ประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง
AUTOMATIC ENTRANCE-EXIT DOOR FOR PETS

โดย

ปัณณทัต ไชยรัตน์

ปฏิมา เนตรล้อมวงศ์

ภัทราพร พิมพา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง
AUTOMATIC ENTRANCE-EXIT DOOR FOR PETS

โดย

ปัณณทัต	ไชยรัตน์	57010674
ปฏิมา	เนตรล้อมวงศ์	57010720
ภัทราพร	พิมพา	57010966

อาจารย์ที่ปรึกษา

ผศ. ดร.สิรภพ ตู้ประกาย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ปริญญานิพนธ์ปีการศึกษา 2560
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง
automatic entrance-exit door for pets

ผู้จัดทำ

1.	ปัณณทัต	ไชยรัตน์	57010674
2.	ปฏิมา	เนตรล้อมวงศ์	57010720
3.	ภัทราพร	พิมพา	57010966

..... อาจารย์ที่ปรึกษา

(ผศ. ดร.สิรภพ ตู้ประกาย)

กิตติกรรมประกาศ

ปริญญานิพนธ์เล่มนี้อาจไม่สำเร็จไปด้วยดี หากไม่ได้รับความช่วยเหลือ และความร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลที่กลุ่มของข้าพเจ้าต้องกล่าวถึงเพราะเป็นบุคคลสำคัญที่ทำให้ปริญญานิพนธ์นี้สำเร็จได้ คือ ผศ. ดร.สิรภพ ตู้ประกาย อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่คอยให้คำปรึกษา ให้คำแนะนำ ช่วยในการวางแผน และดำเนินงานมาโดยตลอด

ขอขอบพระคุณคณาจารย์ ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ประสิทธิ์ประสาทวิชาความรู้ให้แก่ข้าพเจ้า

ขอขอบคุณเพื่อน, พี่, น้อง และทุกๆ คน ที่คอยให้ความช่วยเหลือ ความห่วงใยมาโดยตลอด และเป็นส่วนหนึ่งที่ทำให้ปริญญานิพนธ์เล่มนี้ลุล่วงมาได้

และอันดับสุดท้าย ต้องกราบขอบพระคุณบุคคลที่สำคัญยิ่ง ที่ทำให้ข้าพเจ้ามีวันนี้ได้ คือ บิดา มารดา ที่เป็นผู้ให้กำเนิด และเลี้ยงดูข้าพเจ้าเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และคอยเอาใจใส่ให้ความช่วยเหลือข้าพเจ้าในทุกๆ ด้านเป็นอย่างดี

ปัณณทัต ไชยรัตน์
ปฏิมา เนตรล้อมวงศ์
ภัทราพร พิมพา
ผู้จัดทำ

ประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง
Automatic entrance-exit door for pets

โดย ปัณณทัต ไชยรัตน์ 57010674
ปฏิมา เนตรล้อมวงศ์ 57010720
ภัทราพร พิมพา 57010966

อาจารย์ที่ปรึกษา ผศ. ดร.สิรภพ ตู้ประกาย

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการสร้างประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง โดยมีวัตถุประสงค์ เพื่อศึกษาและฝึกการใช้งานอุปกรณ์เซนเซอร์ควบคู่กับไมโครคอนโทรลเลอร์, ศึกษาการเขียนโปรแกรมคำสั่งควบคุม, ทำการเปิด-ปิดประตูด้วยสัญญาณ Radio Frequency Identification (RFID), Ultrasonic Sensor Module หรือสั่งงานผ่านแอปพลิเคชั่นแอดรอยด์ ซึ่งแบ่งการทำงานออกเป็น 2 ส่วน ในส่วนแรก คือ การเปิด-เปิดกลอนแม่เหล็กไฟฟ้าโดยใช้สัญญาณ Radio Frequency Identification (RFID) สำหรับฝั่งขาเข้า และใช้ Ultrasonic Sensor Module สำหรับฝั่งขาออก สำหรับการตรวจสอบการเปิด-ปิดของบานพับ จะใช้ Hall Effect Sensor ส่วนที่สอง คือ การเปิด-ปิดกลอนแม่เหล็กไฟฟ้าควบคุมผ่านอินเทอร์เน็ต โดยการสั่งงานผ่านแอปพลิเคชั่นแอนดรอยด์

