

บทที่ 11

การเขียนโปรแกรมเบื้องต้นสำหรับหุ่นยนต์

จุดประสงค์ทั่วไป

1. เพื่อให้ นักศึกษามีความรู้ความเข้าใจเกี่ยวกับการเขียนโปรแกรมเบื้องต้นสำหรับหุ่นยนต์

จุดประสงค์เชิงพฤติกรรม (เพื่อให้ผู้เรียน)

1. อธิบายหลักการเขียนโปรแกรมเบื้องต้นสำหรับหุ่นยนต์ได้อย่างถูกต้อง
2. อธิบายขั้นตอนการเขียนโปรแกรมเบื้องต้นสำหรับหุ่นยนต์ได้อย่างถูกต้อง

ทฤษฎีการเรียนรู้งานการเขียนโปรแกรมเบื้องต้นสำหรับหุ่นยนต์

1. ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์(อังกฤษ : microcontroller มักย่อว่า μC , uC หรือ MCU) คือ อุปกรณ์ควบคุมขนาดเล็ก ซึ่งบรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์โดยไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู, หน่วยความจำ และพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน โดยทำการบรรจุเข้าไว้ในตัวถังเดียวกัน โครงสร้างโดยทั่วไป ของไมโครคอนโทรลเลอร์นั้นสามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ๆ ดังต่อไปนี้

1.1. หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)

1.2. หน่วยความจำ (Memory) สามารถแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำที่มีไว้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของเครื่องคอมพิวเตอร์ตั้งโต๊ะ คือ ข้อมูลใดๆ ที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยง อีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกับกระดานจดในการคำนวณของซีพียู และเป็นที่พักข้อมูลชั่วคราวขณะทำงาน แต่หากไม่มีไฟเลี้ยง ข้อมูลก็จะหายไปคล้ายกับหน่วยความจำแรม (RAM) ในเครื่องคอมพิวเตอร์ทั่วไป แต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่ หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแรม ซึ่งข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Electrically Erasable Programmable Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยงก็ตาม

1.3. ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port) มี 2 ลักษณะคือ พอร์ตอินพุต (Input Port)

และพอร์ตส่งสัญญาณหรือพอร์ตเอาต์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอก ถือว่าเป็นส่วนที่สำคัญมาก ใช้ร่วมกันระหว่างพอร์ตอินพุต เพื่อรับสัญญาณ อาจจะ

ด้วยการกดสวิตช์ เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาต์พุต เพื่อแสดงผลเช่น การติดสว่างของหลอดไฟ เป็นต้น

1.4. ช่องทางเดินของสัญญาณ หรือบัส (BUS) คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง ซีพียู หน่วยความจำและพอร์ต เป็นลักษณะของสายสัญญาณ จำนวนมากอยู่ภายในตัวไมโครคอนโทรลเลอร์ โดยแบ่งเป็นบัสข้อมูล (Data Bus) , บัสแอดเดรส (Address Bus) และบัสควบคุม (Control Bus)

1.5. วงจร านิตสัญญาณนาฬิกา เป็นองค์ประกอบที่สำคัญมากอีกส่วนหนึ่ง เนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์ จะขึ้นอยู่กับกรก หนดจังหวะ หากสัญญาณนาฬิกามีความถี่สูง จังหวะการทำงานก็จะสามารถทำได้ถี่ขึ้นส่งผลให้ไมโครคอนโทรลเลอร์ตัวนั้น มีความเร็วในการประมวลผลสูงตามไปด้วย

ที่มา : <https://th.wikipedia.org/wiki/ไมโครคอนโทรลเลอร์>

ส่วนพิเศษอื่นๆ จะขึ้นอยู่กับกระบวนการผลิตของแต่ละบริษัทที่จะผลิตขึ้นมา ใส่คุณสมบัติพิเศษลงไปเช่น

- **ADC** (Analog to Digital) ส่วนภาครับสัญญาณอนาล็อกแปลงไปเป็นสัญญาณดิจิทัล - **DAC** (Digital to Analog) ส่วนภาคส่งสัญญาณดิจิทัลแปลงไปเป็นสัญญาณอนาล็อก - **I2C** (Inter Integrate Circuit Bus) เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) เพื่อใช้ ติดต่อสื่อสาร ระหว่าง ไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอก ซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือ serial data (SDA) และสาย serial clock (SCL) ซึ่งสามารถ เชื่อมต่ออุปกรณ์ จำนวนหลายๆ ตัว เข้าด้วยกันได้ ทำให้ MCU ใช้พอร์ตเพียง 2 พอร์ตเท่านั้น
- **SPI** (Serial Peripheral Interface) เป็นการเชื่อมต่อกับอุปกรณ์เพื่อรับส่งข้อมูลแบบซิงโครนัส (Synchronize) มีสัญญาณนาฬิกาเข้ามาเกี่ยวข้องกับระหว่างไมโครคอนโทรลเลอร์ (Microcontroller) หรือจะเป็นอุปกรณ์ภายนอกที่มีการรับส่งข้อมูลแบบ SPI อุปกรณ์ที่ทำหน้าที่เป็นมาสเตอร์ (Master) โดยปกติแล้วจะเป็นไมโครคอนโทรลเลอร์ หรืออาจกล่าวได้ว่าอุปกรณ์ Master จะต้องควบคุมอุปกรณ์ Slave ได้ โดยปกติตัว Slave มักจะเป็นไอซี (IC) หน้าที่พิเศษต่างๆ เช่น ไอซีอุณหภูมิ, ไอซีฐานเวลานาฬิกาจริง (Real-Time Clock) หรืออาจเป็นไมโครคอนโทรลเลอร์ ที่ทำหน้าที่ในโหมด Slave ก็ได้เช่นกัน
- **PWM** (Pulse Width Modulation) การสร้างสัญญาณพัลส์แบบสแควร์เวฟ ที่สามารถปรับเปลี่ยนความถี่และ Duty Cycle ได้เพื่อนำไปควบคุมอุปกรณ์ต่างๆ เช่น มอเตอร์

