง่ายกว่า

แต่อย่างไรก็ดี ต้องติดตั้ง python ตามข้อ1 ก่อน ส่วน esptool.py ไม่จำเป็นต้องติดตั้งก็ได้

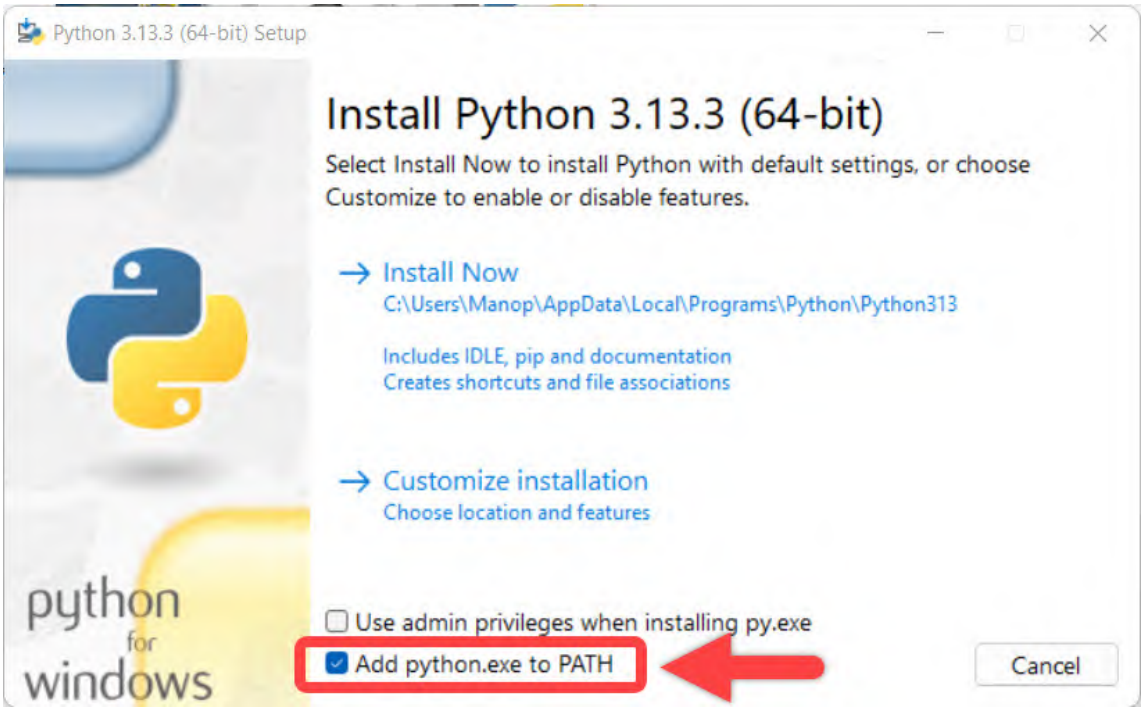
2.4.2 ติดตั้ง Python และ esptool.py

esptool.py เป็นเครื่องมือที่พัฒนาด้วย Python สำหรับสื่อสารกับ Bootloader ของชิป Espressif (ESP8266, ESP32, ESP32-S2, etc.) เพื่ออัปโหลดเฟิร์มแวร์, อ่านข้อมูลจากชิป และอื่นๆ

1. **ต้องมี Python ติดตั้งบนคอมพิวเตอร์:** หากยังไม่มี ให้ดาวน์โหลดและติดตั้ง Python 3.x จาก https://www.python.org/downloads/