ABSTRACT

This thesis is presented an automatic entrance-exit door for pets. The objective is learning and training to use a sensor with microcontroller, learn programming to command microcontroller and open-close the door with Radio Frequency Identification (RFID), ultrasonic sensor module or control by android's application. This thesis is divided into two parts: the first part is the opening and closing solenoid using RFID for the entrance and using ultrasonic sensor module for the exit. For checking the opening and closing of the hinges, use hall effect sensor. The second part is the opening and closing solenoid control via the internet, control by android's application.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันแต่ละครอบครัวนิยมเลี้ยงสัตว์เลี้ยงกันเพิ่มมากขึ้น สำหรับประตูสัตว์เลี้ยงนั้นเป็นอุปกรณ์เลี้ยงสัตว์สากลในต่างประเทศ ที่สามารถหาซื้อได้ง่ายๆ ตามร้านขายอุปกรณ์สัตว์เลี้ยง (Pet Shop) ทั่วๆไป เนื่องจากกฎหมายคุ้มครองสัตว์เลี้ยง ที่กำหนดไม่ให้ขังสัตว์เลี้ยงไว้ในบ้านเกินระยะเวลาที่กำหนด คือ 10 ชั่วโมง สำหรับในประเทศไทยนั้น ถือว่าประตูสัตว์เลี้ยงเป็นสิ่งที่แปลกใหม่พอสมควร เพราะสมัยก่อนคนไทยนิยมเลี้ยงสัตว์เลี้ยงไว้นอกบ้าน แต่ด้วยยุคสมัยที่เปลี่ยนไป ผู้คนนิยมเลี้ยงสัตว์เลี้ยงไว้ในบ้านกันมากขึ้น เพราะด้วยความใกล้ชิด และห่วงความปลอดภัยของสัตว์เลี้ยงก็ตาม จากพฤติกรรมที่เปลี่ยนไปของผู้เลี้ยง จึงเป็นที่มาสำหรับการทำปริญญานิพนธ์ชิ้นนี้ โดยประตูอัตโนมัตินี้จะแตกต่างจากอุปกรณ์ที่ขายทั่วๆ ไปตามท้องตลาด ตรงที่สามารถระบุตัวสัตว์เลี้ยงในการใช้งานประตู และสามารถสั่งการเปิด-ปิดผ่านโทรศัพท์ได้ ทำให้สามารถกำหนดได้ว่าจะเปิด-ปิดประตูได้เมื่อไร ซึ่งจะลดปัญหาได้หลากหลายอย่าง เช่น ไม่ต้องคอยลุกมาเปิด-ปิดประตูเวลาปล่อยสัตว์เลี้ยงให้ไปขับถ่ายหรือวิ่งเล่น, สามารถวางอุปกรณ์ไว้ภายนอกตัวบ้าน และหมดความกังวลว่าจะต้องขังสัตว์เลี้ยงของเราไว้ในบ้าน เวลาไม่อยู่บ้านนานๆได้