- **UART** (Universal Asynchronous Receiver Transmitter) ทำหน้าที่รับส่งข้อมูลแบบอะซิงโครนัสสำหรับมาตรฐานการรับส่งข้อมูลแบบ RS-232

ไมโครคอนโทรลเลอร์ มีด้วยกันหลายประเภทแบ่งตามสถาปัตยกรรม (การผลิตและกระบวนการทำงานระบบการประมวลผล) ที่มีใช้ในปัจจุบันยกตัวอย่างดังนี้

1. ไมโครคอนโทรลเลอร์ตระกูล PIC (บริษัทผู้ผลิต Microchip ไมโครชิป)
2. ไมโครคอนโทรลเลอร์ตระกูล MCS51 (บริษัทผู้ผลิต Atmel, Phillips)
3. ไมโครคอนโทรลเลอร์ตระกูล AVR (บริษัทผู้ผลิต Atmel)
4. ไมโครคอนโทรลเลอร์ตระกูล ARM7, ARM9 (บริษัทผู้ผลิต Atmel, Phillips, Analog Device, Sumsung, STMicroelectronics)
5. ไมโครคอนโทรลเลอร์ตระกูล Basic Stamp (บริษัทผู้ผลิต Parallax)
6. ไมโครคอนโทรลเลอร์ตระกูล PSOC (บริษัทผู้ผลิต CYPRESS)
7. ไมโครคอนโทรลเลอร์ตระกูล MSP (บริษัทผู้ผลิต Texas Instruments)
8. ไมโครคอนโทรลเลอร์ตระกูล 68HC (บริษัทผู้ผลิต MOTOROLA)
9. ไมโครคอนโทรลเลอร์ตระกูล H8 (บริษัทผู้ผลิต Renesas)
10. ไมโครคอนโทรลเลอร์ตระกูล RABBIT (บริษัทผู้ผลิต RABBIT SEMICONDUCTOR)
11. ไมโครคอนโทรลเลอร์ตระกูล Z80 (บริษัทผู้ผลิต Zilog)

ภาษาที่ใช้เขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ มีดังนี้

1. ภาษา Assembly
2. ภาษา Basic
3. ภาษา C
4. ภาษา Pascal
5. ภาษา Java
6. ภาษาอื่น ๆ อีกหลากหลายภาษาซึ่งได้รับการพัฒนาอย่างต่อเนื่อง

ภาษาดังกล่าวที่กล่าวในเบื้องต้น ไมโครคอนโทรลเลอร์บางตระกูล จะใช้ได้ครบทุกภาษา แต่บางตระกูลจะได้บางภาษาขึ้นอยู่กับบริษัทผู้ผลิต Software (โดยทั่วไปจะเรียกว่า Editor And Compiler) ที่ใช้เขียนภาษาไมโครคอนโทรลเลอร์นั้นจะผลิตออกมาให้ Support หรือไม่ ที่มา :

<http://www.engineer007.com/articles/507518/>

2. Arduino

Arduino คือ โครงการที่นำชิปไอซีไมโครคอนโทรลเลอร์ตระกูลต่างๆ มาใช้ร่วมกันในภาษา C ซึ่งภาษา C นี้เป็นลักษณะเฉพาะ คือมีการเขียนไลบรารีของ Arduino ขึ้นมาเพื่อให้การสั่งงานไมโครคอนโทรลเลอร์ที่แตกต่างกัน สามารถใช้งานโค้ดตัวเดียวกันได้ โดยตัวโครงการได้ออกบอร์ดทดลองมาหลายรูปแบบ เพื่อใช้งานกับ IDE ของตนเอง สาเหตุหลักที่ทำให้ Arduino เป็นนิยมนมาก เป็นเพราะซอฟต์แวร์ที่ใช้งานร่วมกันสามารถโหลดได้ฟรี และตัวบอร์ดทดลองยังถูกแจกแถม ทำให้ผู้ผลิตเงินนำไปผลิตและขายออกตลาดมาในราคาที่ถูกมากๆ โดยบอร์ดที่ถูกที่สุดในตอนนี้คือบอร์ด Arduino ที่มีราคาเพียง 120 – 150 บาทเท่านั้น Arduino ต่างอะไรกับ AVR แบบปกติ



รูปที่ 11.1 แสดงตัวอย่าง ชิป AVR แบบต่าง ๆ

จากที่ได้กล่าวไปแล้ว Arduino นั้นได้ใช้ชิป AVR เป็นหลักใน Arduino แทบรุ่น สาเหตุมาจากไมโครคอนโทรลเลอร์ของตระกูล AVR นั้นมีความทันสมัย ในชิปในบางตัวสามารถเชื่อมต่อผ่าน USB ได้โดยตรง สามารถใช้กับคอมพิวเตอร์สมัยใหม่ได้เป็นอย่างดี และในไมโครคอนโทรลเลอร์ตระกูล AVR ยังมีส่วนของโปรแกรมพิเศษที่เรียกว่า Bootloader อยู่ในระดับต่ำกว่าส่วนโปรแกรมปกติ ซึ่งจะเป็นส่วนโปรแกรมที่จะถูกเรียกขึ้นมาก่อนการเรียกโปรแกรมปกติ ทำให้สามารถเขียนสั่งให้ทำงานใดๆก็ได้ ก่อนการเรียกโปรแกรมปกติ ทำให้ Arduino นั้นอาศัยส่วนโปรแกรมพิเศษนี้ในการทำให้ชิปสามารถโปรแกรมผ่านพอร์ตอนุกรมชนิด UART ได้ จึงทำให้การเขียนโปรแกรมลงไปในชิปใช้เพียง USB to UART ก็เพียงพอแล้ว แต่การโปรแกรมด้วยการใช้โปรโตคอล UART ก็มีข้อเสียตรงที่ต้องใช้เวลาในการบูตเข้าโปรแกรมปกติประมาณ 1 – 2 วินาทีบอร์ด Arduino มีด้วยกันหลักๆ ประมาณ 9 บอร์ด ดังนี้

