**บทที่ 2**

**ทฤษฎี และงานวิจัยที่เกี่ยวข้อง**

**2.1 RFID ( Radio Frequency Identification )**

เป็นระบบระบุลักษณะของวัตถุ ด้วยคลื่นความถี่วิทยุที่ได้ถูกพัฒนามาตั้งแต่ปี ค.ศ. 1980 มีวัตถุประสงค์หลักเพื่อนำไปใช้งานแทนระบบบาร์โค้ด (Barcode) โดยจุดเด่นของ RFID อยู่ที่การอานข้อมูลจากแท็ก (Tag) ได้หลาย ๆ แท็กแบบไร้สัมผัสและสามารถอ่านค่าได้ แม้ในสภาพที่ทัศนวิสัยไม่ดีทนตอความเปียกชื้น แรงสั่นสะเทือน การกระทบกระแทก สามารถอ่านข้อมูลได้ ด้วยความเร็วสูง โดยข้อมูลจะถูกเก็บไว้ในไมโครชิปที่อยู่ในแท็กในปัจจุบันได้มีการนำ RFID ไปประยุกต์ใช้งานในด้านอื่น ๆ นอกเหนือจากนำมาใช้ แทนระบบบาร์โค้ดแบบเดิม เช่น ใช้ในบัตรชนิดต่าง ๆ เช่น บัตรสำหรับใช้ผ่านเข้าออกสถานที่ต่าง ๆ บัตรที่จอดรถ ตามศูนย์การค้าต่าง ๆ ที่เราอาจพบเห็นอยู่ในรูปของแท็กสินค้ามีขนาดเล็กจนสามารถแทรกลงระหว่างชั้นของเนื้อกระดาษได้หรือเป็นแคปซูลขนาดเล็กฝังเอาไว้ในตัวสัตว์เพื่อบันทึกกประวัติต่าง ๆ เป็นต้น ในระบบ RFID จะมีองค์ ประกอบหลัก ๆ อยู่ 2 ส่วนด้วยกัน ส่วนแรกคือ ทรานสปอนเดอร์ หรือแทก (Transponder/Tag) ที่ใช้ติดกับวัตถุต่าง ๆ ที่เราต้องการโดยแท็ก ที่ว่าจะบันทึกข้อมูลเกี่ยวกับ วัตถุชิ้นนั้น ๆเอาไว้ ส่วนที่สองคือเครื่องสำหรับ อ่าน/เขียนข้อมูลภายในแทก (Interrogator/Reader) ด้วยคลื่นความถี่วิทยุ

**2.1.1 แท็ก ( Tag )**

โครงสร้างภายในแท็ก ประกอบไปด้วย 2 ส่วนใหญ่ ๆ ได้แก่ขดลวดขนาดเล็ก ซึ่งทำหน้าที่เป็นสายอากาศ สำหรับส่งสัญญาณคลื่นความถี่วิทยุและสร้างพลังงานป้อนในส่วนของไมโครชิพ ที่ทำหน้าที่เก็บข้อมูลของวัตถุเช่น รหัสสินค้าโดยทั่วไป Tag อาจอยู่ในชนิดทั้งเป็นกระดาษ แผ่นฟิล์ม แผ่นพลาสติก มีขนาดและรูปร่างต่าง ๆ ทั้งนี้ขึ้นอยู่กับวัสดุที่จะนำเอาไปติดและมีหลายรูปแบบเช่นขนาด เท่าบัตรเครดิต เหรียญกระดุม ฉลากสินค้า แคปซูล โดยสามารถแบ่งออกเป็น 2 ชนิดใหญ่ ๆ ได้แก่

**2.1.1.1 Passive RFID Tag**

แท็กชนิดนี้ไม่ต้องอาศัยแหล่งจ่ายไฟภายนอกใด ๆเพราะภายในแถบจะมีวงจรกำเนิดไฟฟ้าขนาดเล็ก เป็นแหล่งจ่ายไฟในตัว ทำให้การอ่านข้อมูลทำได้ไม่ไกลมากนักระยะอ่านสูงสุดอยู่ประมาณ 1 เมตร ขึ้นอยู่กับความแรงของเครื่องส่ง และคลื่นความถี่วิทยุที่ใช้ปกติแท็ก ชนิดนี้มักมีหน่วยความจำขนาดเล็กโดยทั่วไปมีขนาดประมาณ 16-24 ไบต์ มีขนาดเล็กและน้ำหนักเบา

**2.1.1.2 Active RFID Tag**

แท็กชนิดนี้ต้องอาศัยแหล่งจ่ายไฟจากแบตเตอรี่ภายนอก เพื่อจ่ายพลังงานให้กับวงจรภายใน แท็กชนิดนี้มีหน่วยความจำขนาดใหญ่ได้ถึง 1 เมกะไบต์ และสามารถอ่านได้ในระยะไกลสูงสุดประมาณ 10 เมตร แม้ว่าแท็กจะมีข้อดีอยู่หลายข้อแต่ก็มีข้อเสียด้วยเช่น มีราคาต่อหน่วยแพง มีขนาดค่อนข้างใหญ่ มีระยะเวลาในการทำงานที่จำกัด

**2.1.1.3 RFID Tag จากรูปแบบการอ่านเขียน**

แบ่งได้ 3 แบบ คือแบบที่สามารถอ่านและเขียนข้อมูลได้อย่างอิสระ Read write แบบเขียนได้เพียงครั้งเดียวเท่านั้นแต่อ่านได้อย่างอิสระ writhe once read many แบบที่ 3 แบบอ่านได้เพียงครั้งเดียว Read Only





**รูปที่ 2.1 RFID Tag**

**2.1.2 เครื่องอ่าน ( Reader )**

หน้าที่ของเครื่องอ่านคือ การเชื่อมต่อเพื่อเขียนหรืออ่านข้อมูลลงใน Tag ด้วยสัญญาณความถี่วิทยุภายในเครื่องอ่าน จะประกอบด้วยเสาอากาศที่ทำจากขดลวดทองแดง เพื่อใช้รับส่งสัญญาณวิทยุ และวงจรควบคุมการอ่านเขียน จำพวกไมโครคอนโทรลเลอร์และส่วนของการติดต่อกับคอมพิวเตอร์ โดยทั่วไปสารประกอบส่วนประกอบหลักดังนี้

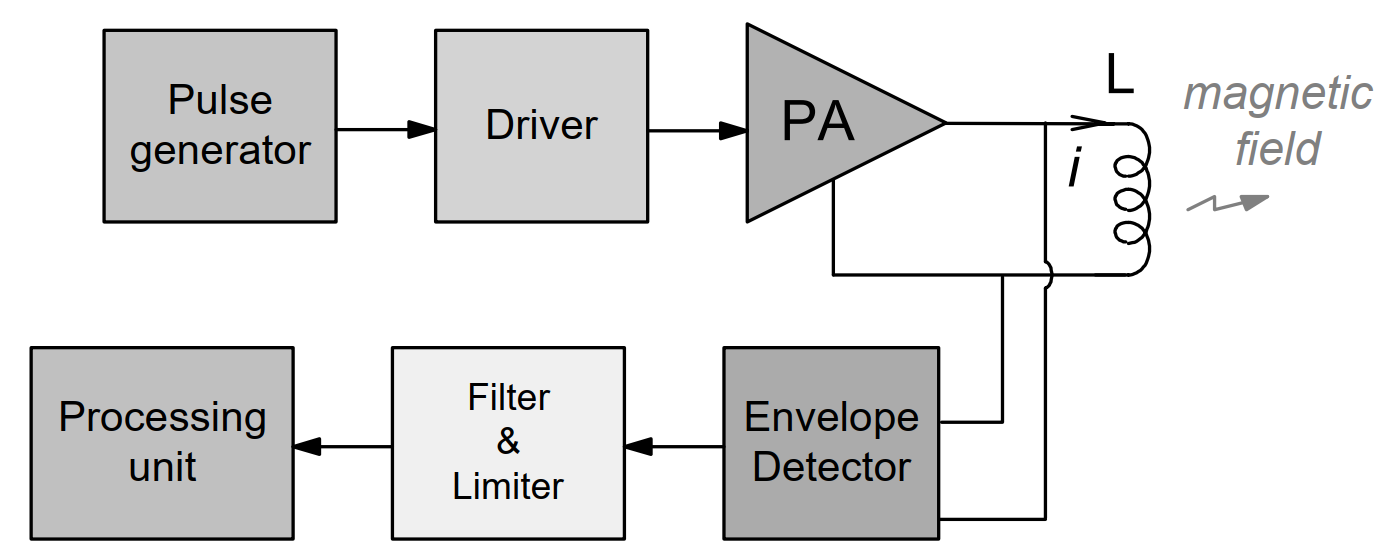
1) ภาครับส่งสัญญาณวิทยุ (Transceiver)

2) ภาคสร้างสัญญาณพาหะ (Carrier)

3) ขดลวดที่ทำหน้าที่เป็นเสาอากาศ (Antenna)

4) วงจรจูนสัญญาณ (Tuner)

5) หน่วยประมวลผลข้อมูลและภาพติดต่อกับคอมพิวเตอร์ (Processing Unit)



**รูปที่ 2.2 โครงสร้างภายในเครื่องอ่าน RFID**



**รูปที่ 2.3 RFID Reader**

**2.1.3 หลักการและเทคนิคที่ใช้รับและส่งข้อมูลระหว่าง แท็กและเครื่องอ่าน**

โดยมากเทคนิคในการรับส่งข้อมูลระหว่างเครื่องอ่านและ แท็ก จะใช้หลักการมอดูเลตทางแอมพลิจูด (Amplitude Modulation : AM) หรือใช้การมอดูเลตทางแอมพลิจูดบวกกับการเข้ารหัส แมนเชสเตอร์ ( Manchester encoded AM ) แต่ถว่าในปัจจุบันก็มีแขกที่ใช้การมอดูเลตแบบอื่นเช่นการมอดูเลชั่นแบบ เฟสชิฟคีย์อิง (Phase Shift keying : PSK ) ฟรีเควนซี่ชิฟฟคีย์อิง

( Frequency Shift keying : FSK ) หรือการมอดูเลตทางความถี่ Frequency Modulation : FM ในการรับส่งข้อมูลหรือสัญญาณวิทยุระหว่างแท็กกับเครื่องอ่าน จะทำได้อย่างมีประสิทธิภาพต่อเมื่อ

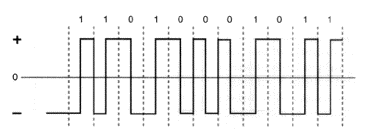
สายอากาศมีความยาวที่เหมาะสมกับความถี่พาหะที่ใช้งาน เช่นเมื่อความถี่ใช้งานเป็น 13.56MHz ความยาวของเสาอากาศ (เป็นเส้นตรง) ที่เหมาะสมก็คือ 22.12 เมตร แน่นอน ว่าในทางปฏิบัติเราคงไม่สามารถนำเสาอากาศที่ใหญ่ขนาดนั้นมาใช้งานกับแท็กขนาดเล็กได้ สายอากาศที่ดูเหมาะจะใช้ร่วมกับแท็กมากที่สุดก็คือสายอากาศที่เป็นขดลวดขนาดเล็กหรือที่มีชื่ออย่างเป็นทางการว่าสายอากาศแบบแมกเนติกไดโพล (magnetic dipole antenna) รูปแบบของ สายอากาศแบบนี้ก จะมีอยู่หลากหลาย ทั้งแบบที่เป็นขดลวดพันบนแกนอากาศ หรือแกนเฟอร์ไรต์ แบบที่เป็นวงลูป ทำขึ้นนจากลายทองแดง บนแผนวงจรพิมพ์ ทั้งที่เป็นลูปแบบวงกลมและสี่เหลี่ยม ทั้งนี้ความเหมาะสมในการใช้งานก็แตกต่างกันทไปตามความถี่พาหะและประเภทของงานด้วยเช่นกัน