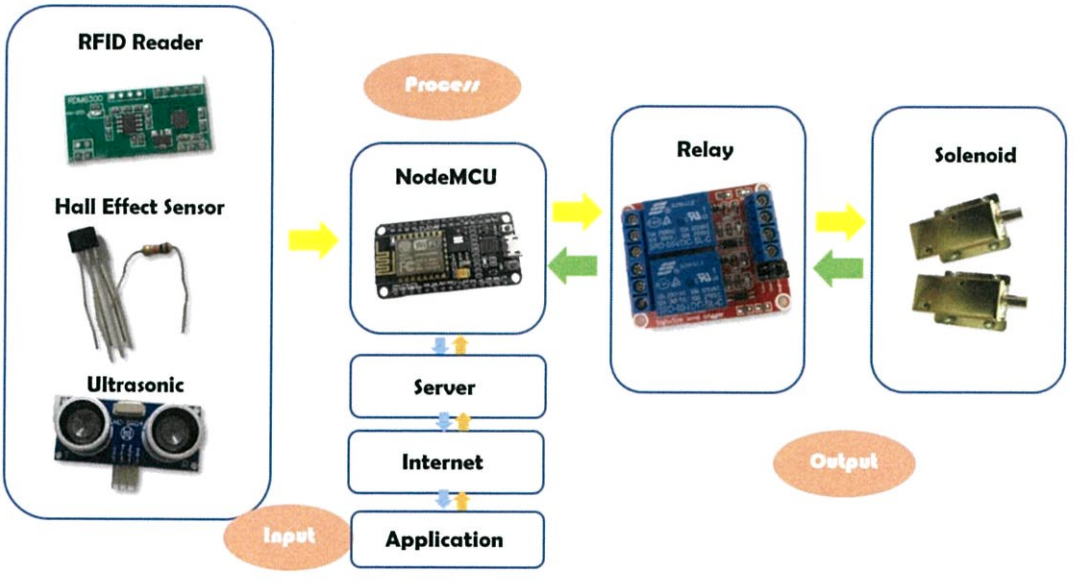
ปริญญานิพนธ์นี้เป็นการสร้างประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง ซึ่งแบ่งการทำงานออกเป็น 2 ส่วน ในส่วนแรก คือ การเปิด-เปิดกลอนแม่เหล็กไฟฟ้าโดยใช้สัญญาณ Radio Frequency Identification (RFID) สำหรับฝั่งขาเข้า และใช้ Ultrasonic Sensor Module สำหรับฝั่งขาออก ส่วนที่สอง คือ การเปิด-ปิดกลอนแม่เหล็กไฟฟ้าควบคุมผ่านอินเทอร์เน็ต โดยการสั่งงานผ่านแอปพลิเคชั่นแอนดรอยด์

1.2 วัตถุประสงค์

1) เพื่อศึกษา และฝึกการใช้งานอุปกรณ์ไมโครคอนโทรลเลอร์ควบคู่กับเซนเซอร์ชนิดต่างๆ และศึกษาการเขียนโปรแกรมคำสั่งควบคุม
2) เพื่อทำการเปิด-ปิดกลอนไฟฟ้า ด้วยการรับสัญญาณ Radio Frequency Identification (RFID) และการเขียนโปรแกรมคำสั่งควบคุม Ultrasonic Sensor Module
3) เพื่อสร้าง และออกแบบระบบประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง พร้อมควบคุมการเปิด-ปิดผ่านแอปพลิเคชั่นแอนดรอยด์

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์ชิ้นนี้คณะนี้ผู้จัดทำได้มีการนำไมโครคอนโทรลเลอร์ (Arduino) มาประยุกต์ใช้ในการทำงานร่วมกับกลอนแม่เหล็กไฟฟ้า และอุปกรณ์ต่างๆ ในการทำงานจะแบ่งเป็นฝั่งขาเข้า และฝั่งขาออก โดยฝั่งขาเข้าจะทำการรับสัญญาณ Radio Frequency Identification (RFID) จาก RFID Reader Module เพื่อทำการควบคุมการเปิด-ปิดของกลอนแม่เหล็กไฟฟ้าผ่าน Relay ในส่วนของฝั่งขาออก จะทำการใช้ Ultrasonic Sensor Module เพื่อเขียนโปรแกรมควบคุมการเปิด-ปิดของกลอนแม่เหล็กไฟฟ้าผ่าน Relay และมีการนำ Hall Effect Sensor มาใช้ทำการตรวจสอบตำแหน่งของบานพับประตู เพื่อจะได้กำหนดการเปิด-ปิดของกลอนแม่เหล็กไฟฟ้า จากนั้นจะเชื่อมต่อข้อมูลจากอุปกรณ์ไมโครคอนโทรลเลอร์ (NodeMCU) กับเซิฟเวอร์ NETPIE (Network Platform for Internet of Everything) เพื่อควบคุมผ่านอินเทอร์เน็ต โดยการสั่งงานผ่านแอปพลิเคชั่นแอนดรอยด์ แสดง Block Diagram ดังรูปที่ 1.1