2.1. Arduino Uno



รูปที่ 11.2 บอร์ด Arduino Uno R3

คำว่า Uno เป็นภาษาอิตาลี ซึ่งแปลว่าหนึ่ง เป็นบอร์ด Arduino รุ่นแรกที่ยังคงมีขนาดประมาณ 68.6x53.4mm เป็นบอร์ดมาตรฐานที่นิยมใช้งานมากที่สุด เนื่องจากเป็นขนาดที่เหมาะสมสำหรับการเริ่มต้นเรียนรู้ Arduino และมี Shields ให้เลือกใช้งานได้มากกว่าบอร์ด Arduino รุ่นอื่นๆ ที่ออกแบบมาเฉพาะมากกว่า โดยบอร์ด Arduino Uno ได้มีการพัฒนาเรื่อยมา ตั้งแต่ R2 R3 และรุ่นย่อยที่เปลี่ยนชิปไอซีเป็นแบบ SMD



รูปที่

11.3 บอร์ด Arduino Uno R2 รูปที่ 11.4 บอร์ด Arduino Uno SMD

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรลเลอร์ ATmega328

ใช้แรงดันไฟฟ้า 5V

รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ) 7 – 12V

รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด) 6 – 20V

พอร์ต Digital I/O 14 พอร์ต (มี 6 พอร์ต PWM output) พอร์ต Analog

Input 6 พอร์ต

กระแสไฟที่จ่ายได้ในแต่ละพอร์ต 40mA

กระแสไฟที่จ่ายได้ในพอร์ต 3.3V 50mA

พื้นที่โปรแกรมภายใน 32KB พื้นที่โปรแกรม, 500B ใช้โดย Bootloader พื้นที่แรม 2KB

พื้นที่หน่วยความจำถาวร (EEPROM) 1KB

ความถี่คริสตัล 16MHz

ขนาด 68.6x53.4 mm

น้ำหนัก 25 กรัม

2.2. Arduino Duo



รูปที่ 11.5 บอร์ด Arduino Duo

Duo เป็นภาษาอิตาลี แปลว่า สอง เป็นรุ่นที่เพิ่มพอร์ตให้มากขึ้นเป็น 54 พอร์ตดิจิตอลอินพุต เอาต์พุต และ 12 พอร์ตอนาล็อกอินพุต 2 พอร์ตอนาล็อกเอาต์พุต เพิ่มพื้นที่โปรแกรมเป็น 512KB สามารถใช้งานพื้นที่ได้เต็มไม่มี Bootloader เนื่องจากสามารถใช้กับพอร์ต USB ได้โดยตรง มีขนาดบอร์ด 101.52x53.3mm สามารถใช้ Shields ของ Arduino Uno ได้ แต่บางตัวจำเป็นต้องแก้ไขให้ถูกต้อง จากรูปที่ 4 จะเห็นว่าบอร์ดได้เปลี่ยนมาใช้ชิปไอซีแบบ SMD จึงไม่นิยมนำมาใช้ในแบบ Standalone แต่นิยมนำมาใช้ในงานที่จำเป็นต้องพื้นที่โปรแกรมมากขึ้น ทำงานที่ซับซ้อนมากขึ้น บอร์ด Arduino Duo ใช้ชิปไอซีเบอร์ AT91SAM3X8E ซึ่งเป็นชิปไอซีที่ใช้เทคโนโลยี ARM Core สถาปัตยกรรม 32 บิต แรงความถี่คริสตัลขึ้นไปถึง 84Mhz จึงทำให้สามารถงานด้านการ คำนวณ หรือการประมวลผลอัลกอริทึมได้เร็วกว่า Arduino Uno มาก แต่เนื่องจากชิปไอซีทำงานที่

แรงดัน 3.3V ดังนั้นการนำไปใช้งานกับเซ็นเซอร์ควรระวังไม่ให้แรงดัน 5V ไหลเข้าบอร์ด ควรใช้วงจรแบ่งแรงดันเพื่อช่วยให้ลอจิกลดแรงดันลงมาให้เหมาะสม

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรลเลอร์ AT91SAM3X8E

ใช้แรงดันไฟฟ้า 3.3V

รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ) 7 – 12V

รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด) 6 – 16V

พอร์ต Digital I/O 54 พอร์ต (มี 12 พอร์ต PWM output) พอร์ต Analog Input 2 พอร์ต

กระแสไฟรวมที่จ่ายได้ในทุกพอร์ต 130mA

กระแสไฟที่จ่ายได้ในพอร์ต 3.3V 800mA

กระแสไฟที่จ่ายได้ในพอร์ต 5V 800mA

พื้นที่โปรแกรมภายใน 512KB พื้นที่โปรแกรม

พื้นที่แรม 2KB

พื้นที่หน่วยความจำถาวร (EEPROM) 96KB

ความถี่คริสตัล 84MHz

ขนาด 101.52x53.3 mm

น้ำหนัก 36 กรัม

2.3. Arduino Leonardo



รูปที่ 11.6 บอร์ด Arduino Leonard

บอร์ด Arduino Leonard เป็นบอร์ดที่เลือกใช้ชิปไอซีเบอร์ ATmega32u4 ที่รองรับการเชื่อมต่อกับพอร์ต USB ได้โดยตรง ทำให้บอร์ดสามารถเขียนโปรแกรมเพื่อจำลองตัวเองให้เป็นเมาส์หรือคีย์บอร์ดได้ ทำงานที่แรงดัน 5V ทำให้ไม่มีปัญหากับเซ็นเซอร์ หรือ Shields ที่ใช้งานกับ Arduino Uno