นอกจากการรับส่งข้อมูลแล้วสาย อากาศก็ยังทำหน้าที่เป็นแหล่งจ่ายพลังงานให้กับแท็กด้วย โดยอาศัยหลักการทำงาน ตามแนวคิดของไมเคิล ฟาราเดย์ เรื่องแรงดันเหนี่ยวนำในขดลวดที่เกิดขึ้นนจากเส้นแรงแม่เหล็ก (จากเครื่องอ่าน) ที่มีค่าเปลี่ยนแปลงไปตามเวลา (Time-varying magnetic field) พุ่งผ่านสายอากาศของแท็ก เมื่อแท็กและเครื่องอ่านตั้งอยู่ห่างกันในระยะ 0.16 เท่าของความยาวคลื่นพาหะ ที่ใช้เรียกปรากฏการณ์ที่เกิดขึ้นนี้ว่า transformer-type coupling ซึ่งเป็นปรากฏ-การณ์แบบเดี่ยวกับการเกิดแรงดันไฟฟ้า เหนี่ยวนำขึ้นระหว่างขดลวดปฐมภูมิ ( primary ) และขดลวดทิตย์ภูมิ (secondary) ในหม้อแปลงไฟฟ้า (transformer) จะเป็นวงจรพื้นฐานสำหรับอธิบายกลไกที่

เกิดขึ้นในการส่งข้อมูลของแท็ก

**2.1.4 การเข้ารหัสแมนเชสเตอร์**

เป็นการเข้ารหัสข้อมูลดิจิทัลวิธี หนึ่งก่อนที่ข้อมูลซึ่งผ่านการเข้ารหัสแล้วจะถูกส่งไปมอดูเลต เพื่อแก้ปัญหาเกี่ยวกับ การซิงโครไนซ์ของข้อมูล เนื่องจากการส่งกระจายสัญญาณตามปกตินั้นหากมีการส่งสัญญาณดิจิทัลในระดับเดียวติดต่อกนเป็น ช่วงยาว เช่น ส่งสัญญาณดิจิทัลที่มีค่าลอจิกเป็น 1 ออกไป 20 บิตติดต่อกน จะทำให้การซิงโครไนซ์ของข้อมูลเกิดการคลาดเคลื่อน (โดยปกติวงจรดิจิทัลจะปรับการซิงโครไนซ์ของข้อมูล ได้เฉพาะในช่วงที่มีการเปลี่ยนระดับของข้อมูลจาก 1 เป็น 0 หรือ จาก 0 เป็น 1) และทำให้รับข้อมูลผิดพลาดเพื่อป้องกันปัญหา ดังกล่าวจึงจะต้องมีการนำสัญญาณดิจิทัลปกติไปผ่านเข้ารหัสเสียก่อน โดยการเข้ารหัสแบบแมนเชสเตอร์ จะเปลี่ยน ให้สัญญาณดิจิทัล ลอจิก 0 ถูกแทนด้วยการเปลี่ยนค่าจากลอจิก 1 เป็น 0 และสัญญาณดิจิทัล ลอจิก 1 แทนด้วยการเปลี่ยนค่าจากลอจิก 0 เป็น 1 ข้อดี ของการเข้ารหัสแบบนีก็คือทำให้ การเปลี่ยนระดับของข้อมูลทุก ๆ ครั้งเป็นไปอย่างแน่นอน หรือ เกิดการเข้าจังหวะ (synchronize) กันของข้อมูลนั้นเอง แต่ว่า การเข้ารหัสแบบนี้ก็มีข้อเสียอยู่กล่าวคือช่วงความถี่ที่ใชในการส่งข้อมูล ต้องเพิ่มขึ้นเป็น 2 เท่า ดังรูปที่ 2.4



**รูปที่ 2.4 สัญญาณรูปคลื่นที่เข้ารหัสแบบแมนเชสเตอร์**

**2.1.5 ขั้นตอนการทำงานของเครื่องอ่านกับแท็ก**

1. เครื่องอ่านจะทำการส่งสัญญาณวิทยุอย่างต่อเนื่อง หรือเป็นจังหวะและรอคอยสัญญาณตอบจากแท็ก

2. เมื่อแท็กได้รับสัญญาณคลื่นวิทยุที่ส่งมาจากเครื่อง อ่านในระดับที่เพียงพอ จะทำเหนี่ยวนำเพื่อสร้างพลังงานป้อนให้แท็กทำงาน โดยแท็กจะสรางสัญญาณนาฬิกาเพื่อ กระตุ้นให้วงจรภาคดิจิทัลในแท็กทำงาน

3. วงจรภาคดิจิทัลจะไปอ่านข้อมูลจากหน่วยความจำ ภายในและเข้ารหัสข้อมูลแล้วส่งไปยังภาคอนาล็อก ทีทำหน้าที่มอดูเลตข้อมูล

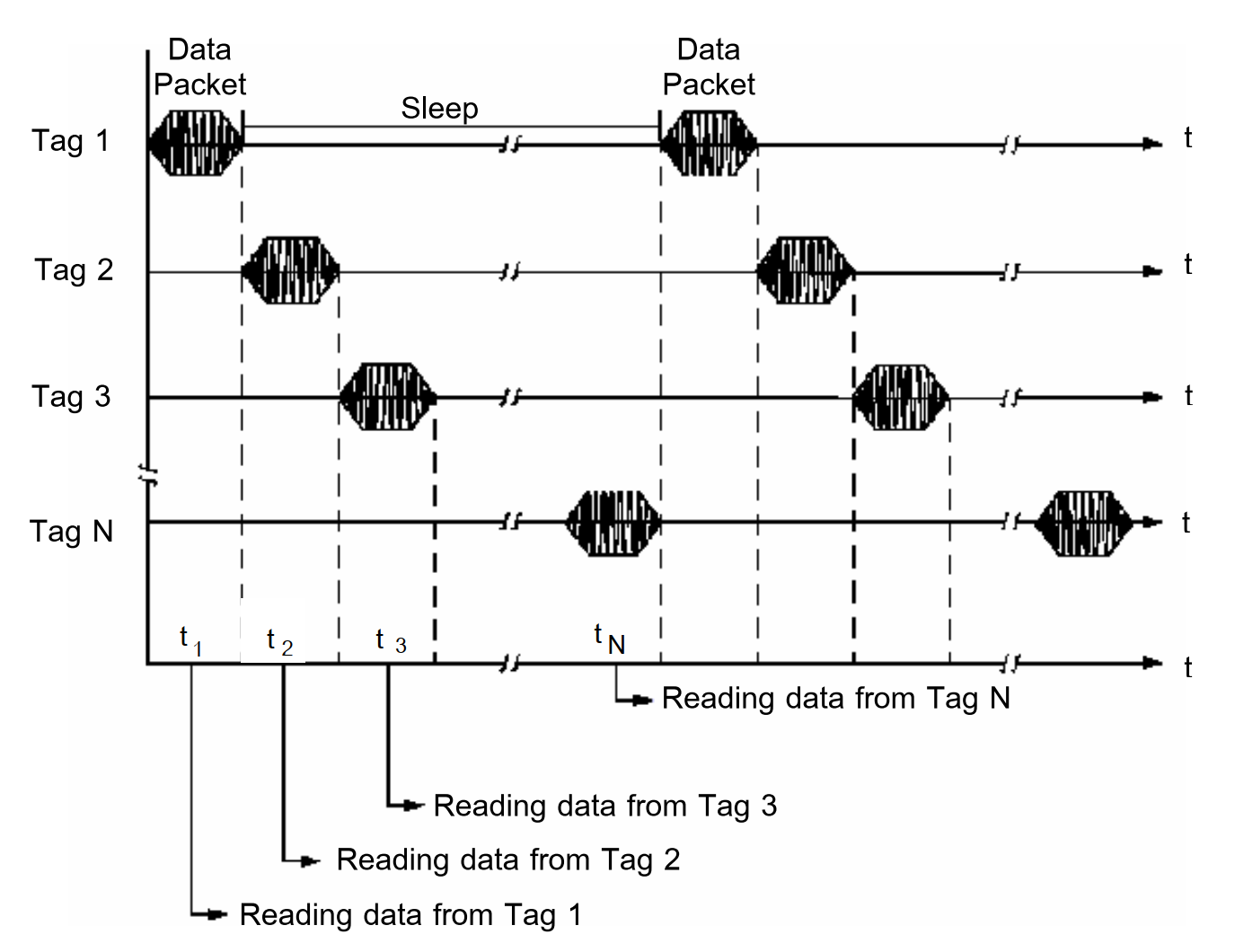
4. ข้อมูลที่ถูกมอดูเลตจะถูกส่งไปขดลวดที่ทำหน้าที่ เป็นสายอากาศ เพื่อสงไปยังเครื่องอ่าน

5.เครื่องอ่านจะสามารถตรวจจับสัญญาณการเปลี่ยน แปลงของแอมพลิจูด(Envelope Detector) และใช้ พีก ดีเทกเตอร์ (Peak Detector) ในการแปลงสัญญาณข้อมูลที่ มอดูเลตแล้วจากแท็ก

6. เครื่องอ่านจะถอดรหัสข้อมูล และส่งไปยังคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมต่อไป

**2.1.6 การชนกันของข้อมูลคืออะไร**

เมื่อมีป้ายหลาย ๆ อันเข้ามาอยู่ใกล้เครื่องอ่าน เมื่อป้ายมีพลังงานเพียงพอ ป้ายแต่ละอันจะพยายามส่งข้อมูลของตัวเองมาที่เครื่องอ่านพร้อม ๆ กัน ทำให้เครื่องอ่านไม่สามารถแยกแยะข้อมูลที่ส่งมาได้ ซึ่งเราเรียกปรากฏการณ์นี้ว่า การชนกันของข้อมูล (Collision) วิธีการแก้ไขโดยการทำการเพิ่มฟังก์ชั่นป้องกันการชนกันบนป้ายและเครื่องอ่าน (Anti-collision) ซึ่งจะมีหลายเทคนิค เช่น จัดคิวการอ่านป้ายโดย ทำเป็นช่วงเวลาสั้น ๆ เมื่อป้ายโดนอ่านแล้วจะไม่มีการอ่านซ้ำอีกเช่น เทคนิค SDMA : Space Division Multiple Access TDMA, FDMA, CDMA หรือเทคนิคขั้นสูงจะใช้ FTDMA และการกระโดดความถี่ (frequency hopping) เข้าช่วย



**รูปที่ 2.5 Anti Collision**

**2.1.6 การป้องกันการชินกันของสัญญาณข้อมูล ( Anti Collision)**

การอ่านข้อมูลจากแท็กได้หลาย ๆ แท็กในเวลาเดี่ยวกันเป็นข้อดีข้อหนึ่งของ RFID จะทำให้การอ่านข้อมูลของแท็กจำนวนมากทำได้อย่างรวดเร็ว ซึ่งสิ่งที่ทำให้การอ่านข้อมูล จากแท็กได้พร้อม ๆ กัน นั้นก็คืออัลกอริทึมที่ใช้ในการป้องกัน การชนของข้อมูล (Anti-Collision) ที่อยู่ภายในระบบ RFID ดังรูปที่ 2.5 จากรูปที่ 2.5 แสดงอัลกอริทึมที่ใช้ป้องกันการชนข้อมูลของแท็กบางชนิด โดยหลักการของการอ่านข้อมูลจากแท็กจะอ่านเป็นลำดับในเวลาที่กำหนด แต่ละแท็กจะไม่ส่ง ข้อมูลไปยังเครื่องอ่านทันที่จะมีการจัดสรรลำดับเวลา (Time Slot) ในการส่งข้อมูลที่เวลาต่างๆ กัน ตามอัลกอริทึมที่ กำหนดทำให้ข้อมูลที่เครื่องอ่านรับได้ไม่มีการชนของข้อมูลที่ส่งมาจากแท็กหลายแท็กพร้อมกัน