รูปที่ 1.1 BLOCK DIAGRAM ของประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ปริญญานิพนธ์นี้คณะผู้จัดทำได้มีการวางแผน และศึกษาเกี่ยวกับอุปกรณ์ที่นำมาใช้กับปริญญานิพนธ์อย่างหลากหลาย อาทิเช่น ไมโครคอนโทรลเลอร์, อาร์เอฟไอดี และฮอลล์เอฟเฟกต์ เป็นต้น เพื่อใช้ออกแบบระบบเปิด-ปิดอัตโนมัติสำหรับสัตว์เลี้ยง

2.1 ความรู้พื้นฐานเกี่ยวอาร์เอฟไอดี

เทคโนโลยีอาร์เอฟไอดี (RFID : Radio Frequency Identification) ทำให้การดำเนินธุรกิจ มีประสิทธิภาพมากยิ่งขึ้น หลายองค์กรได้นำเอาเทคโนโลยีนี้ มาพัฒนาร่วมกับการใช้งานภายในองค์กรในรูปแบบต่างๆ โดยใช้ประโยชน์จากสัญญาณความถี่วิทยุโดยใช้เสาอากาศขนาดเล็กเป็นตัวเชื่อมสัญญาณระหว่างตัวส่ง และตัวรับ เราสามารถที่จะสร้างเสาอากาศขนาดเล็กมาก ซึ่งโดยปกติเสาอากาศ จะถูกฝังลงไปเป็นส่วนหนึ่งของเนื้อพลาสติกของบัตร ซึ่งมีตัวส่งกำลังเพื่อให้ทำงานได้ด้วย มีการนำเทคโนโลยีอาร์เอฟไอดี มาใช้ในบัตรต่างๆ เช่น บัตรจอดรถ, บัตรประจำตัวประชาชน, บัตรเอทีเอ็ม, บัตรสำหรับผ่านเข้าออก และฉลากสินค้า เป็นต้น

2.1.1 ความเป็นมาของเทคโนโลยีอาร์เอฟไอดี

อาร์เอฟไอดี ถูกพัฒนาในปี ค.ศ. 1970 อุปกรณ์อาร์เอฟไอดี มีการประดิษฐ์ใช้งานครั้งแรก เป็นของ Leon Theremin เพื่อนำไปใช้ในการบ่งชี้วัตถุในระยะไกล และอ่านข้อมูลจากป้าย (RFID Tag) ได้พร้อม ๆ กันหลายป้าย โดยที่เครื่องอ่านไม่ต้องสัมผัสกับตัวป้าย (RFID Tag)

เริ่มแรก การใช้อาร์เอฟไอดีในเชิงพาณิชย์ จะถูกใช้ในระบบกันขโมยสินค้า โดยที่ตัวสินค้าจะมีการติดอาร์เอฟไอดีแบบ 1 บิต (มีค่า ‘0’ หรือ ‘1’) เมื่อมีการชำระเงินค่าสินค้าเครื่องอ่านและเขียนข้อมูลอาร์เอฟไอดี จะทำการเปลี่ยนค่าบิตเป็น ‘0’ ทำให้นำสินค้าออกจากร้านได้ แต่หากมีการนำสินค้าออกจากร้านโดยที่วัถตุที่ติด RFID มีบิตเป็น ‘1’ สัญญาณกันขโมยจะดังขึ้น