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรเลอร์ ATmega32u4

ใช้แรงดันไฟฟ้า 5V

รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ) 7 – 12V

รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด) 6 – 20V

พอร์ต Digital I/O 20 พอร์ต (มี 7 พอร์ต PWM output) พอร์ต Analog Input 12 พอร์ต

กระแสไฟฟารวมที่จ่ายได้ในทุกพอร์ต 40mA

กระแสไฟที่จ่ายได้ในพอร์ต 3.3V 50mA

พื้นที่โปรแกรมภายใน 32KB แต่ 4KB ถูกใช้โดย Bootloader พื้นที่แรม 2.5KB

พื้นที่หน่วยความจำถาวร (EEPROM) 1KB

ความถี่คริสตัล 16MHz

ขนาด 68.6x53.3 mm

น้ำหนัก 20 กรัม

2.4. Arduino MEGA ADK



รูปที่ 11.7 บอร์ด Arduino MEGA ADK R3

บอร์ด Arduino MEGA ADK ใช้ชิปไมโครคอนโทรเลอร์เบอร์ ATmega2560 มีชิปไอซี USB Host เบอร์ MAX3421e มาให้บนบอร์ด ใช้สำหรับเชื่อมต่อกับโทรศัพท์มือถือแอนดรอยด์ผ่าน OTG มีพอร์ตดิจิตอลอินพุตเอาต์พุตจำนวน 54 พอร์ต มีนาฬิกาอินพุตมาให้ 16 พอร์ต ทำงานที่ ความถี่ 16MHz บอร์ด Arduino MEGA ADK จะแตกต่างกับบอร์ด Arduino Duo ตรงที่ชิปบนบอร์ดนั้นฉลาดไม่เท่า และใช้ความถี่ต่ำกว่า ดังนั้นจึงไม่เหมาะจะนำไปใช้กับงานคำนวณ แต่เหมาะสำหรับงานที่ใช้การเชื่อมต่อกับโทรศัพท์มือถือแอนดรอยด์มากกว่า

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรเลอร์ ATmega2560

ใช้แรงดันไฟฟ้า 5V

รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ) 7 – 12V

รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด) 6 – 20V

พอร์ต Digital I/O 54 พอร์ต (มี 15 พอร์ต PWM output) พอร์ต Analog Input 16 พอร์ต

กระแสไฟฟ้ารวมที่จ่ายได้ในทุกพอร์ต 40mA

กระแสไฟที่จ่ายได้ในพอร์ต 3.3V 50mA

พื้นที่โปรแกรมภายใน 256KB แต่ 8KB ถูกใช้โดย Bootloader พื้นที่แรม 8KB

พื้นที่หน่วยความจำถาวร (EEPROM) 4KB

ความถี่คริสตัล 16MHz

ขนาด 101.52x53.3 mm

น้ำหนัก 36 กรัม

2.5. Arduino Mega 2560

รูปที่ 11.8 บอร์ด Arduino Mega 2560 R3

บอร์ด Arduino Mega 2560 จะเหมือนกับ Arduino MEGA ADK ต่างกันตรงที่บนบอร์ดไม่มี USB Host มาให้ การโปรแกรมยังต้องทำผ่านโปรโตคอลUART อยู่บนบอร์ดใช้ชิปไอซีไมโครคอนโทรเลอร์เบอร์ ATmega2560

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรเลอร์ ATmega2560

ใช้แรงดันไฟฟ้า 5V

รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ) 7 – 12V

รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด) 6 – 20V

พอร์ต Digital I/O 54 พอร์ต (มี 15 พอร์ต PWM output) พอร์ต Analog

Input 16 พอร์ต

กระแสไฟฟารวมที่จ่ายได้ในทุกพอร์ต 40mA

กระแสไฟที่จ่ายได้ในพอร์ต 3.3V 50mA

พื้นที่โปรแกรมภายใน 256KB แต่ 8KB ถูกใช้โดย Bootloader พื้นที่แรม 8KB

พื้นที่หน่วยความจำถาวร (EEPROM) 4KB

ความถี่คริสตัล 16MHz

2.6. Arduino Micro

รูปที่ 11.9 บอร์ด Arduino Micro

บอร์ด Arduino Micro ออกแบบให้มีขนาดเล็ก และทันสมัยกว่าบอร์ด Arduino Mini หรือ Arduino Nano เนื่องจากบนบอร์ดใช้ชิปไอซีไมโครคอนโทรเลอร์เบอร์ ATmega32u4 ซึ่งมา พอร์ต USB สามารถเชื่อมต่อกับคอมพิวเตอร์ได้โดยตรง และมีดิจิตอลอินพุตเอาต์พุตมากถึง 20 พอร์ต มีพื้นที่เก็บโปรแกรมขนาด 32KB แต่ต้องใช้พื้นที่สำหรับ Bootloader ไป 4KB มีขนาดเพียง 48x18mm เนื่องจากบอร์ดใช้ชิปไอซีตัวเดียวกับ Arduino Leonardo ทำให้สามารถทำให้บอร์ดจำลองตัวเองเป็นเมาส์หรือคีย์บอร์ดเชื่อมต่อกับคอมพิวเตอร์ได้