**2.1.7 คลื่นความถี่ที่ใช้ในระบบ RFID**

ในปัจจุบันคลื่นที่ใช้งานกันในระบบ RFID จะอยู่ ในย่านความถี่ ISM (Industrial-Scientific-Medical) ซึ่งเป็นย่านความถี่ ที่กำหนดการใช่งานในเชิงอุตสาหกรรม วิทยาศาสตร์ และการแพทย์ สามารถใช้งานได้โดยไม่ตรงกับย่านความถี่ ที่ใชเงานในการสื่อสารทั่วไป สำหรับคลื่นที่ใช้กันในระบบ RFID อาจแบงออกได้เป็น 4 ย่านความถี่ ใช้งานหลัก ได้แก่

1. ย่านความถี่ต่ำ (Low Frequency : LH )ต่ำกว่า 150 kHz

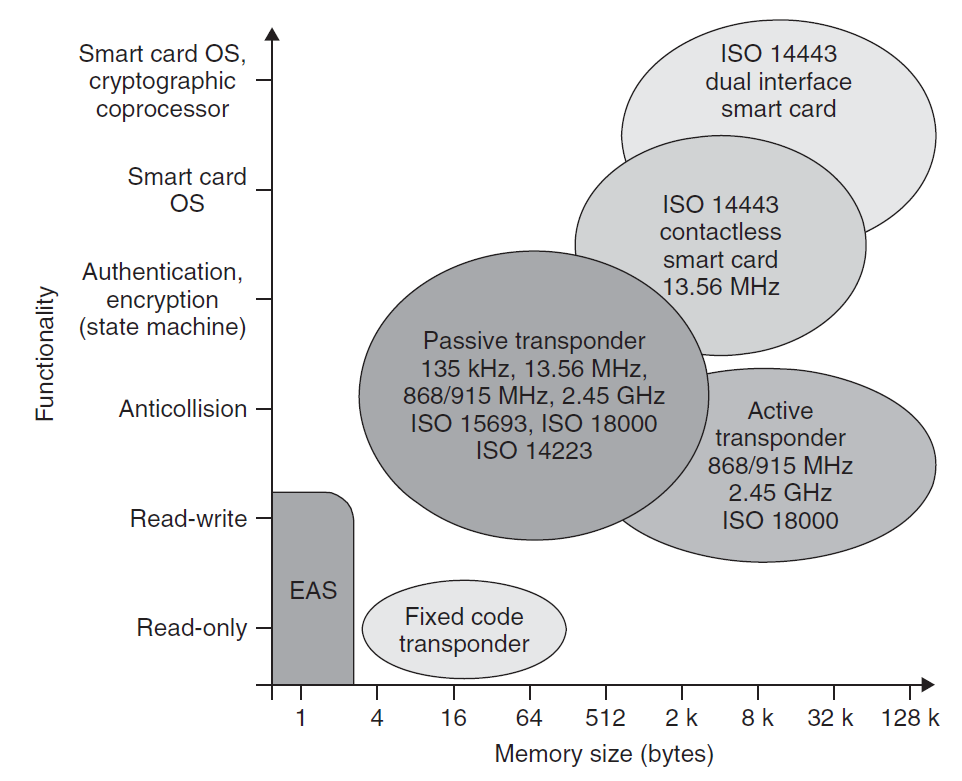
2. ย่านความถี่สูง (High Frequency: HF) 13.56 MHz

3. ย่านความถี่สูงมาก (Ultra High Frequency: UHF) 433/868/915 MHz

4. ย่านความถี่ไมโครเวฟ (Microwave Frequency) 2.4-5.8 GHz

การใช้งาน 2 ย่านความถี่แรกจะเหมาะสำหรับใช้กับ งานที่มีระยะการสื่อสารข้อมูลในระยะใกล้ (LH ระยะอ่านประมาณ 10 - 20 เซนตเมตริ และ HF ระยะอานประมาณ 1 เมตร) เช่น การตรวจสอบการผ่านเข้าออกพื้นที่การตรวจหา และเก็บประวัติในสัตว์ ส่วนย่านความถี่สูงมาก จะถูกใช้กับงานที่มีระยะการสื่อสารข้อมูลในระยะไกลู(UHF ระยะอานประมาณ 1-10 เมตร) เช่น ระบบเก็บคาบริการทางด่วน และในปัจจุบันระบบ RFID กำลังถูกวิจัยและพัฒนาในย่านความถี่ไมโครเวฟ ที่ความถี่ 2.4 GHz และความถี่ 5.8 GHz เพื่อใช้งานที่ต้องการระยะอ่านที่ไกลกวา 10 เมตร

**2.1.8 มาตรฐาน RFID**



**รูปที่ 2.6 มาตรฐาน RFID**

**2.1.8.1 1 Bit Transponder (EAS)**

ป้ายอาร์เอฟไอดี จะส่งเพียงสองสถานะไปยังเครื่องอ่าน คือ 1 หรือ 0 เพื่อแสดงว่ามีแท็ก อยู่ในบริเวณเครื่องอ่านหรือไม่ เนื่องจากมีการส่งเพียงสองสถานะ จึงเรียกวว่า

1 bit Transponder หรืออีกชื่อหนึ่งคือ (EAS : Electronic Article Surveillance ส่วนมาก จะใช้ระบบ EAS นี้ตามศูนย์การค้า โดยติดเครื่องอ่านไว้ที่ทางเข้า-ออก เพื่อป้องกันการขโมยสินค้า เนื่องจากป้าย อาร์เอฟไอดี ของระบบ EAS มีขนาดเล็กสามารถนำไปติดตั้งกับสินค้าได้ง่าย

**2.1.8.2 มาตรฐาน RFID ในป้ายประเภทป้าย Passive**

เป็นมาตรฐานสำหรับการระบุรหัสประจำตัวสัตว์ ทำงานที่ความถี่ 125-134 KHz หรือทั่วไปจะเรียกบัตร 125 KHz ระยะรับส่งข้อมูลสูงสุดไม่เกิน 1.2 เมตร อ่านข้อมูลได้อย่างเดียว ไม่สามารถเขียนข้อมูลลงไปที่ป้าย RFID ได้ ประกอบด้วยมาตรฐานต่าง ๆ ดังนี้

1. ISO11784 มาตรฐานโครงสร้างของข้อมูลบนป้าย RFID ประกอบด้วยข้อมูลตัวเลขขนาด 64 บิต โดยมีการระบุถึงประเทศ, ชนิดของสัตว์ และรหัสของสัตว์

2. ISO11785 เป็นมาตรฐานที่ใช้ระบุการทำงาน, วิธีการส่งข้อมูล ตลอดจนเทคนิคต่าง ๆ ในการติดต่อระหว่างป้ายกับเครื่องอ่าน ซึ่งได้กำหนดวิธีการส่งไว้ 2 ลักษณะคือ Full-Duplex และ Half-Duplex จะมีข้อแตกต่างกันดังตารางที่ 2.1

**ตารางที่ 2.1 ตารางเปรียบเทียบ**

|  |  |  |
| --- | --- | --- |
| **Protocol** | **Full Duplex** | **Half Duplex** |
| **Modulation** | **ASK** | **FSK** |
| **Frequency** | **129-133.2 KHz** | **124.2 KHz = 1** |
| **135.2-139.4 KHz** | **134.2 KHz = 2** |
| **Channel code** | **Differential Biphase (DBP)** | **None** |
| **Symbol time** | **0.23845 ms** | **0.1288 ms 1** |
|  | **0.1192 ms 0** |
| **Telegram (bit)** | **128** | **112** |

**2.1.8.3 ISO10536**

เป็นมาตรฐานสำหรับ Identification Cards ชนิดหนึ่ง หรือเราเรียกโดยทั่วไปว่าบัตร 125 KHz ในแต่ละบัตรจะมีเลข ID เฉพาะของแต่ละบัตรไม่ซ้ำกัน ไม่สามารถเขียนข้อมูลลงไปในบัตรได้ ระยะการรับส่งไม่เกิน 150 mm เหมาะกับงานด้านความปลอดภัย การเข้า/ออกบ้านหรือสำนักงาน

**2.1.8.4 มาตรฐาน ISO14443 Type A (MIFARE)**

เป็นสิทธิบัตรของบริษัท NXP ใช้คลื่นความถี่ 13.56 MHz ระยะรับส่งข้อมูลประมาณ 10 cm ข้อดีของมาตรฐาน MIFARE คือ

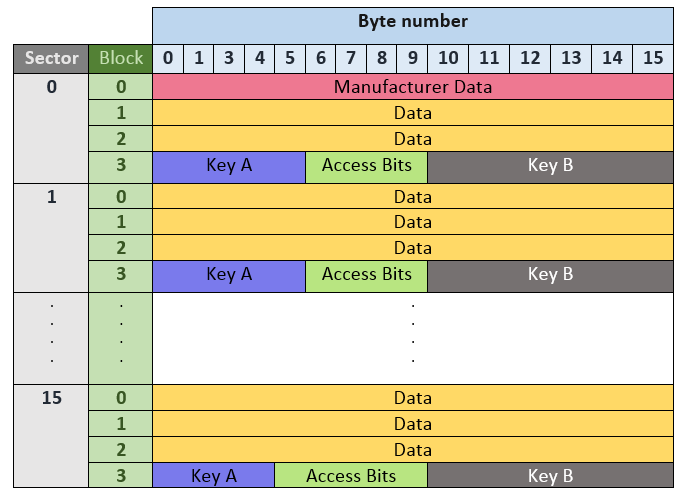
1. มีการเข้ารหัสในการเข้าถึงข้อมูลบนบัตร

2. บัตร MIFARE ไม่ต้องสอดเข้าไปที่เครื่องอ่านบัตร เมื่อบัตรอยู่ในระยะของเครื่องอ่าน เครื่องก็สามารถส่งคำสั่งไปที่บัตรได้

3. แต่ละบัตรมีเลข Serial number ที่ไม่ซ้ำกัน

ข้อเสียของมาตรฐาน MIFARE

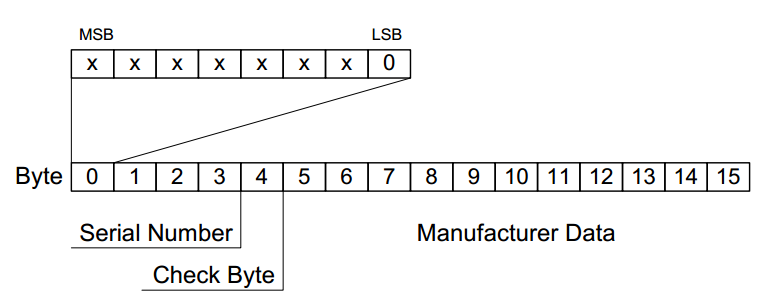
1. มีราคาสูง



**รูปที่ 2.7 โครงสร้างข้อมูลของบัตร MIFARE**

จากรู)ที่ 2.7 เป็นโครงสร้างข้อมูลของบัตร MIFARE ขนาด 1 Kbyte บัตรแบ่งข้อมูลออกเป็นเซ็คเตอร์ (Sector) ทั้งหมด 16 เซ็คเตอร์ (ตั้งแต่ 0 ถึง 15) ในแต่ละเซ็คเตอร์ประกอบด้วยชุดข้อมูล 4 บล็อค (Block) โดยบล็อกต่าง ๆ เก็บข้อมูลบนบัตรดังนี้

Sector 0 Block 0 เก็บข้อมูลของบัตรหรือค่า UID (Unique Identifier) ขนาด 16 บิต สามารถอ่านได้เพียงอย่างเดียวไม่สามารถเขียนได้