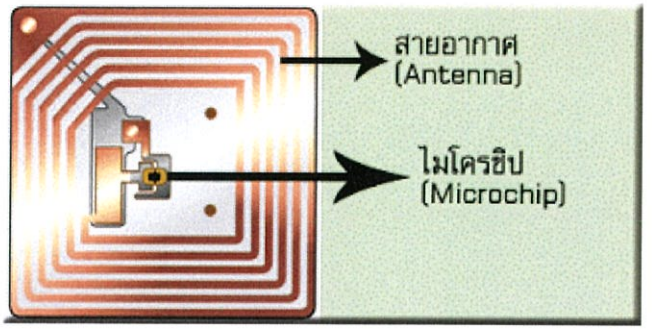
อาร์เอฟไอดีได้รับความนิยมอีกครั้ง ในปี ค.ศ. 1999 เมื่อ UCC (Uniform Code Council) EAN International บริษัท Procter & Gamble และ บริษัท Gillette ได้ร่วมก่อตั้งศูนย์ Auto-ID ขึ้นที่สถานบันเทคโนโลยีแมสซาชูเซตส์ (MIT) ประเทศสหรัฐอเมริกา เพื่อพัฒนาแนวทางการใช้อาร์เอฟไอดีสำหรับติดตามสินค้า ภายหลังได้มีการนำอาร์เอฟไอดีมาประยุกต์ใช้กับงานกันอย่างแพร่หลาย ไม่ว่าจะเป็นการเก็บค่าทางด่วนอัตโนมัติ หรือในทางด้านการเกษตร สำหรับติดที่สัตว์ เพื่อใช้เก็บข้อมูลการฉีดวัคซีนของแต่ละตัว เป็นต้น

2.1.2 ส่วนประกอบของอาร์เอฟไอดี

ในระบบอาร์เอฟไอดี จะมีองค์ประกอบหลัก ๆ อยู่ 2 ส่วนด้วยกัน

2.1.2.1 ทรานสปอนเดอร์ หรือป้าย (Transponder/RFID Tag)

ใช้สำหรับติดกับวัตถุต่างๆ โดยโครงสร้างภายในป้าย จะประกอบด้วย 2 ส่วนหลักๆ คือ สายอากาศ และไมโครชิป ที่มีการบันทึกหมายเลขหรือข้อมูลที่เกี่ยวกับวัตถุนั้นๆ



รูปที่ 2.1 โครงสร้างภายในป้าย [1]

1) ไมโครชิป (Microchip) ทำหน้าที่เก็บข้อมูลที่เกี่ยวกับวัตถุ ในหน่วยความจำ อาจเป็นแบบอ่านอย่างเดียว (ROM) ซึ่งจะเก็บข้อมูลด้วยความปลอดภัย เช่น สิทธิในการเข้าออกประตู เป็นต้น หรือทั้งอ่านและเขียน (RAM) ที่เก็บข้อมูลชั่วคราวในระหว่างการติดต่อสื่อสาร นอกจาก ROM และRAM แล้ว ยังมีหน่วยความจำแบบ EEPROM เพื่อเก็บข้อมูลการสื่อสาร และข้อมูลยังอยู่แม้จะไม่มีพลังงานไฟฟ้าเข้ามาแล้ว

2) สายอากาศ (Antenna) ขดลวดขนาดเล็ก ทำหน้าที่เป็นเสาอากาศ สำหรับรับ ส่งสัญญาณคลื่นความถี่วิทยุ และสร้างพลังงานให้ไมโครชิป โดยสายอากาศจะแผ่สัญญาณวิทยุจำนวนหนึ่งออกมากระตุ้นให้ป้ายอ่านหรือเขียนข้อมูลลงไป มีได้หลากหลายขนาดและรูปร่าง เพื่อให้เหมาะสมกับวัตถุที่จะนำป้ายไปติดตั้ง