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรเลอร์ ATmega32u4

ใช้แรงดันไฟฟ้า 5V

รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ) 7 – 12V

รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด) 6 – 20V

พอร์ต Digital I/O 20 พอร์ต (มี 7 พอร์ต PWM output) พอร์ต Analog

Input 12 พอร์ต

กระแสไฟที่จ่ายได้ในแต่ละพอร์ต 40mA

กระแสไฟที่จ่ายได้ในพอร์ต 3.3V 50mA

พื้นที่โปรแกรมภายใน 32KB พื้นที่โปรแกรม, 4KB ใช้โดย Bootloader พื้นที่แรม 2.5KB

พื้นที่หน่วยความจำถาวร (EEPROM) 1KB

ความถี่คริสตัล 16MHz

ขนาด 48x18 mm

น้ำหนัก 13 กรัม

2.7. Arduino Nano

รูปที่ 11.10 บอร์ด Arduino Nano

บอร์ด Arduino Nano ออกแบบมาให้มีขนาดเล็ก และใช้กับงานทั่วไป ใช้ชิปไอซี ไมโครคอนโทรลเลอร์เบอร์ ATmega168 หรือเบอร์ ATmega328 (มีรุ่น 2.3 กับ 3 ตอนซื้อต้องเช็คดีๆ ก่อน) โปรแกรมผ่านโปรโตคอล UART มีชิป USB to UART มาให้ ใช้ Mini USB เชื่อมต่อกับ คอมพิวเตอร์ มีพอร์ตดิจิตอลอินพุตเอาต์พุต 14 พอร์ต มีพอร์ตอนาล็อกอินพุต 8 พอร์ต บนบอร์ดยังมี เรกกูเลเตอร์ สามารถจ่ายไฟได้ตั้งแต่ 7 – 12V เพื่อให้บอร์ดทำงานได้ (จ่ายไฟที่ขา VIN)กรณีมี แหล่งจ่ายไฟ 5V อยู่แล้วก็จ่ายเข้าได้เลยที่ขา 5V

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรลเลอร์ ATmega168 หรือ ATmega328

ใช้แรงดันไฟฟ้า 5V

รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ) 7 – 12V

รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด) 6 – 20V

พอร์ต Digital I/O 14 พอร์ต (มี 6 พอร์ต PWM output) พอร์ต Analog Input

6 พอร์ต

กระแสไฟที่จ่ายได้ในแต่ละพอร์ต 40mA

กระแสไฟที่จ่ายได้ในพอร์ต 3.3V 50mA

พื้นที่โปรแกรมภายใน 16KB หรือ 32KB พื้นที่โปรแกรม, 500B ใช้โดย Bootloader

พื้นที่แรม 1 หรือ 2KB

พื้นที่หน่วยความจำถาวร (EEPROM) 512B หรือ 1KB

ความถี่คริสตัล 16MHz

ขนาด 45x18 mm

น้ำหนัก 5 กรัม

2.8. Arduino Mini

รูปที่ 11.11 บอร์ด Arduino Mini

บอร์ด Arduino Mini มีขนาดเล็กกว่าบอร์ด Arduino อื่นๆอยู่มาก แต่ยังคงความสามารถไว้เท่ากับบอร์ด Arduino Uno R3 แถมยังมีพอร์ต A6 และ A7 เพิ่มขึ้น ทำให้บอร์ดมีอนาล็อกอินพุตเพิ่มมากขึ้น จากเดิมมี 6 พอร์ต เพิ่มเป็น 8 พอร์ต เนื่องจากบอร์ด Arduino Mini เน้นที่ขนาดเล็ก ดังนั้นจึงไม่สามารถทำการโปรแกรมได้โดยตรง หากต้องการโปรแกรมบอร์ดจำเป็นต้องซื้อโมดูล USB to UART มาใช้แยกต่างหาก แต่ข้อดีของการไม่สามารถเสียบโปรแกรมได้โดยตรงคือหากโปรเจกต์อยู่ตัวแล้ว ความเสี่ยงที่จะถูกนำมาเขียนโปรแกรมเข้าไปใหม่ก็จะลดน้อยลง บอร์ด Arduino Mini ยังคงมีรูปแบบคล้ายๆกับ Arduino เดิม คือใช้ชิป ATmega328 ที่ความถี่ 16MHz ภายในบอร์ดสามารถใช้แหล่งจ่ายไฟ 7 – 12V มาจ่ายได้ หากมีแหล่งจ่ายไฟ 5V ก็สามารถนำมาจ่ายได้เลย