**รูปที่ 2.8 Sector 0**

Block 0, 1, 2 ของทุก Sector (ยกเว้น Sector 0 ที่ใช้ได้เฉพาะ Block 1,2) ใช้เก็บข้อมูลที่ผู้ใช้ต้องการเขียน-อ่านบัตร โดยบัตร MIFARE ยังมีฟังก์ชั่น Value Block เขียนข้อมูลลงไปเฉพาะ 4 ไบต์แรก (ไบต์ที่ 0-3) และ Backup ข้อมูลไว้ที่ไบต์ที่ 8-11 Invert ข้อมูลไว้ในไบต์ 4-7 เก็บข้อมูลตำแหน่ง (Address) ของ Block ขนาด 1 ไบต์ไว้ที่ไบต์ที่ 12 กับ 14 และ Invert ข้อมูลตำแหน่งไว้ที่ 13 กับ 15 ฟังก์ชั่น Value Block เหมาะสำหรับระบบที่ต้องการความเร็วในการอ่านเขียนข้อมูล ไม่ต้องการใช้ข้อมูลทั้ง Block ต้องการเพียงเขียน อ่าน เพิ่ม หรือลด และมีระบบป้องกันข้อมูล

Block 3 ในแต่ละ Sector เก็บข้อมูลสิทธิ์การเข้าถึงข้อมูลใน Block นั้น ประกอบด้วย

Key A เก็บรหัสการเข้าถึงข้อมูลของบัตร ไว้ที่ Block 3 Byte ที่ 0 – 5

Key B เก็บรหัสการเข้าถึงข้อมูลของบัตร ไว้ที่ Block 3 Byte ที่ 11 – 15

Access Bits เป็นค่าที่ใช้กำหนดสิทธิ์ของ Key A และ Key B สามารถอ่าน/เขียน Block ไหนได้บ้าง ใน Sector นั้น เช่น ผู้ใช้บัตรอาจกำหนดให้ ตนเองถือ Key A อยู่ซึ่งกำหนดสิทธิ์ใน Access bits แล้วให้สามารถอ่านหรือเขียนข้อมูลได้ แต่ผู้ใช้ทั่วไปถือ Key B สามารถอ่านข้อมูลได้เพียงอย่างเดียว

**2.2 มอเตอร์ไฟฟ้ากระแสตรง ( DC Motor )**

มอเตอร์ไฟฟ้า คืออุปกรณ์ ที่ทำหน้าที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกล ในรูปของการหมุนเคลื่อนที่ โดยเมื่อจ่ายไฟให้กับมอเตอร์ จะทำให้แกนของมอเตอร์หมุน จึงสามารถนำการหมุนของแกนมอเตอร์ไปใช้ในการขับเคลื่อนวัตถุให้เกิดการเคลื่อนที่

**2.2.1 ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง**

**2.2.1.1 ขดลวดสนามแม่เหล็ก (Field Coil)**

คือขดลวดที่ถูกพันอยู่กับขั้วแม่เหล็กที่ยึดติดกับโครงมอเตอร์ ทำหน้าที่กำเนิดขั้วแม่เหล็กขั้วเหนือ (N) และขั้วใต้ (S) แทนแม่เหล็กถาวรขดลวดที่ใช้เป็นขดลวดอาบน้ายาฉนวน สนามแม่เหล็กจะเกิดขึ้นเมื่อจ่ายแรงดันไฟตรงให้มอเตอร์

**2.2.1.2 ขั้วแม่เหล็ก (Pole Pieces)**

คือแกนสำหรับรองรับขดลวดสนามแม่เหล็กถูกยึดติดกับโครงมอเตอร์ด้านใน ขั้วแม่เหล็กทำมาจากแผ่นเหล็กอ่อนบาง ๆ อัดซ้อนกัน (Lamination Sheet Steel) เพื่อลดการเกิดกระแสไหลวน (Edy Current) ที่จะทำให้ความเข้าของสนามแม่เหล็กลดลง ขั้วแม่เหล็กทำหน้าที่ให้กำเนิดขั้วสนามแม่เหล็กมีความเข้มสูงสุด แทนขั้วสนามแม่เหล็กถาวร ผิวด้านหน้าของขั้วแม่เหล็กทำให้โค้งรับกับอาร์เมเจอร์พอดี

**2.2.1.3 โครงมอเตอร์ (Motor Frame)**

คือส่วนเปลือกหุ้มภายนอกของมอเตอร์ และยึดส่วนอยู่กับที่(Stator) ของมอเตอร์ไว้ภายในร่วมกับฝาปิดหัวท้ายของมอเตอร์ โครงมอเตอร์ทำหน้าที่เป็นทางเดินของเส้นแรงแม่เหล็กระหว่างขั้วแม่เหล็กให้เกิดสนามแม่เหล็กครบวงจร

**2.2.1.4 อาร์เมเจอร์ (Armature)**

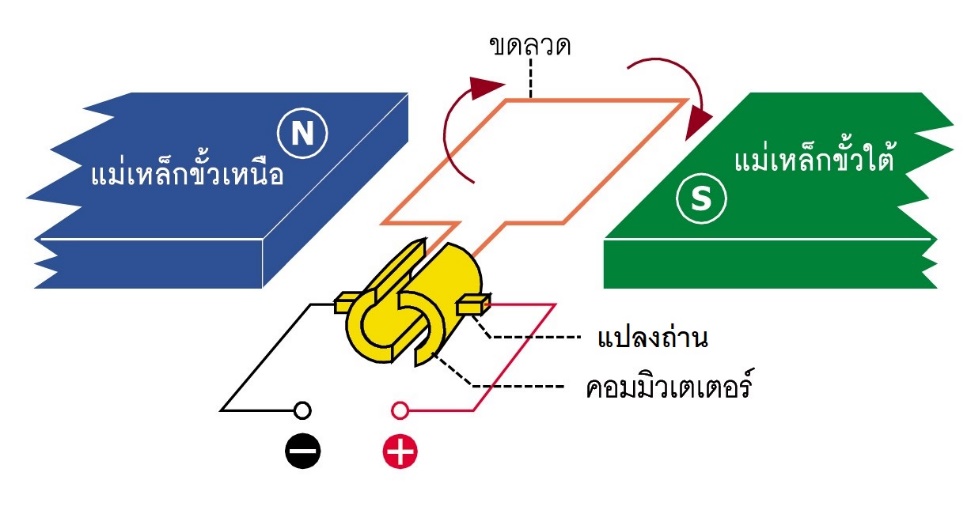
คือส่วนเคลื่อนที่(Rotor) ถูกยึดติดกับเพลา (Shaft) และรองรับการหมุนด้วยที่รองรับการหมุน (Bearing) ตัวอาร์เมเจอร์ทำจากเหล็กแผ่นบาง ๆ อัดซ้อนกัน ถูกเซาะร่องออกเป็นส่วน ๆ เพื่อไว้พันขดลวดอาร์เมเจอร์ (Armature Winding) ขดลวดอาร์เมเจอร์เป็นขดลวดอาบน้ายาฉนวน ร่องขดลวดอาร์เมเจอร์จะมีขดลวดพันอยู่และมีลิ่มไฟเบอร์อัดแน่นขึดขดลวดอาร์เมเจอร์ไว้ ปลายขดลวดอาร์เมเจอร์ต่อไว้กับคอมมิวเตเตอร์ อาร์เมอเจอร์ผลักดันของสนามแม่เหล็กทั้งสอง ทำให้อาร์เมเจอร์หมุนเคลื่อนที่

**2.2.1.5 คอมมิวเตเตอร์ (Commutator)**

คือส่วนเคลื่อนที่อีกส่วนหนึ่ง ถูกยึดติดเข้ากับอาร์เมเจอร์และเพลาร่วมกัน คอมมิวเตเตอร์ทำจากแท่งทองแดงแข็งประกอบเข้าด้วยกันี้เป็นรูปทรงกระบอก แต่ละแท่งทองแดงของคอมมิวเตเตอร์ถูกแยกอกจากกันด้วยฉนวนไมก้า (Mica) อาร์เมเจอร์ คอมมิวเตเตอร์ทำหน้าที่เป็นขั้วรับแรงดันไฟตรงที่จ่ายมาจากแปรงถ่าน เพื่อส่งไปให้ขดลวดอาร์เมอร์

**2.2.1.6 แปรงถ่าน (Brush)**

คือตัวสัมผัสกับคอมมิวเตเตอร์ ทำเป็นแท่งสี่เหลี่ยมผลิตมาจากคาร์บอน หรือแกรไฟต์ผสมผงทองแดง เพื่อให้แข็งและนำไฟฟ้าได้ดี มีสายตัวนำต่อร่วมกับแปรงถ่านเพื่อไปรับแรงดันไฟตรงที่จ่ายเข้ามา แปรงถ่านทำหน้าที่รับแรงดันไฟตรงจกแหล่งจ่าย จ่ายผ่านไปให้คอมมิวเตเตอร์



**รูปที่ 2.8 ส่วนประกอบและการทำงานของมอเตอร์ไฟฟ้ากระแสตรง**

**2.2.2** การทำงานของมอเตอร์ไฟฟ้ากระแสตรง

**การทำงานเบื้องต้นของมอเตอร์ไฟฟ้ากระแสตรง เมื่อมีแรงดันไฟตรงจ่ายผ่านแปรงถ่านไปคอมมิวเตเตอร์ ผ่านไปให้ขดลวดตัวนำทีอาร์เมเจอร์ ทำให้ขดลวดอาร์เมเจอร์เกิดสนามแม่เหล็กไฟฟ้าขึ้นมา ทางด้านซ้ายมือเป็นขั้วเหนือ (N) แล้ะด้านขวาเป็นขั้วใต้ (S) เหมือนกับขั้วแม่เหล็กถาวรที่ว่างอยู่ใกล้ ๆ เกิดอำนาจแม่เหล็กผลักดันกัน อาร์เมเจอร์หมุนไปในทิศทางตามเข็มนาฬิกา พร้อมกับคอมมิวเตเตอร์หมุนตามไปด้วย แปรงถ่านสัมผัสกับส่วนของคอมมิวเตเตอร์ เปลี่ยนไปในอีกปลายหนึ่งของขดลวด แต่มีผลทำให้เกิดขั้วแม่เหล็กที่อาร์เมเจอร์เหมือนกับชั้วแม่เหล็กถาวรที่อยู่ใกล้ ๆ อีกครั้ง ทำให้อาร์เมเจอร์ยังคงถูกผลักให้หมุนไปในทิศทางตามเข็มนาฬิกาตลอดเวลา เกิดการหมุนของอาร์เมเจอร์คือมอเตอร์ไฟฟ้าทำงาน**

**2.3** หม้อแปลงไฟฟ้า **( Transformer )**

**การทำงานของหม้อแปลงไฟฟ้านั้น อาศัยหลักการความสัมพันธ์ระหว่างกระแสไฟฟ้ากับเส้นแรงแม่เหล็กในการสร้างแรงเคลื่อนเหนี่ยวนำให้กับตัวนำ คือ เมื่อมีกระแสไหลผ่านขดลวดตัวนำ ก็จะทำให้เกิดเส้นแรงแม่เหล็กรอบ ๆตัวนำนั้น และถ้ากระแสที่ป้อนมีขนาดและทิศทางที่เปลี่ยนแปลงไปมา ก็จะทำให้สนามแม่เหล็กที่เกิดขึ้นมีการเปลี่ยนแปลงตามไปด้วย ถ้าสนามแม่เหล็กที่มีการเปลี่ยนแปลงดังกล่าวตัดผ่านตัวนำ ก็จะเกิดแรงเคลื่อนเหนี่ยวนำขึ้นที่ตัวนำนั้น โดยขนาดของแรงเคลื่อนเหนี่ยวนำจะสัมพันธ์กับ ความเข้มของสนามแม่เหล็ก และความเร็วในการตัดผ่านตัวนำของสนามแม่เหล็ก**

**2.3.1** ชนิดของหม้อแปลงไฟฟ้า

**2.3.1.1** หม้อแปลงชนิด แกนเหล็ก **(Iron Core Transformer)**

**หม้อแปลงแบบนี้จะใช้ แผ่นเหล็กอ่อนหลาย ๆแผ่นส่วนใหญ่จะใช้รูปทรงตัว E กับ ตัว I ประกอบกันเป็นแกนซึ่งส่วนใหญ่จะใช้ในงานทั่วไปที่มีความถี่ไม่สูงนัก เช่นหม้อแปลงในงานส่งกำลังไฟฟ้า หรือหม้อแปลงแปลง แรงดันไฟฟ้าตามบ้าน เป็นแรงดันต่ำ ๆตามที่ต้องการ หม้อแปลงชนิดนี้จะมีประสิทธิภาพสูงที่สุด**