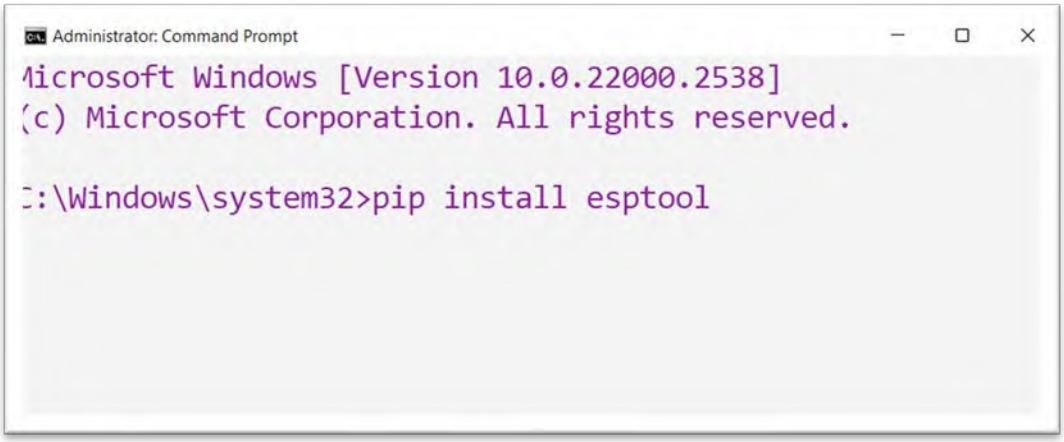


ในระหว่างการติดตั้งบน Windows แนะนำให้เลือก "Add Python to PATH"

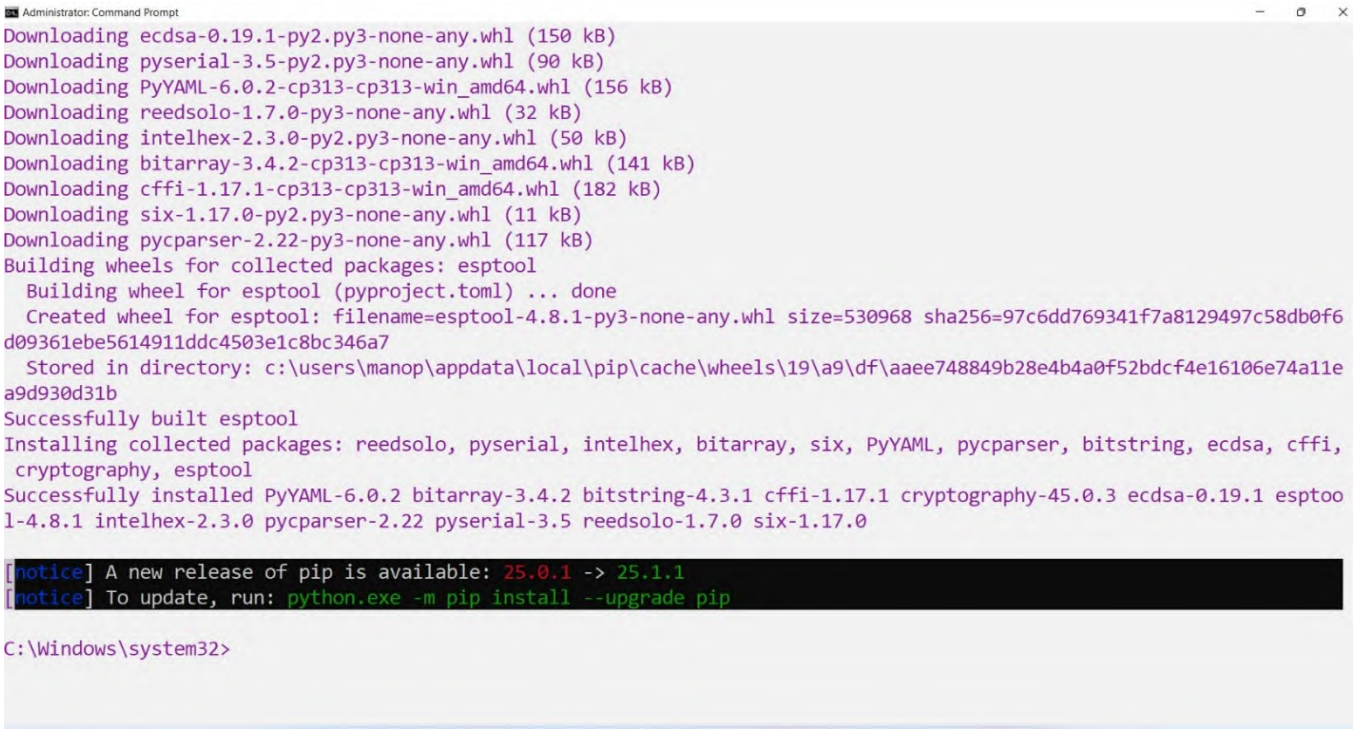


2. **ติดตั้ง esptool.py ผ่าน pip:** เปิด Command Prompt (Windows) หรือ Terminal (macOS/Linux) แล้วพิมพ์คำสั่ง:

pip install esptool แล้วกดปุ่ม **Enter**

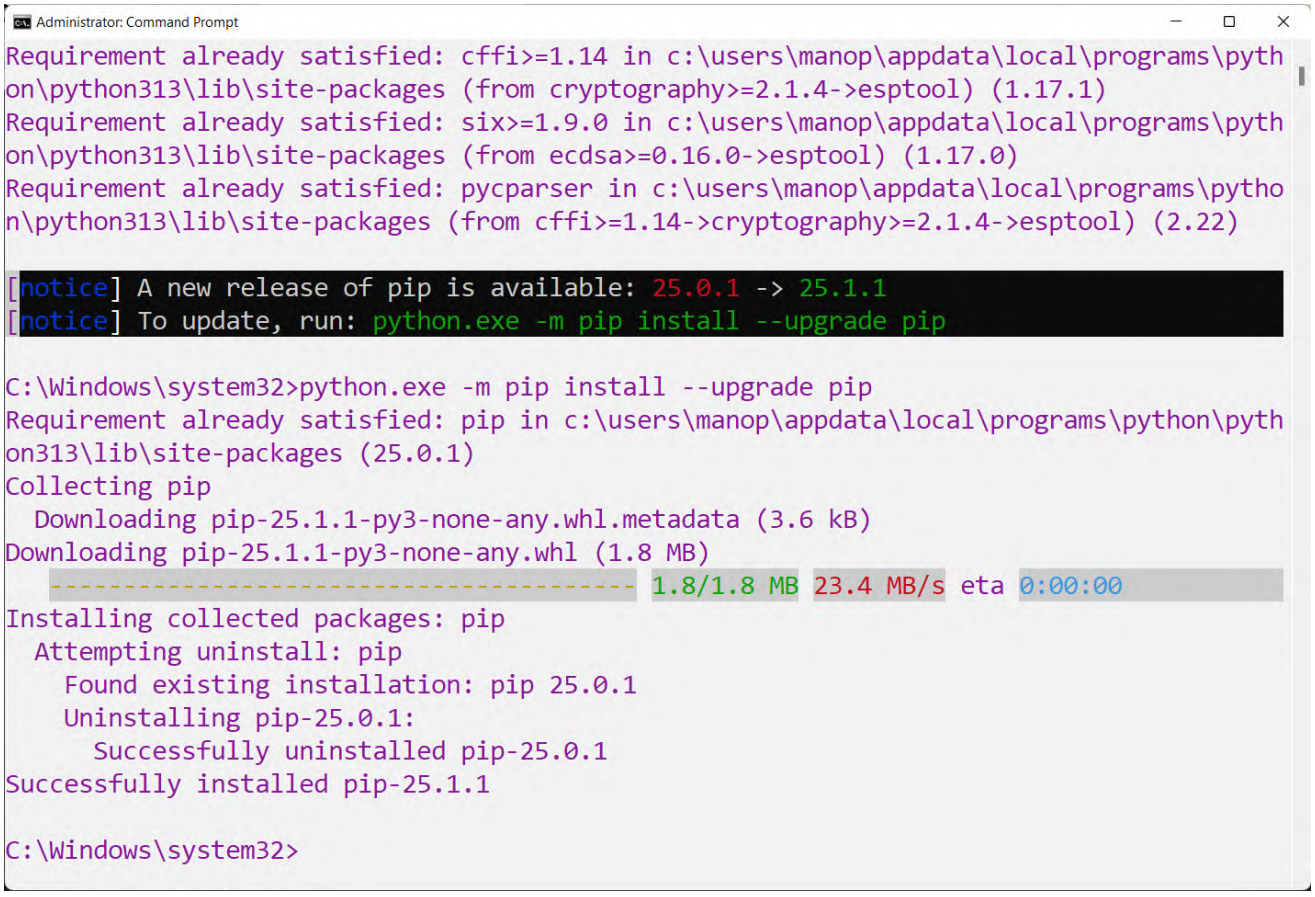


เมื่อติดตั้งเสร็จแล้วจะมีข้อความแจ้งในหน้าต่าง พร้อมกับแสดงเวอร์ชันที่ติดตั้ง



หากต้องการหากต้องการอัปเกรดเป็นเวอร์ชันล่าสุด พิมพ์ **python.exe -m pip install –upgrade pip**