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรลเลอร์ ATmega328

ใช้แรงดันไฟฟ้า 5V

รองรับการจ่ายแรงดันไฟฟ้า (ที่แนะนำ) 7 – 12V

รองรับการจ่ายแรงดันไฟฟ้า (ที่จำกัด) 6 – 20V

พอร์ต Digital I/O 14 พอร์ต (มี 6 พอร์ต PWM output) พอร์ต Analog

Input 8 พอร์ต

กระแสไฟที่จ่ายได้ในแต่ละพอร์ต 40mA

พื้นที่โปรแกรมภายใน 32KB พื้นที่โปรแกรม, 2KB ใช้โดย Bootloader พื้นที่แรม 2KB

พื้นที่หน่วยความจำถาวร (EEPROM) 1KB

ความถี่คริสตัล 16MHz

ขนาด 30x18 mm

น้ำหนัก ไม่ระบุ

2.9. Arduino Pro Mini

รูปที่ 11.12 บอร์ด Arduino Pro Mini

บอร์ด Arduino Pro Mini เป็นบอร์ดที่แตกต่างจากบอร์ด Arduino Mini คือย้ายช่อง A4 A5 A6 A7 ออกมาภายในบอร์ด เพื่อให้บอร์ดมีขนาดเล็กกว่าเดิมอีก และมีให้เลือกใช้ทั้ง 5V และ 3.3V ก่อนซื้อจึงควรดูให้แน่ใจเสียก่อน บอร์ด Arduino Pro Mini ได้ใช้ไอซีเบอร์ ATmega328 เช่นเดิม แต่มีขนาดใหญ่ขึ้นเล็กน้อย ทำให้ไอซีดูเต็มบอร์ดมากขึ้น และในโมเดลที่ใช้แรงดันไฟ 3.3V ลดความถี่ลงเป็น 8MHz ใช้พื้นที่ Bootloader น้อยลง เหลือเพียง 500B การโปรแกรมยังคงต้องใช้โมดูล USB to UART ในการเชื่อมต่อเพื่อโปรแกรมเช่นเดิม

ข้อมูลจำเพาะ

ชิปไอซีไมโครคอนโทรเลอร์ ATmega328

ใช้แรงดันไฟฟ้า 3.3V หรือ 5V

รองรับการจ่ายแรงดันไฟฟ้า 3.35 – 12V (ในโมเดลใช้ไฟ 3.3V) หรือ 5 – 12V (ในโมเดลใช้ไฟ 5V)

พอร์ต Digital I/O 14 พอร์ต (มี 6 พอร์ต PWM output)

พอร์ต Analog Input 6 พอร์ต

กระแสไฟที่จ่ายได้ในแต่ละพอร์ต 40mA

พื้นที่โปรแกรมภายใน 32KB พื้นที่โปรแกรม, 500B ใช้โดย Bootloader พื้นที่แรม 2KB

พื้นที่หน่วยความจำถาวร (EEPROM) 1KB

ความถี่คริสตัล 8MHz (ในโมเดลใช้ไฟ 3.3V) หรือ 16MHz (ในโมเดลใช้ไฟ 5V)

3. การติดตั้งโปรแกรมและไดร์เวอร์

Arduino จะใช้โปรแกรมที่เรียกว่า Arduino IDE ในการเขียนโปรแกรม และคอมไพล์ลงบอร์ด โดยขนาดของโปรแกรม Arduino โดยปกติแล้วจะใหญ่กว่าโค้ด AVR ปกติเนื่องจากโค้ด AVR เป็นการเข้าถึงจากรีจิสเตอร์โดยตรง แต่โค้ด Arduino เข้าถึงผ่านฟังก์ชัน เพื่อให้สามารถเขียนโค้ดได้ง่ายมากกว่าการเขียนโค้ดแบบ AVR

3.1. การดาวน์โหลดโปรแกรม Arduino IDE

ดาวน์โหลดไฟล์โปรแกรมได้จากเว็บไซต์ <http://www.arduino.cc/en/Main/Software> เลือก ระบบปฏิบัติการที่ต้องการจะติดตั้ง (ตัวอย่างผมใช้ Windows 10 จึงเลือก Windows Installer)

รูปที่ 11.13 เว็บไซต์ <http://www.arduino.cc/en/Main/Software>

จากนั้นจึงแสดงหน้าต่างเชิญให้ร่วมบริจาค หากไม่ต้องการบริจาคสามารถคลิกปุ่ม JUST DOWNLOAD เพื่อเริ่มดาวน์โหลดโปรแกรมได้เลย

รูปที่ 11.14 แสดงตำแหน่งการเลือก JUST DOWNLOAD

การดาวน์โหลดจะเริ่มขึ้นอัตโนมัติ และรองนกว่าการดาวน์โหลดจะเสร็จสิ้น แล้วจึงทำขั้นตอนถัดไป

รูปที่ 11.15 แสดงการดาวน์โหลดโปรแกรม

3.2. การติดตั้งโปรแกรม Arduino IDE

3.2.1. ดาวน์โหลดเสร็จแล้วให้เปิดไฟล์ติดตั้งขึ้นมาได้เลย กดปุ่ม I Agree ได้เลย

รูปที่ 11.16 ตาแหน่งกดปุ่ม I Agree

3.2.2. มีตัวเลือกให้เลือกติดตั้ง แนะนำให้เลือกทั้งหมด (เลือกทั้งหมด) แล้วคลิกปุ่ม Next >

รูปที่ 11.17 การเลือก components ที่ต้องการติดตั้ง

3.2.3. เลือกตำแหน่งที่ติดตั้งโปรแกรม ให้เลือกตามที่โปรแกรมกำหนดมาให้ แล้วคลิกปุ่ม
Install