**2.3.1.2** หม้อแปลงชนิดแกนเฟอร์ไรท์ **(Ferrite Core Transformer)**

**หม้อแปลงชนิดนี้ส่วนใหญ่จะใช้ในงานที่มีความถี่สูง เช่นในเครื่องรับ เครื่องส่ง วิทยุ หรือในวงจรสวิตชิ่ง เพราะไม่สามารถใช้หม้อแปลงชนิดแกนเหล็กได้**

**2.3.1.3** หม้อแปลงชนิดแกนอากาศ **(Air Core Transformer)**

**หม้อแปลงชนิดนี้จะใช้ในงานความถี่สูงมาก ๆ เช่นในเครื่องรับ เครื่องส่งวิทยุ ความถี่สูง เพราะไม่สามารถใช้หม้อแปลงชนิดอื่นได้เนื่องจากจะเกิดความสูญเสียอย่างมาก**

**2.3.2** ข้อควรระวังในการใช้งาน

1. **เลือกชนิดหม้อแปลงให้เหมาะสมกับความถี่ที่ใช้งาน**

2. **การใช้งานหม้อแปลงควรคำนึงถึงอัตราการทนกำลังของหม้อแปลงด้วยมิฉะนั้นจะทำให้หม้อแปลงไหม้ได้ เนื่องจากมีกระแสไหลสูงเกินไป**

3. **หม้อแปลงเป็นอุปกรณ์ที่ใช้งานกับไฟสลับ จึงไม่ควรป้อนไฟตรงเข้าที่ขั้วหม้อแปลงเพราะอาจจะทำให้หม้อแปลงไหม้ได้**

4. **ถ้าต่อใช้งานหม้อแปลงในลักษณะ ออโต้ทรานส์ฟอร์เมอร์ (**Auto Transformer) **ควรระวังถูกไฟฟ้าดูดด้วยเนื่องจากไม่มีการแยกการเชื่อมต่อทางไฟฟ้าจากแหล่งจ่ายไฟบ้าน เหมือนกับหม้อแปลงที่ใช้งานในลักษณะปกติ**

5. **หม้อแปลงเป็นอุปกรณ์ที่ทำงานด้วยสนามแม่เหล็ก ในขณะใช้งานจึงควรระวังไม่นำไปไกลอุปกรณ์ที่มีผลต่อสนามแม่เหล็ก เช่น แผ่นดิสก์ เทปเสียง หรือ จอภาพโทรทัศน์**



รูปที่ **2.9** หม้อแปลงไฟฟ้า **220V AC to 12V DC**

**2.4** อาร์ดุยโน่ บอร์ด **( Board Arduino )**

**บอร์ดอาร์ดุยโน เป็นบอร์ดไมโครคอนโทรลเลอร์ตระกูล** AVR **ที่มีการพัฒนาแบบ** Open Source **คือ มีการเปิดเผยข้อมูลทั้งด้าน** Hardware **และ** Software **ตัวบอร์ดอาร์ดุยโนถูกออกแบบมาให้ใช้งานได้ง่าย และสามารถใช้ต่ออุปกรณ์เสริมต่างๆ โดยผู้ใช้สามารถต่อวงจรอิเล็กทรอนิกส์จากภายนอกแล้วเชื่อม ต่อมาที่ขา** I/O **ของบอร์ด หรือเลือดต่อกับบอร์ดเสริมประเภทต่างๆได้**

**2.4.1 จุดเด่นของบอร์ดอาร์ดุยโน**

1. ง่ายต่อการพัฒนา มีรูปแบบคำสั่งพื้นฐานไม่ซับซ้อน

2. Open Hardware ทำให้ผู้ใช้สามารถนำบอร์ดไปต่อยอดใช้งานได้หลายด้าน

3. Arduino Community กลุ่มคนที่ร่วมกันตั้งใจพัฒนา

4. ราคาถูก

5. Cross Platform สามารถพัฒนาโปรแกรมบน OS ใดก็ได้

**2.4.2 บอร์ด อาร์ดุยโน่มีด้วยกันหลักๆประมาณ 9 บอร์ดดังนี้**

**2.4.2.1 Arduino Uno**

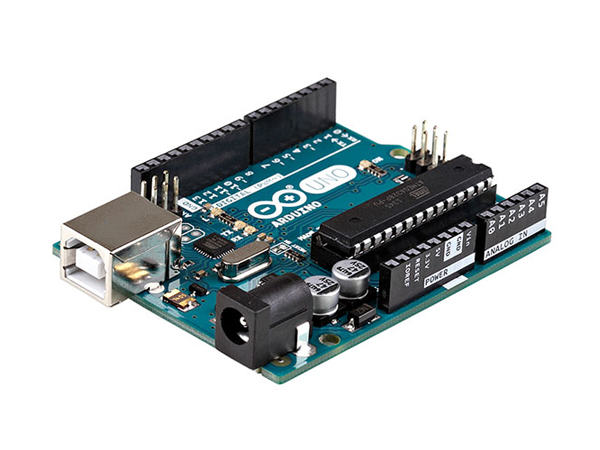
คำว่า Uno เป็นภาษาอิตาลีซึ่งแปลว่าหนึ่ง เป็นบอร์ด Arduino รุ่นแรกที่ออกมามีขนาดประมาณ 68.6x53.4mm เป็นบอร์ดมาตรฐานที่นิยมใช้งานมากที่สุด เนื่องจากเป็นขนาดที่เหมาะสำหรับการเริ่มต้นเรียนรู้ Arduino และมี Shields ให้เลือกใช้งานได้มากกว่าบอร์ด Arduino รุ่นอื่น ๆ ที่ออกแบบมาเฉพาะมากกว่า โดยบอร์ด Arduino Uno ได้มีการพัฒนาเรื่อยมา ตั้งแต่ R2 R3 และรุ่นย่อยที่เปลี่ยนชิพไอซีเป็นแบบ SMD

**ตารางที่ 2.1 ข้อมูลจำเพาะของ Arduino Uno**

|  |  |
| --- | --- |
| ชิปไอซีไมโครคอนโทรเลอร์ | 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 | 16MHz |

**ตารางที่ 2.1 ข้อมูลจำเพาะของ Arduino Uno (ต่อ)**

|  |  |
| --- | --- |
| ขนาด | 68.6x53.4 mm |
| น้ำหนัก | 25 กรัม |



**รูปที่ 2.10 บอร์ด Arduino Uno R3**

**2.4.2.2 Arduino Duo**

Duo เป็นภาษาอิตาลี แปลว่า สอง เป็นรุ่นที่เพิ่มพอร์ตให้มากขึ้นเป็น 54 พอร์ตดิจิตอลอินพุตเอาต์พุต และ 12 พอร์ตอนาล็อกอินพุต 2 พอร์ตอนาล็อกเอาต์พุต เพิ่มพื้นที่โปรแกรมเป็น 512KB สามารถใช้งานพื้นที่ได้เต็มไม่มี Bootloader เนื่องจากสามารถใช้กับพอร์ต USB ได้โดยตรง มีขนาดบอร์ด 101.52x53.3mm ใช้ชิปไอซีเบอร์ AT91SAM3X8E ซึ่งเป็นชิปไอซีที่ใช้เทคโนโลยี ARM Core สถาปัตยกรรม 32 บิต เร่งความถี่คริสตอลขึ้นไปสูงถึง 84MHz จึงทำให้สามารถงานด้านการคำนวณ หรือการประมวลผลอัลกอริทึมได้เร็วกว่า Arduino Uno มาก แต่เนื่องจากชิพไอซีทำงานที่แรงดัน 3.3V ดังนั้นการนำไปใช้งานกับเซ็นเซอร์ควรระวังไม่ให้แรงดัน 5V ไหลเข้าบอร์ด ควรใช้วงจรแบ่งแรงดันเพื่อช่วยให้ลอจิกลดแรงดันลงมาให้เหมาะสม

**ตารางที่ 2.2 ข้อมูลจำเพาะของ Arduino Duo**

|  |  |
| --- | --- |
| ชิปไอซีไมโครคอนโทรเลอร์ | 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 |

**ตารางที่ 2.2 ข้อมูลจำเพาะของ Arduino Duo (ต่อ)**

|  |  |
| --- | --- |
| กระแสไฟที่จ่ายได้ในแต่ละพอร์ต | 130mA |
| กระแสไฟที่จ่ายได้ในพอร์ต | 3.3V 800mA |
| พื้นที่โปรแกรมภายใน | 512KB พื้นที่โปรแกรม |
| พื้นที่แรม | 2KB |
| พื้นที่หน่วยความจำถาวร (EEPROM) | 96KB |
| ความถี่คริสตัล 16MHz | 84MHz |
| ขนาด | 101.52x53.3 mm |
| น้ำหนัก | 36 กรัม |

**2.4.2.3 Arduino Leonardo**

Arduino Leonardo เป็นบอร์ดที่เลือกใช้ชิพไอซีเบอร์ ATmega32u4 ที่รองรับการเชื่อมต่อกับพอร์ต USB ได้โดยตรง ทำให้บอร์ดสามารถเขียนโปรแกรมเพื่อจำลองตัวเองให้เป็นเมาส์ หรือคีย์บอร์ดได้ ทำงานที่แรงดัน 5V ทำให้ไม่มีปัญหากับเซ็นเซอร์ หรือ Shields ที่ใช้งานกับ Arduino Uno

**ตารางที่ 2.3 ข้อมูลจำเพาะของ Arduino Leonardo**

|  |  |
| --- | --- |
| 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 แต่ 4KB ถูกใช้โดย Bootloader |
| พื้นที่แรม | 2.5KB |
| พื้นที่หน่วยความจำถาวร (EEPROM) | 1KB |
| ความถี่คริสตัล 16MHz | 16MHz |
| ขนาด | 68.6x53.3 mm |

**ตารางที่ 2.3 ข้อมูลจำเพาะของ Arduino Leonardo (ต่อ)**

|  |  |
| --- | --- |
| น้ำหนัก | 20 กรัม |

**2.4.2.4 Arduino MEGA ADK**

ใช้ชิปไมโครคอนโทรเลอร์เบอร์ ATmaega2560 มีชิพไอซี USB Host เบอร์ MAX3421e มาให้บนบอร์ด ใช้สำหรับเชื่อมต่อกับโทรศัพท์มือถือแอนดรอยด์ผ่าน OTG มีพอร์ตดิจิตอลอินพุตเอาต์พุตจำนวน 54 พอร์ต มีอนาล็อกอินพุตมาให้ 16 พอร์ต ทำงานที่ความถี่ 16MHz บอร์ด Arduino MEGA ADK เหมาะสำหรับงานที่ใช้การเชื่อมต่อกับโทรศัพท์มือถือแอนดรอยด์

**ตารางที่ 2.4 ข้อมูลจำเพาะของ Arduino MEGA ADK**

|  |  |
| --- | --- |
| ชิปไอซีไมโครคอนโทรเลอร์ | ATmega32u4 |
| 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 |
| กระแสไฟที่จ่ายได้ในแต่ละพอร์ต | 40mA |
| กระแสไฟที่จ่ายได้ในพอร์ต | 3.3V 50mA |
| พื้นที่โปรแกรมภายใน | 256KB แต่ 8KB ถูกใช้โดย Bootloader |
| พื้นที่แรม | 8KB |
| พื้นที่หน่วยความจำถาวร (EEPROM) | 4KB |
| ความถี่คริสตัล 16MHz | 16MHz |
| ขนาด | 101.52x53.3 mm |
| น้ำหนัก | 36 กรัม |