ป้ายอาจอยู่ในรูปแบบที่เป็นกระดาษ แผ่นฟิล์ม พลาสติก ที่มีขนาดและรูปร่างต่างๆ แต่จะอยู่ในรูปแบบใด สามารถแบ่งประเภทของป้ายได้ 2 ชนิด ได้แก่ แบบแพสซีฟ (Passive Tag) และแบบแอ็กทีฟ (Active Tag) โดยแต่ละชนิดจะแตกต่างกันตามรูปแบบการนำไปใช้งาน โครงสร้างภายใน และหลักการทำงาน ดังนี้

1) Passive Tag ไม่มีแหล่งพลังงาน หรือแบตเตอรี่ภายในป้าย เนื่องจาก อาศัยพลังงานไฟฟ้าที่เกิดจากการเหนี่ยวนำคลื่นแม่เหล็กไฟฟ้าจาก Reader ที่มีวงจรกำเนิดไฟฟ้าขนาดเล็กอยู่ในตัว เรียกว่า Transceiver โครงสร้างภายในป้ายชนิดนี้ ประกอบด้วยส่วนการควบคุมการทำงานของภาครับส่งสัญญาณวิทยุ (Analog Front-End), ส่วนควบคุมภาคลอจิก (Digital Control Unit) และส่วนของหน่วยความจำ (Memory)

2) Active Tag จะมีแบตเตอรี่อยู่ภายใน ซึ่งใช้เป็นแหล่งจ่ายไฟขนาดเล็ก เพื่อป้อนพลังงานไฟฟ้าให้ป้ายทำงาน จึงทำให้ป้ายมีอายุการใช้งานจำกัด เมื่อแบตเตอรี่หมดจะไม่สามารถนำป้ายมาใช้งานได้อีก แต่สามารถออกแบบวงจรของป้ายให้มีอายุการใช้งานนานขึ้น โดยจะต้องใช้กระแสไฟน้อยๆ

ตารางที่ 2.1 เปรียบเทียบข้อดี ข้อเสียของป้ายสองชนิด

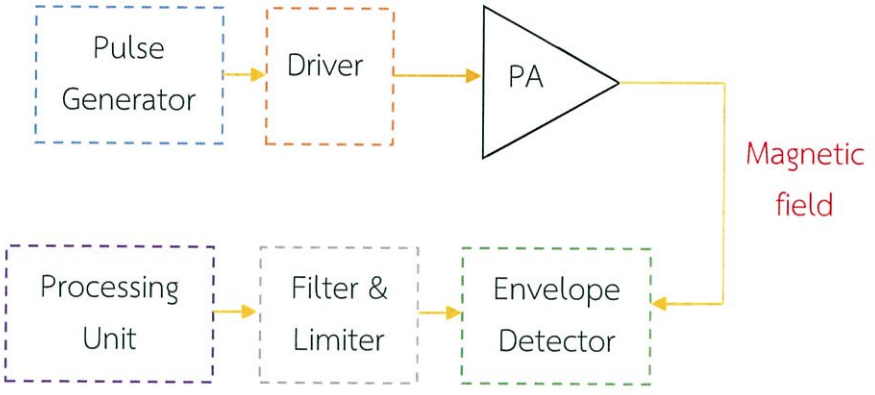
	ข้อดี	ข้อเสีย
Passive Tag	น้ำหนักเบา	ระยะการรับส่งข้อมูลสั้น (ระยะไกลสุดเพียง 1.5 เมตร)
	Tag มีขนาดเล็ก	หน่วยความจำมีขนาดเล็ก (ประมาณ 32 ถึง 128 บิต)
	ราคาถูก	Reader ต้องมีกำลังส่งที่สูง
	อายุการใช้งานไม่จำกัด	อาจเกิดผิดพลาดหากทำงานในบริเวณที่มีสัญญาณรบกวน
Active Tag	มีหน่วยความจำขนาดใหญ่ (ประมาณ1เมกะไบต์)	ราคาสูง
	ระยะการรับส่งข้อมูลไกล (ระยะไกลสุด 6 เมตร)	ระยะเวลาในการทำงานที่จำกัด ตามอายุของแบตเตอร์รี่ประมาณ 3–7 ปี
	ทำงานบริเวณสัญญาณรบกวนได้ดี	Tag มีขนาดใหญ่