รูปที่ 11.18 เลือกตำแหน่งที่ติดตั้งโปรแกรม

3.2.4. โปรแกรมจะติดตั้ง รอจนกว่าจะเสร็จและโปรแกรมแจ้ง Completed ขึ้นมา

รูปที่ 11.19 แสดงการติดตั้งโปรแกรม

3.2.5. ติดตั้งเสร็จสมบูรณ์ คลิกปุ่ม Close เพื่อปิดโปรแกรมลงไปได้เลย

รูปที่ 11.20 แสดงผลการติดตั้งโปรแกรมสำเร็จ

3.2.6. ตรวจสอบ Desktop จะมีไอคอนโปรแกรม Arduino

รูปที่ 11.21 แสดง icon โปรแกรม Arduino IDE

3.3. การตั้งค่าโปรแกรม Arduino IDE

3.3.1. เปิดโปรแกรม Arduino IDE แสดงผลดังรูป

รูปที่ 11.22 แสดงการเปิดโปรแกรม Arduino IDE

3.3.2. เลือกเมนู File > Preferences ดังรูป

รูปที่ 11.23 แสดงการเลือกเมนู File > Preferences

เมนูต่างๆที่สำคัญ มีดังนี้

- Sketchbook - ที่อยู่ค่าเริ่มต้นของโฟลเดอร์โปรเจค
- Editor font size - กรณีที่ได้มีตัวหนังสือที่เล็กมากๆ เราสามารถปรับให้ค่าเพิ่มมากขึ้นได้ เพื่อให้ตัวอักษรตัวใหญ่และอ่านง่ายขึ้น
- Show verbose output during - ใช้ในกรณีที่จะให้โปรแกรมโชว์ว่ามันทำอะไรอยู่
พื้นหลัง เช่น ไปเรียกไฟล์โปรแกรมคอมไพล์ที่ไฟล์ไหน พาทไหน
- Display line number - แสดงหมายเลขบรรทัด

- Check for updates on startup - ตรวจสอบว่ามีอัปเดตใหม่หรือไม่ เมื่อเปิดโปรแกรม
- Save when verifying or uploading - บันทึกโปรเจกต์ทันทีที่ verifying หรือ uploading

ตัวอย่างการเลือกเมนู

รูปที่ 11.24 แสดงผลการเลือกเมนู เมนู File > Preferences และตัวอย่างการเลือกการตั้งค่า

4. โครงสร้างภาษา C Arduino เบื้องต้น

โครงสร้างโปรแกรมภาษา C บน Arduino IDE จะมีลักษณะแบบเดียวกับ C ทั่วไป ดังนี้

1. ปรีโพรเซสเซอร์ไดเรกทีฟ (Preprocessor directives)
2. ส่วนของการกำหนดค่า (Global declarations)
3. ฟังก์ชัน setup() และ ฟังก์ชัน loop()
4. การสร้างฟังก์ชัน และการใช้งานฟังก์ชัน (Users-defined function)
5. ส่วนอธิบายโปรแกรม (Program comments)

4.1. ปรีโพรเซสเซอร์ไดเรกทีฟ (Preprocessor directives)

โดยปกติแล้วเกือบทุกโปรแกรมต้องมี โดยส่วนนี้จะเป็นส่วนที่คอมไพเลอร์จะมีการ

ประมวลผลและทำตามคำสั่งก่อนที่จะมีการคอมไพล์โปรแกรม ซึ่งจะเริ่มต้นด้วยเครื่องหมายไดเรกทีฟ (directive) หรือเครื่องหมายสี่เหลี่ยม # แล้วจึงตามด้วยชื่อคำสั่งที่ต้องการเรียกใช้ หรือกำหนดโดยปกติแล้วส่วนนี้จะอยู่ในส่วนบนสุด หรือส่วนหัวของโปรแกรม และต้องอยู่นอกฟังก์ชันหลักใดๆก็ตาม

4.1.1. #include

#include เป็นคำสั่งที่ใช้อ้างอิงไฟล์ภายนอก เพื่อเรียกใช้ฟังก์ชันหรือตัวแปรที่มีการสร้างหรือกำหนดไว้ในไฟล์นั้น

รูปแบบการใช้งานคือ

```
#include <ชื่อไฟล์.h>
```

ตัวอย่าง เช่น

```
#include <Wire.h>
```

```
#include <Time.h>
```

จากตัวอย่าง มีการอ้างอิงไฟล์ Wire.h และไฟล์ Time.h ซึ่งเป็นไลบรารีพื้นฐานที่มีอยู่ใน Arduino ทำให้สามารถใช้ฟังก์ชันเกี่ยวกับเวลาที่ไลบรารี Time มีการสร้างไว้ให้ใช้งานได้ การอ้างอิงไฟล์จากภายในหรือการอ้างอิงไฟล์ไลบรารีที่มีอยู่แล้วใน Arduino หรือเป็นไลบรารีที่เราเพิ่มไปจะใช้เครื่องหมาย <> ในการครอบชื่อไฟล์ไว้ เพื่อให้โปรแกรมคอมไพเลอร์เข้าใจว่าควรไปหาไฟล์เหล่านี้จากในโฟลเดอร์ไลบรารี แต่หากอ้างอิงไฟล์ที่อยู่ในโฟลเดอร์โปรเจกต์ จะต้องใช้เครื่องหมาย "" ครอบแทน ซึ่งคอมไพเลอร์จะไปหาไฟล์นี้โดยอ้างอิงจากไฟล์โปรแกรมที่คอมไพเลอร์อยู่

เช่น `#include "myFunction.h"`

จากตัวอย่างด้านบน คอมไพเลอร์จะไปหาไฟล์ myFunction.h ภายในโฟลเดอร์โปรเจกต์หรืองานทันที หากไม่พบก็จะแจ้งเป็นข้อผิดพลาดออกมา

4.1.2. #define

#define เป็นคำสั่งที่ใช้ในการแทนข้อความที่กำหนดไว้ ด้วยข้อความที่ที่กำหนดไว้ ซึ่งการใช้คำสั่งนี้ ข้อดีคือไม่มีการอ้างอิงกับตัวโปรแกรมเลย

รูปแบบ

```
#define NAME VALUE
```

ตัวอย่างเช่น

```
#define LEDPIN 13
```

จากตัวอย่าง LEDPIN จะอยู่ส่วนใดของโค้ดโปรแกรมก็ตาม คอมไพเลอร์จะแทนค่าว่า LEDPIN ด้วยเลข 13 แทน ซึ่งข้อดีคือไม่ต้องสร้างเป็นตัวแปรขึ้นมาเพื่อเปลืองพื้นที่แรม และยังช่วยให้โปรแกรมทำงานเร็วขึ้นอีกด้วยเพราะซีพียูไม่ต้องไปขอข้อมูลมาจากแรมหลายๆทอด