**2.4.2.5 Arduino Mega 2560**

Arduino Mega 2560 จะเหมือนกับ Arduino MEGA ADK ต่างกันตรงที่บนบอร์ดไม่มี USB Host มาให้การโปรแกรมยังต้องทำผ่านโปรโตคอล UART อยู่ บนบอร์ดใช้ชิพไอซีไมโครคอนโทรเลอร์เบอร์ ATmega 2560

**ตารางที่ 2.5 ข้อมูลจำเพาะของ Arduino MEGA 2560**

|  |  |
| --- | --- |
| 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 |
| กระแสไฟที่จ่ายได้ในแต่ละพอร์ต | 40mA |
| กระแสไฟที่จ่ายได้ในพอร์ต | 3.3V 50mA |
| พื้นที่โปรแกรมภายใน | 256KB แต่ 8KB ถูกใช้โดย Bootloader |
| พื้นที่แรม | 8KB |
| พื้นที่หน่วยความจำถาวร (EEPROM) | 4KB |
| ความถี่คริสตัล 16MHz | 16MHz |

**2.4.2.6 Arduino Micro**

Arduino Micro ออกแบบให้มีขนาดที่เล็ก และทันสมัยกว่าบอร์ด Arduino Mini หรือ Arduino Nano เนื่องจากบนบอร์ดใช้ชิพไอซีไมโครคอนโทรเลอร์เบอร์ ATmega32u4 ซึ่งมาพอร์ต USB สามารถเชื่อมต่อกับคอมพิวเตอร์ได้โดยตรง และมีดิจิตอลอินพุตเอาต์พุตมากถึง 20 พอร์ต มีพื้นที่เก็บโปรแกรมขนาด 32KB แต่ต้องใช้พื้นที่สำหรับ Bootloader ไป 4KB มีขนาดเพียง 48x18mm เนื่องจากบอร์ดใช้ชิปไอซีตัวเดียวกับ Arduino Leonardo ทำให้สามารถทำให้บอร์ดจำลองตัวเองเป็นเมาส์หรือคีย์บอร์ดเชื่อมต่อกับคอมพิวเตอร์ได้

**ตารางที่ 2.6 ข้อมูลจำเพาะของ Arduino Micro**

|  |  |
| --- | --- |
| 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 |

**ตารางที่ 2.6 ข้อมูลจำเพาะของ Arduino Micro** (ต่อ)

|  |  |
| --- | --- |
| กระแสไฟที่จ่ายได้ในแต่ละพอร์ต | 40mA |
| กระแสไฟที่จ่ายได้ในพอร์ต | 3.3V 50mA |
| พื้นที่โปรแกรมภายใน | 32KB แต่ 4KB ถูกใช้โดย Bootloader |
| พื้นที่แรม | 2.5KB |
| พื้นที่หน่วยความจำถาวร (EEPROM) | 1KB |
| ความถี่คริสตัล 16MHz | 16MHz |
| ขนาด | 48x18 mm |
| น้ำหนัก | 13 กรัม |

**2.4.2.7 Arduino Nano**

Arduino Nanoออกแบบมาให้มีขนาดเล็กและใช้กับงานทั่ว ๆ ไปใช้ชิปไอซีไมโครคอนโทรเลอร์เบอร์ ATmega168 หรือเบอร์ ATmega328 โปรแกรมผ่านโปรโตคอล UART มีชิพUSB to UART มาให้ ใช้ Mini USB เชื่อมต่อกับคอมพิวเตอร์ มีพอร์ตดิจิตอลอินพุตเอาต์พุต 14 พอร์ต มีพอร์ตอนาล็อกอินพุต 8 พอร์ต บนบอร์ดยังมีเร็คกูเลเตอร์สามารถจ่ายไฟได้ตั้งแต่ 7 – 12V เพื่อให้บอร์ดทำงานได้ (จ่ายไฟที่ขา VIN) กรณีมีแหล่งจ่ายไฟ 5V อยู่แล้วก็จ่ายเข้าได้เลยที่ขา 5V

**ตารางที่ 2.7 ข้อมูลจำเพาะของ Arduino Nano**

|  |  |
| --- | --- |
| ชิปไอซีไมโครคอนโทรเลอร์ | 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, 500B ใช้โดยBootloader |
| พื้นที่แรม | 1 หรือ 2KB |
| พื้นที่หน่วยความจำถาวร (EEPROM) | 512B หรือ 1KB |
| ความถี่คริสตัล 16MHz | 16MHz |

**ตารางที่ 2.7 ข้อมูลจำเพาะของ Arduino Nano (ต่อ)**

|  |  |
| --- | --- |
| ขนาด | 45x18 mm |
| น้ำหนัก | 5 กรัม |

**2.4.2.8 Arduino Mini**

Arduino Mini มีขนาดเล็กกว่าบอร์ด Arduino อื่น ๆอยู่มาก แต่ยังคงความสามารถไว้เท่ากับบอร์ด Arduino Uno R3 แถมยังมีพอร์ต A6 และ A7 เพิ่มขึ้นมา ทำให้บอร์ดมีอนาล็อกอินพุตเพิ่มมากขึ้น จากเดิมมี 6 พอร์ต เพิ่มเป็น 8 พอร์ต เนื่องจากบอร์ด Arduino Mini เน้นที่ขนาดเล็ก ดังนั้นจึงไม่สามารถทำการโปรแกรมได้โดยตรง หากต้องการโปรแกรมบอร์ดจำเป็นต้องซื้อโมดูล USB to UART มาใช้แยกตางหาก บอร์ด Arduino Mini ยังคงมีรูปแบบคล้ายๆกับ Arduino เดิม คือใช้ชิพ ATmega328 ที่ความถี่ 16MHz ภายในบอร์ดสามารถใช้แหล่งจ่ายไฟ 7 – 12V มาจ่ายได้ หากมีแหล่งจ่ายไฟ 5V ก็สามารถนำมาจ่ายได้

**ตารางที่ 2.8 ข้อมูลจำเพาะของ Arduino Mini**

|  |  |
| --- | --- |
| ชิปไอซีไมโครคอนโทรเลอร์ | 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 |
| กระแสไฟที่จ่ายได้ในพอร์ต | 3.3V 50mA |
| พื้นที่โปรแกรมภายใน | 32KB พื้นที่โปรแกรม, 2KB ใช้โดย Bootloader |
| พื้นที่แรม | 2KB |
| พื้นที่หน่วยความจำถาวร (EEPROM) | 1KB |
| ความถี่คริสตัล 16MHz | 16MHz |
| ขนาด | 30x18 mm |
| น้ำหนัก | ไม่ระบุ |

**2.4.2.9 Arduino Pro Mini**

บอร์ด Arduino Pro Mini เป็นบอร์ดที่แตกต่างจากบอร์ด Arduino Mini คือย้ายช่อง A4 A5 A6 A7 ออกมาภายในบอร์ด เพื่อให้บอร์ดมีขนาดที่เล็กลงกว่าเดิมอีก 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

**ตารางที่ 2.9 ข้อมูลจำเพาะของ Arduino Pro Mini**

|  |  |
| --- | --- |
| ชิปไอซีไมโครคอนโทรเลอร์ | 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 |
| กระแสไฟที่จ่ายได้ในพอร์ต | 3.3V 50mA |
| พื้นที่โปรแกรมภายใน | 32KB พื้นที่โปรแกรม, 2KB ใช้โดย Bootloader |
| พื้นที่แรม | 2KB |
| พื้นที่หน่วยความจำถาวร (EEPROM) | 1KB |
| ความถี่คริสตัล 16MHz | 8MHz (ในโมเดลใช้ไฟ 3.3V) หรือ 16MHz  (ในโมเดลใช้ไฟ 5V) |
| ขนาด | 30x18 mm |
| น้ำหนัก | ไม่ระบุ |

**2.5 ภาษา C**

ภาษาซีพัฒนาขึ้นมาในปี 1970 โดย Dennis Ritchie แห่ง Bell Telephone Labora-tories, Inc. (ปัจจุบันคือ AT&T Bell Laboratories) ซึ่งภาษาซีนั้นมีต้นกำเนิดมาจากภาษา 2 ภาษา คือ ภาษา BCPL และ ภาษา B ซึ่งต่างก็เป็นภาษาที่พัฒนาขึ้นมาใน Bell Laboratories เช่นกัน ภาษาซีนั้นถูกใช้งานอยู่เพียงใน Bell Laboratories จนกระทั่งปี 1978 Brian Kernighan และ Ritchieนั้นเป็นที่รู้จักกันในชื่อของ"K&R C"หลังจากที่ตีพิมพ์ข้อกำหนดของ K&R นักคอมพิวเตอร์มืออาชีพรู้สึกประทับใจกับคุณสมบัติที่น่าสนใจของภาษาซี และเริ่มส่งเสริมการใช้งานภาษาซีมากขึ้น ในกลางปี 1980 ภาษาซีก็กลายเป็นภาษาที่ได้รับความนิยมโดยทั่วไป มีการพัฒนาตัวแปลโปรแกรม และตัวแปลคำสั่งภาษาซีจำนวนมากสำหรับคอมพิวเตอร์ทุกขนาด และภาษาซีก็ถูกนำมาไปใช้สำหรับพัฒนาโปรแกรมเชิงพาณิชย์เป็นจำนวนมาก ยิ่งไปกว่านั้นโปรแกรมเชิงพาณิชย์ที่เคยพัฒนาขึ้นมาโดยใช้ภาษาอื่น ก็ถูกเขียนขึ้นใหม่โดยใช้ภาษาซี เนื่องจากความต้องการใช้ความได้เปรียบทางด้านประสิทธิภาพ และความสามารถในการเคลื่อนย้ายได้ของภาษาซี ตัวแปลโปรแกรมภาษาซีที่ได้รับการพัฒนาขึ้นมาในเชิงพาณิชย์นั้น จะมีความแตกต่างกับข้อกำหนดของ Kernighan และ Ritchie อยู่บ้าง จากจุดนี้เองทำให้เกิดความไม่เข้ากันระหว่างตัวแปลโปรแกรมภาษาซีซึ่งก็ทำให้สูญเสียคุณสมบัติการเคลื่อนย้ายได้ซึ่งเป็นคุณสมบัติที่สำคัญของภาษา ดังนั้นสถาบันมาตรฐานแห่งชาติอเมริกัน (American National Standard Institute) หรือ แอนซี (ANSI) จึงเริ่มจัดทำมาตรฐานของภาษาซีขึ้น (ANSI committee X3J11X ซึ่งกำลังอยู่ในช่วงการดำเนินงาน

**2.5.1 คุณสมบัติของภาษา C**

1. เป็นภาษาที่มีลักษณะเป็นโครงสร้างจึงเขียนโปรแกรมง่าย โปรแกรมที่ เขียนขึ้นจะทำงานได้อย่างมีประสิทธิภาพสูง สั่งงานคอมพิวเตอร์ได้รวดเร็วกว่าภาษาระดับสูงอื่น ๆ

2. สั่งงานอุปกรณ์ในระบบคอมพิวเตอร์ได้เกือบทุกส่วนของฮาร์ดแวร์ ซึ่งภาษาระดับสูงภาษาอื่นทำงานดังกล่าวได้น้อยกว่า

3. คอมไพเลอร์ภาษาซีทุกโปรแกรมในท้องตลาดจะทำงานอ้างอิง มาตรฐาน(ANSI= American National Standards Institute) เกือบทั้งหมด จึงทำให้โปรแกรมที่เขียนขึ้นด้วยภาษาซีสามารถนำไปใช้กับคอมพิวเตอร์ได้ทุกรุ่นที่มาตรฐาน ANSI รับรอง

4. โปรแกรมที่เขียนขึ้นด้วยภาษาซีสามารถนำไปใช้กับเครื่องคอมพิวเตอร์ ที่ใช้ซีพียูต่างเบอร์กันได้ หรือกล่าวได้ว่าโปรแกรมมีความยืดหยุ่น (portability) สูง