เสร็จแล้วกดปุ่ม Enter อีกครั้งแล้วรอจนกว่าจะมีข้อความแจ้ง



3. หากต้องการตรวจสอบเวอร์ชันให้พิมพ์ esptool version แล้วกด Enter

2.4.3 การเชื่อมต่อบอร์ดและค้นหาพอร์ต

1. เสียบบอร์ด ESP32 เข้ากับคอมพิวเตอร์ผ่านสาย USB
2. ค้นหาหมายเลขพอร์ต (Port) ที่บอร์ด ESP32 เชื่อมต่ออยู่ (ตามวิธีในหัวข้อ 2.3)
   - Windows: เช่น COM3, COM4
   - macOS: เช่น /dev/cu.usbserial-xxxx หรือ /dev/cu.SLAB_USBtoUART
   - Linux: เช่น /dev/ttyUSB0

2.4.4 การลบข้อมูลใน Flash (Erasing the flash)

ก่อนติดตั้งเฟิร์มแวร์ใหม่ แนะนำให้ลบข้อมูลทั้งหมดในหน่วยความจำ Flash ของ ESP32 ก่อน เพื่อป้องกันปัญหาที่อาจเกิดจากข้อมูลเก่าที่หลงเหลืออยู่ ในส่วนของการ ลบเฟิร์มแวร์นั้น สามารถทำได้ทั้งวิธี Command Line หรือ ใช้โปรแกรมสำเร็จรูปแบบ GUI ก็ได้เหมือนกัน ส่วนนนี้แล้วแต่คนถนัดการใช้โปรแกรมแบบ GUI ต้องดาวน์โหลดโปรแกรมมาก่อน แต่ถ้าจะใช้แบบ command line ก็เปิด terminal พิมพ์คำสั่งได้เลย แต่ต้องดูหมายเลข port ที่เชื่อมต่อด้วย

1. เปิด Command Prompt หรือ Terminal
2. พิมพ์คำสั่ง esptool.py โดยแทนที่ [PORT] ด้วยหมายเลขพอร์ตที่ถูกต้องด้วย

esptool.py --chip esp32 --port [PORT] erase_flash

ตัวอย่างสำหรับ Windows: esptool.py --chip esp32 --port COM3 erase_flash

ตัวอย่างสำหรับ macOS/Linux: esptool.py --chip esp32 --port /dev/ttyUSB0 erase_flash

นี่คือการแฟลชเฟิร์มแวร์ด้วยการพิมพ์คำสั่ง Command-Line

หากไม่ถนัดการพิมพ์คำสั่งแบบ **command line** ก็มีทางเลือก:

🔧 **ทางเลือกการใช้เครื่องมือแบบ GUI ที่ใช้งานง่ายกว่า**

หากไม่ต้องการพิมพ์คำสั่ง ก็ใช้โปรแกรม GUI เหล่านี้แทนได้เลย จะปลอดภัย และ ใช้งานง่ายกว่า

✅ **1. ESPHome-Flasher**

- มีปุ่ม "Erase Flash" โดยตรง
- ใช้งานง่าย เหมาะกับผู้ไม่ถนัด Command Line

📌 ใช้ได้กับ: ESP32 / ESP8266

🔗 https://github.com/esphome/esphome-flasher/releases

✅ **2. ESP32 Flash Download Tool (Espressif Official)**

- มี checkbox ให้เลือก **“Erase Flash”**
- รองรับพาร์ติชันแบบละเอียด

📌 ใช้ได้กับ: ESP32 โดยตรง

🔗 https://www.espressif.com/en/support/download/other-tools

✅ **3. NodeMCU PyFlasher (สำหรับ ESP8266)**

- มีปุ่ม **“Erase Flash before flashing”**
- ใช้งานง่ายมากในสาย MicroPython

แต่ในหนังสือเล่มนี้ผู้เขียนขอเลือกใช้ **ESP32 Flash Download Tool** มาเป็นตัวอย่าง เพราะเป็นโปรแกรมของ **Espressif** เอง

ให้ผู้อ่านเข้าไปในเว็บไซต์ https://www.espressif.com/en/support/download/other-tools