4.2. ส่วนของการก าหนดค่า (Global declarations)

ส่วนนี้จะเป็นส่วนที่ใช้ในการก าหนดชนิดตัวแปรแบบนอกฟังก์ชัน หรือประกาศฟังก์ชัน เพื่อให้ฟังก์ชันที่ประกาศสามารถก าหนด หรือเรียกใช้ได้จากทุกส่วนของโปรแกรม เช่น `int pin = 13;`

```
void blink(void) ;
```

4.3. ฟังก์ชัน `setup()` และฟังก์ชัน `loop()`

ฟังก์ชัน `setup()` และฟังก์ชัน `loop()` เป็นค าสั่งที่ถูกบังคับให้ต้องมีในทุกโปรแกรม โดยฟังก์ชัน `setup()` จะเป็นฟังก์ชันแรกที่ถูกเรียกใช้ นิยมใช้ก าหนดค่าหรือเริ่มต้นใช้งานไลบรารีต่างๆ เช่น ในฟังก์ชัน `setup()` จะมีค าสั่ง `pinMode()` เพื่อก าหนดให้ขาใดๆก็ตามเป็นดิจิตอลอินพุตหรือเอาต์พุต ส่วนฟังก์ชัน `loop()` จะเป็นฟังก์ชันที่ทำงานหลังจากฟังก์ชัน `setup()` ได้ทำงานเสร็จสิ้นไปแล้ว และมีการวนรอบแบบไม่รู้จบ เมื่อฟังก์ชัน `loop()` งานครบตามค าสั่งแล้ว ฟังก์ชัน `loop()` ก็จะถูกเรียกขึ้นมาใช้อีก

ตัวอย่าง

```
int pin = 13;

void setup() {
  pinMode(pin, OUTPUT);
}

void loop() {
  digitalWrite(pin, HIGH);

  delay(1000);
  digitalWrite(pin, LOW);
  delay(1000);
}
```

จากตัวอย่าง จะเห็นว่ามีก ารประกาศตัวแปรแบบนอกฟังก์ชัน ทำให้สามารถก าหนด หรือเรียกใช้จากในฟังก์ชันใดๆก็ตามได้ ในฟังก์ชัน `setup()` ได้มีการก าหนดให้ขาที่ 13 เป็นดิจิตอล เอาต์พุต และในฟังก์ชัน `loop()` มีการก าหนดให้พอร์ต 13 มีลอจิกเป็น 1 และใช้ฟังก์ชัน `delay()` ในการหน่วงเวลา 1 วินาที แล้วจึงก าหนดให้พอร์ต 13 มีสถานะลอจิกเป็น 0 แล้วจึงหน่วงเวลา 1 วินาที จบฟังก์ชัน `loop()` และจะเริ่มทำฟังก์ชัน `loop()` ใหม่ ผลที่ได้คือไฟกระพริบบนบอร์ด Arduino Uno ในพอร์ตที่ 13 ทำงานแบบไม่รู้จบ

ในทุกๆการทำงานของฟังก์ชัน จะต้องเริ่มด้วยการก าหนดค่าที่ส่งกลับ ตามด้วยชื่อฟังก์ชัน แล้วตามด้วยเครื่องหมายปีกกาเปิด { และจบด้วยเครื่องหมายปีกกาปิด } ภายในฟังก์ชัน หากจะเรียก

ฟังก์ชันใช้งานย่อยใดๆ จะต้องมีเครื่องหมายเซมิโคลอน ; ต่อท้ายเสมอ

4.4. การสร้างฟังก์ชัน และการใช้งานฟังก์ชัน (Users-defined function)

ในการสร้างฟังก์ชันขึ้นมา คาสั่งต่างๆที่อยู่ภายในฟังก์ชัน ต้องอยู่ภายใต้เครื่องหมายปีกกาเปิด { และ ปิด } เท่านั้น ภายใต้เครื่องหมาย {} เราสามารถนำฟังก์ชันหรือคาสั่งใดๆก็ได้มาใส่ได้ แต่จะต้อง คั่นแต่ละคาสั่งด้วยเครื่องหมายเซมิโคลอน ; โดยจะนำคาสั่งทั้งหมดไว้บรรทัดเดียวกันเลยหรือแยก บรรทัดกันก็ได้เพื่อความสวยงามของโค้ด (ไม่มีผลกับขนาดของโปรแกรมหลังคอมไพล์) ตัวอย่าง

```
void Mode(int pin) {  
    pinMode(pin, OUTPUT);  
}  
  
void setup() {  
    Mode(13);  
}
```

4.5. ส่วนอธิบายโปรแกรม (Program comments)

ส่วนอธิบายโปรแกรม หรือการคอมเมนต์โปรแกรม การคอมเมนต์โค้ดมีอยู่ 2 รูปแบบ คือ เปิดด้วย /* และปิดด้วย */ เป็นการคอมเมนต์โค้ดแบบข้ามบรรทัด คือตราบใดที่ยังไม่มี */ตรงส่วนนั้นจะเป็นคอมเมนต์ทั้งหมด เช่น

```
/*
```

```
This code by IOXhop.com
```

```
17/5/2558
```

```
*/
```

```
void setup() { .... }
```

และแบบที่ 2 เป็นการคอมเมนต์บรรทัดเดียว คือเปิดด้วยเครื่องหมาย //

และปิดด้วยการขึ้น บรรทัดใหม่ เช่น

```
void setup() {  
    pinMode(13, OUTPUT); // Set pin 13 to output  
}
```

ที่มา : <https://www.ioxhop.com/article?tskp=10>