5. สามารถนำภาษาซีไปใช้ในการเขียนโปรแกรมประยุกต์ได้หลายระดับ เช่น เขียนโปรแกรมจัดระบบงาน (OS) คอมไพเลอร์ของภาษาอื่น โปรแกรมสื่อสารข้อมูลโปรแกรมจัดฐานข้อมูล โปรแกรมปัญญาประดิษฐ์ (AI = Artificial Intelligence) รวมทั้งโปรแกรมคำนวณงานทางด้านวิทยาศาสตร์และวิศวกรรมศาสตร์

6. มีโปรแกรมช่วย (tool box) ที่ช่วยในการเขียนโปรแกรมมาก และ ราคาไม่แพงหาซื้อได้ง่าย

7. สามารถประกาศข้อมูลได้หลายชนิดและหลายรูปแบบ ทำให้สะดวก รวดเร็วต่อการพัฒนาโปรแกรมตามวัตถุประสงค์ของผู้ใช้

8. ประยุกต์ใช้ในงานสื่อสารข้อมูล และงานควบคุมที่ต้องการความแม่นยำ ในเรื่องเวลา (real time application) ได้ดีกว่าภาษาระดับสูงอื่น ๆ หลายๆภาษา

9. สามารถเขียนโปรแกรมด้วยเทคนิคแบบโอโอพี (OOP = Object Oriented Programming) ได้หากใช้ภาษาซีรุ่น TURBO C++ ขึ้นไป ทำให้สามารถพัฒนาโปรแกรมประยุกต์เพื่อใช้งานได้กว้างขวางมากยิ่งขึ้นกว่าเดิม

**2.5.2 โครงสร้างของโปรแกรม**

**2.5.2.1 Compiler Directive**

เป็นส่วนของโปรแกรมที่ใช้สำหรับเป็นตัวบอกคอมไพเลอร์ว่าให้รวมไฟล์ต่าง ๆ ที่กำหนดไว้ในส่วนนี้กับตัวโปรแกรมที่เขียนขึ้น โดยต้องเขียนตามรูปแบบที่กำหนดไว้

**2.5.2.2 ตัวโปรแกรม (body)**

เป็นส่วนที่ผู้ใช้ต้องเขียนขึ้นเองโดยนำเอาคำสั่งหรือฟังก์ชันมาตรฐานต่าง ๆ มาเรียบเรียงกันขึ้นเป็นโปรแกรมเพื่อสั่งให้คอมพิวเตอร์รับข้อมูลจากอุปกรณ์อินพุตประมวลผล จะขึ้นต้นด้วย main() { } ซึ่งภายในวงเล็บประกอบด้วยคำสั่ง ต่างๆ

**2.5.2.3 ส่วนคำอธิบายโปรแกรม (comment lines)**

ส่วนนี้ใช้ในการอธิบายโปรแกรมอาจพิมพ์ข้อความ สูตรหรืออื่น ๆ ที่เกี่ยวข้องกับโปรแกรมโดยอธิบายส่วนต่างๆ ของโปรแกรม ว่าแต่ละส่วนทำหน้าที่อะไร ตัวแปรแต่ละตัวใช้แทนค่าอะไร ส่วนของโปรแกรมส่วนนี้จะมีหรือไม่มีก็ได้ ถ้าต้องเขียนให้เริ่มต้นด้วยเครื่องหมาย /\* ตามด้วยข้อความที่ต้องการ แล้วปิดท้ายด้วย \*/

**2.5.3 ชนิดของตัวแปร**

**ตารางที่ 2.10 ชนิดของตัวแปร**

|  |  |
| --- | --- |
| Int | เป็นตัวแปรที่เก็บค่าเป็นเลขจำนวนเต็ม Integer |
| Float | เป็นตัวแปรที่เก็บค่าที่เป็นเลขทศนิยม |
| Short | เป็นตัวแปรที่เก็บค่าที่เป็นเลขจำนวนเต็มที่มีค่าน้อยกว่าค่าของตัว แปร ที่ประกาศไว้เป็นชนิด int |
| Long | เป็นตัวแปรที่เก็บค่าที่เป็นเลขจำนวนเต็มที่มีจำนวนบิตมากเป็น 2 เท่าของเดิม |
| Double | เป็นตัวแปรที่เก็บค่าที่เป็นทศนิยมที่มีจำนวนบิตมากเป็น 2 เท่าของเดิม |

**ตารางที่ 2.10 ชนิดของตัวแปร (ต่อ)**

|  |  |
| --- | --- |
| Unsigned | เป็นตัวแปรที่เก็บค่าจำนวนเต็มที่มีค่าเป็นบวกเท่านั้นเอาไว้ |
| Char | เป็นตัวแปรที่เก็บค่าเป็นตัวอักษร |

**2.5.4 Array**

ในการนำค่าไปเก็บในหน่วยความจำหลาย ๆ ค่า จะต้องตั้งชื่อตัวแปรหลายตัวซึ่งทำให้มีปัญหากับการตั้งชื่อดังนั้นการตั้งชื่อตัวแปรสามารถทำได้อีกแบบหนึ่ง คือตั้งชื่อตัวแปรเพียงชื่อเดียวแล้วใช้ตัวเลขกำกับว่าเป็นตัวแปรตัวที่เท่าไร ในชุดนั้นลักษณะการตั้งชื่อตัวแปรแบบนี้เรียกว่าตัวแปรชุดและตัวเลขที่บอกตำแหน่งนั้นเรียกว่า ดรรชนี

1. ตัวแปรชุดชนิด 1 มิติ (One Dimension)

2. ตัวแปรชุดชนิด 2 มิติ (Two-Dimension)

**2.5.5 เครื่องหมายดำเนินการ (Operator)**

**ตารางที่ 2.11 เครื่องหมายดำเนินการ (Operator)**

|  |  |  |
| --- | --- | --- |
| + | การบวก | A+B |
| - | การลบ | A-B |
| \* | การคูณ | A\*B |
| / | การหาร | A/B |
| % | หารเอาแต่เศษไว้ | 5% 3=1 เศษ 2 จะเก็บแต่เศษ 2เอาไว้ |
| -- | การลดค่าครั้งละ 1 เสมอ | A-- // A = A-1 |
| ++ | การเพิ่มค่าครั้งละ 1 เสอม | A++// A = A+1 |

**2.5.6 เครื่องหมายเปรียบเทียบ (Relational and Logical Operator**)

หมายถึง เครื่องหมายที่ใช้ในการเปรียบเทียบและตัดสินใจ ซึ่งผลของการเปรียบเทียบจะได้เป็น 2 กรณี คือ จริง (true) จะให้ค่าเป็น 1 เท็จ (false) จะให้ค่าเป็น 0

**ตารางที่ 2.12 เครื่องหมายเปรียบเทียบ**

|  |  |  |
| --- | --- | --- |
| > | มากกว่า | A>B (Aมากกว่าB) |
| >= | มากกว่าหรือเท่ากับ | A>=B (Aมากกว่าหรือเท่ากับB) |

**ตารางที่ 2.12 เครื่องหมายเปรียบเทียบ (ต่อ)**

|  |  |  |
| --- | --- | --- |
| < | น้อยกว่า | A<B (Aน้อยกว่าB) |
| <= | น้อยกว่าหรือเท่ากับ | A<=B (Aน้อยกว่าหรือเท่ากับB) |
| == | เท่ากับ | A==B (Aเท่ากับB) |
| != | ไม่เท่ากับ | A!=B (Aไม่เท่ากับB) |

**2.5.7 ภาษา C ของอาร์ดุยโน่**

ภาษาซีของอาร์ดุยโนจะแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมออกเป็นส่วนย่อยหลายๆส่วน โดยเรียกแต่ละส่วนว่า ฟังก์ชันและเมื่อนำฟังก์ชันมารวมเข้าด้วยกัน จะเรียกว่าโปรแกรม โดยโปรแกรมเชิง โครงสร้าง (Structure Programming) การเขียนโปรแกรมของอาร์ดุยโนนั้น ๆ ทุกโปรแกรมจะต้อง ประกอบด้วยไปด้วยฟังก์ชันจำนวนเท่าใดก็ได้ แต่อย่างน้อยที่สุดต้องมีฟังก์ชัน จำนวน 2 ฟังก์ชัน คือ setup และ loop ในส่วนแรกของโปรแกรมจะเรียกว่า Header โดยจะประกอบด้วยคำสั่ง #include ซึ่งเป็นคำสั่งพิเศษ ที่เรียกว่า Compiler Directive คำสั่งนี้ไม่ใช่คำสั่งสำหรับสั่งงานในโปรแกรม ดังนั้นจึงต้องไม่ต้องมีเครื่องหมาย ( ; ) ปิดท้าย เหมือนคำสั่งอื่น ๆ โดย Compiler Directive จะใช้ทำหน้าที่สำหรับบอกให้ Compiler รับรู้ เงื่อนไขในการแปลคำสั่งเท่านั้น ซึ่งในกรณีคำสั่ง #include จะใช้สำหรับบอกให้ Compiler รับรู้ว่าการแปล คำสั่งของโปรแกรมนี้มีไฟล์ภายนอกใดบ้างที่จำเป็นต้องใช้ร่วมในการแปลคำสั่งให้กับโปรแกรม สำหรับส่วนที่มี ความสำคัญและจำเป็นที่สุดของโปรแกรมอาร์ดุยโน คือ ฟังก์ชัน setup() และฟังก์ชัน loop() ซึ่งฟังก์ชันทั้ง 2 ส่วนนี้มีรูปโครงสร้างที่เหมือนกันโดย setup() จะเขียนไว้ก่อน loop() ทั้ง 2 ฟังก์ชันมีขอบเขตเริ่มต้นและสิ้นสุด แยกออกจากกัน และอยู่ภายใต้เครื่องหมาย { } ตัวอย่างฟังก์ชัน setup()

**2.5.8 คำสั่งโปรแกรมของอาร์ดุยโน**

จะใช้ Library ของภาษา C/C++ ในการเขียนโปรแกรม ซึ่งภาษา C++ ดังนี้

**2.5.8.1 คำสั่ง void setup**

หน้าที่ของฟังก์ชัน setup() ในอาร์ดุยโนคือ ใช้ทำหน้าที่เป็นส่วนของโปรแกรมย่อย สำหรับใช้บรรจุ คำสั่งต่าง ๆ ที่ใช้สำหรับการกำหนดการทำงานของระบบหรือกำหนดคุณสมบัติของการทำงานให้กับอุปกรณ์ต่าง ๆซึ่งคำสั่งทั้งหมดที่บรรจุไว้ภายใต้ฟังก์ชันของ setup() นี้ จะถูกเรียกขึ้นมาทำงานเพียงรอบเดียวคือตอน เริ่มต้นการทำงานของโปรแกรม (หลังการรีเซตให้ไมโครคอนโทรลเลอร์เริ่มต้นทำงาน) เท่านั้น โดยคำสั่งที่นิยม บรรจุไว้ในฟังก์ชันส่วนนี้ได้แก่ คำสั่งกำหนดโหมดการท างานของ Digital Pin หรือคำสั่งสำหรับกำหนด คุณสมบัติของพอร์ตสื่อสารอนุกรม เป็นต้น

**2.5.8.2 คำสั่ง if…else**

คำสั่ง if…else ใช้สำหรับการตรวจสอบเงื่อนไขที่มีหลายทางเลือกเพิ่มขึ้นอีก 1 ทางเลือก คำสั่ง if...else แบบ 2 ทางเลือก มีความหมายว่า ถ้าเงื่อนไขเป็นจริงให้ทำตามเงื่อนไขของ if ถ้าเงื่อนไขเป็นเท็จให้ทำ ตามเงื่อนไขของ else จะเห็นได้ว่าโปรแกรมจะมีทางเลือกในการทำงานที่เพิ่มขึ้นมากกว่า 1 ทาง รวมเป็น 2 ทาง โดยทางเลือกแรกเป็นทางเลือกที่โปรแกรมจะทำงานเมื่อเงื่อนไขเป็นจริง ส่วนทางเลือกที่ 2 เป็นทางเลือก ที่จะให้โปรแกรมทำงานเมื่อเงื่อนไขเป็นเท็จ

**2.5.8.3 คำสั่ง Serial.print (data) และ Serial.println (data**)

คำสั่งนี้ใช้ทำหน้าที่สำหรับสั่งให้ข้อมูลออกไปยังพอร์ตสื่อสารอนุกรมอย่างต่อเนื่องในลักษณะของการ แสดงข้อมูล ตัวแปรหรือข้อความต่าง ๆ ที่กำหนดไว้ในพารามิเตอร์ภายในวงเล็บ () ออกทางพอร์ตสื่อสาร อนุกรม (พร้อมแสดงผลบนหน้าจอ หากบอร์ดอาร์ดุยโนต่ออยู่กับคอมพิวเตอร์) กลุ่มคำสั่ง Serial.print และ Serial.println โดยทั้ง 2 กลุ่มคำสั่งจะมีรูปแบบการใช้งานและให้ผลการทำงานที่เหมือนกัน แต่ต่างกันที่ Serial.println จะมีการเติมค่ารหัสขึ้นบรรทัดใหม่ (0x0D, 0x0A) ปิดท้ายการ ทำงานของคำสั่งไปด้วย แต่ส่วนอื่น ๆจะเหมือนกันหมด

**2.5.8.4 คำสั่ง void delay (ms)**

คำสั่งนี้ใช้ทำหน้าที่สำหรับหน่วงเวลา โดยมีหน่วยเป็น ms ซึ่งสามารถกำหนดค่าการหน่วงเวลาเป็นค่า ตัวเลขจำนวนเต็มระหว่าง 1 ถึง 4,294,967,295 ms โดยที่ค่าเวลาของ 1ms จะมีค่าเท่ากับ 1/1,000 s

**ตารางที่ 2.13 ค่าข้อมูลที่เก็บได้ของตัวแปร**

|  |  |  |
| --- | --- | --- |
| **ชนิดตัวแปร** | **จำนวนบิต** | **ค่าข้อมูลที่เก็บได้** |
| **char** | **8** | **-128 ถึง +127** |
| **int** | **16** | **-32768 ถึง +32767** |
| **float** | **32** | **3.4E-38 ถึง 3.4E+38** |
| **double** | **64** | **1.7E-308 ถึง +17E+308** |
| **void** | **0** | **0** |

**2.5.9 ตัวแปรแบบอาร์เรย์**

ตัวแปรแบบอาร์เรย์ มีลักษณะต่างจากตัวแปรอื่น ๆ คือ ตัวแปรแบบอื่น ๆ เมื่อประกาศขึ้นมาใช้งาน แล้ว ตัวแปร 1 ตัว สามารถเก็บข้อมูลได้เพียง 1 ค่า เท่านั้น เมื่อต้องการเก็บข้อมูลหลายๆ ค่า ก็จำเป็นต้อง สร้างตัวแปรขึ้นมาหลายๆ ตัว ซึ่งถ้ามีจำนวนตัวแปรในโปรแกรมมาก ๆ จะส่งผลให้เกิดความยุ่งยากในการ กำหนดชื่อและเรียกใช้งาน ภาษาซีจึงได้มีการสร้างตัวแปรแบบอาร์เรย์ขึ้นมาเพื่อใช้แก้ปัญหานี้ โดยอาร์เรย์เป็น ตัวแปรที่ใช้เก็บข้อมูลหลายๆค่าไว้ในตัวแปรเพียงชื่อเดียว แต่มีตัวเลขสำหรับชี้ตำแหน่งการเก็บข้อมูลต่างกัน โดยตัวเลขที่ใช้ทำหน้าที่เป็นตัวชี้ตำแหน่งของข้อมูลตัวแปรแบบ Array เรียกว่า Index Number ค่าลำดับของ ข้อมูลในตัวแปรอาร์เรย์ตำแหน่งแรกจะมีค่าเป็นศูนย์เสมอ

**2.5.10 Preprocessor directives**

โดยปกติแล้วเกือบทุกโปรแกรมต้องมี โดยส่วนนี้จะเป็นส่วนที่คอมไพลเลอร์จะมีการประมวลผลและทำตามคำสั่งก่อนที่จะมีการคอมไพล์โปรแกรม ซึ่งจะเริ่มต้นด้วยเครื่องหมายไดเร็กทีฟ (directive) หรือเครื่องหมายสี่เหลี่ยม # แล้วจึงตามด้วยชื่อคำสั่งที่ต้องการเรียกใช้ หรือกำหนด โดยปกติแล้วส่วนนี้จะอยู่ในส่วนบนสุด หรือส่วนหัวของโปรแกรม และต้องอยู่นอกฟังก์ชันหลักใดๆก็ตาม

#include เป็นคำสั่งที่ใช้อ้างอิงไฟล์ภายนอก เพื่อเรียกใช้ฟังก์ชัน หรือตัวแปรที่มีการสร้างหรือกำหนดไว้ในไฟล์นั้น รูปแบบการใช้งานคือ

#include <ชื่อไฟล์.h>

ตัวอย่างเช่น

#include <Wire.h>

#include <Time.h>

จากตัวอย่าง จะเห็นว่าได้มีการอ้างอิงไฟล์ Wire.h และไฟล์ Time.h ซึ่งเป็นไลบารี่พื้นฐานที่มีอยู่ใน Arduino ทำให้เราสามารถใช้ฟังก์ชั่นเกี่ยวกับเวลาที่ไลบารี่ Time มีการสร้างไว้ให้ใช้งานได้การอ้างอิงไฟล์จากภายใน หรือการอ้างอิงไฟล์ไลบารี่ที่มีอยู่แล้วใน Arduino หรือเป็นไลบารี่ที่เราเพิ่มเข้าไปเอง จะใช้เครื่องหมาย <> ในการคร่อมชื่อไฟล์ไว้ เพื่อให้โปรแกรมคอมไพลเลอร์เข้าใจว่าควรไปหาไฟล์เหล่านี้จากในโฟลเดอร์ไลบารี่ แต่หากต้องการอ้างอิงไฟล์ที่อยู่ในโฟลเดอร์โปรเจค จะต้องใช้เครื่องหมาย "" คร่อมแทน ซึ่งคอมไพล์เลอร์จะวิ่งไปหาไฟล์นี้โดยอ้างอิงจากไฟล์โปรแกรมที่คอมไพล์เลอร์อยู่

เช่น

#include "myFunction.h"

จากตัวอย่างด้านบน คอมไพล์เลอร์จะวิ่งไปหาไฟล์ myFunction.h ภายในโฟลเดอร์โปรเจคทันที หากไม่พบก็จะแจ้งเป็นข้อผิดพลาดออกมา #define เป็นคำสั่งที่ใช้ในการแทนข้อความที่กำหนดไว้ ด้วยข้อความที่กำหนดไว้ ซึ่งการใช้คำสั่งนี้ ข้อดีคือจะไม่มีการอ้างอิงกับตัวโปรแกรมเลยรูปแบบ

#define NAME VALUE

ตัวอย่างเช่น

#define LEDPIN 13

จากตัวอย่าง ไม่ว่าคำว่า LEDPIN จะอยู่ส่วนใดของโค้ดโปรแกรมก็ตาม คอมไพล์เลอร์จะแทนคำว่า LEDPIN ด้วยเลข 13 แทน ซึ่งข้อดีคือเราไม่ต้องสร้างเป็นตัวแปรขึ้นมาเพื่อเปลืองพื้นที่แรม และยังช่วยให้โปรแกรมทำงานเร็วขึ้นอีกด้วยเพราะซีพียูไม่ต้องไปขอข้อมูลมาจากแรมหลายๆทอด

**2.5.11 การกำหนดค่า**

ส่วนของการกำหนดค่า (Global declarations) ส่วนนี้จะเป็นส่วนที่ใช้ในการกำหนดชนิดตัวแปรแบบนอกฟังก์ชั่น หรือประกาศฟังก์ชั่น เพื่อให้ฟังก์ชั่นที่ประกาศสามารถกำหนด หรือเรียกใช้ได้จากทุกส่วนของโปรแกรม

เช่น

int pin = 13;

void blink(void) ;

**2.5.12** **ฟังก์ชั่น setup() และฟังก์ชั่น loop()**

ฟังก์ชั่น setup() และฟังก์ชั่น loop() เป็นคำสั่งที่ถูกบังคับให้ต้องมีในทุกโปรแกรม โดยฟังก์ชั่น setup() จะเป็นฟังก์ชั่นแรกที่ถูกเรียกใช้ นิยมใช้กำหนดค่า หรือเริ่มต้นใช้งานไลบารี่ต่างๆ เช่น ในฟังก์ชั่น setup() จะมีคำสั่ง pinMode() เพื่อกำหนดให้ขาใดๆก็ตามเป็นดิจิตอลอินพุต หรือเอาต์พุต ส่วนฟังก์ชั่น loop() จะเป็นฟังก์ชั่นที่ทำงานหลังจากฟังก์ชั่น setup() ได้ทำงานเสร็จสิ้นไปแล้ว และมีการวนรอบแบบไม่รู้จบ เมื่อฟังก์ชั่น loop() งานครบตามคำสั่งแล้ว ฟังก์ชั่น loop() ก็จะถูกเรียกขึ้นมาใช้อีก

ตัวอย่าง

int pin = 13;

void setup() {

pinMode(pin, OUTPUT);

}

void loop() {

digitalWrite(pin, HIGH);

delay(1000);

digitalWrite(pin, LOW);

delay(1000);

}

**2.5.12 การสร้างฟังก์ชั่น และการใช้งานฟังก์ชั่น (Users-defined function)**

ในการสร้างฟังก์ชั่นขึ้นมา คำสั่งต่างๆที่อยู่ภายในฟังก์ชั่น ต้องอยู่ภายใต้เครื่องหมายปีกกาเปิด { และปีกกาปิด } เท่านั้น ภายใต้เครื่องหมาย {} เราสามารถนำฟังก์ชั่นหรือคำสั่งใดๆก็ได้มาใส่ไว้ แต่จะต้องคั่นแต่ละคำสั่งด้วยเครื่องหมายเซมิโคล่อน ; โดยจะนำคำสั่งทั้งหมดไว้บรรทัดเดียวกันเลย หรือแยกบรรทัดกันก็ได้เพื่อความสวยงามของโค้ด (ไม่มีผลกับขนาดของโปรแกรมหลังคอมไพล์)

ตัวอย่าง

void Mode(int pin) {

pinMode(pin, OUTPUT);

}

void setup() {

Mode(13);

}

**2.5.13 ส่วนอธิบายโปรแกรม (Program comments)**

ส่วนอธิบายโปรแกรม หรือการคอมเม้นโปรแกรมเป็นส่วนที่สำคัญอย่างมากที่จะช่วยให้ผู้ที่ไม่ได้เขียนโปรแกรม หรือเป็นผู้เขียนโปรแกรมเข้าใจโปรแกรมได้ง่ายขึ้นโดยอ่านจากคอมเม้น แทนการทำความเข้าใจโปรแกรมโดยอ่านแต่ละฟังก์ชั่น ส่วนอธิบายโปรแกรม หรือส่วนคอมเม้นนี้ จะไม่มีผลใดๆกับขนาดของโปรแกรมหลังคอมไพล์ เนื่องจากส่วนนี้จะถูกตัดทิ้งทั้งหมดเนื่องจากไม่ได้ถูกนำไปใช้งาน มีผลเพียงแค่ว่าไฟล์โค้ดโปรแกรมจะใหญ่ขึ้นมา หากมีการคอมเม้นโค้ดเยอะๆ แต่ขนาดก็จะเพิ่มขึ้นตามตัวอักษร ดังนั้นการคอมเม้นโค้ดจึงไม่คิดพื้นที่มากนัก แต่ผู้เขียนแนะนำให้คอมเม้นโค้ดให้สั้น และกระชับ เพื่อให้เกิดความรวดเร็วในการทำความเข้าใจ และไม่ยาวจนต้องเลื่อนสกอร์บาร์ไปทางขวาเพื่ออ่านคอมเม้นเพิ่มเติม