นอกจากการแบ่งป้ายตามชนิด ยังสามารถแบ่งได้ตามประเภทรูปแบบในการใช้งานได้ 3 แบบ คือ แบบที่สามารถถูกอ่านและเขียนข้อมูลได้อย่างอิสระ (Read-write), แบบเขียนได้เพียงครั้งเดียวเท่านั้นแต่อ่านได้อย่างอิสระ (Write-One, Read-Many : WORM) และแบบอ่านได้เพียงอย่างเดียว (Read-Only)

2.1.2.2 เครื่องสำหรับอ่าน/เขียนข้อมูลในป้าย (Interrogator / Reader)

การอ่านหรือเขียนข้อมูล จะกระทำผ่านคลื่นความถี่วิทยุ สามารถอ่านหรือรหัสข้อมูลได้โดยไม่ต้องเห็นป้าย ไม่จำเป็นที่เครื่องอ่านเขียนและป้าย จะต้องอยู่ในแนวเส้นตรงกับคลื่นความถี่วิทยุเพียงแค่อยู่ในบริเวณที่สามารถรับคลื่นความถี่วิทยุได้ เครื่องอ่านเขียนก็สามารถทำงานได้ โดยมีหน้าที่หลัก คือการเชื่อมต่อกับป้ายเพื่อทำการอ่านหรือเขียนข้อมูล แล้วทำการตรวจสอบความผิดพลาดของข้อมูล และถอดรหัสสัญญาณข้อมูลที่ได้รับ โครงสร้างภายในเครื่องอ่านเขียนข้อมูลประกอบด้วย ภาครับและส่งสัญญาณวิทยุ (Transceiver), ภาคสร้างสัญญาณพาหะ (Carrier),

ขดลวดที่ทำหน้าที่เป็นสายอากาศ (Antenna), วงจรจูนสัญญาณ (Tuner) และหน่วยประมวลผลข้อมูล (Processing Unit) ดังรูปที่ 2.2

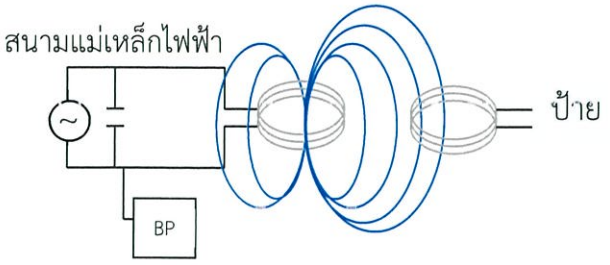


รูปที่ 2.2 Block Diagram โครงสร้างภายในเครื่องอ่าน เขียนข้อมูล

2.1.3 การทำงานของระบบอาร์เอฟไอดี

การทำงานของระบบอาร์เอฟไอดีที่สำคัญ การบรรจุอุปกรณ์ และวงจรอิเล็กทรอนิกส์กับโลหะที่ยืดหยุ่นได้ สำหรับการติดตามหรือทำหน้าที่เป็นเสาอากาศคือ "Inlay" ซึ่งมีความหนาสูงสุดอยู่ที่ 0.375 มิลลิเมตร ทำเป็นแผ่นบางอัดเป็นชั้นๆ ระหว่างกระดาษ, แผ่นฟิล์ม หรือพลาสติก การที่ Inlay มีลักษณะรูปร่างที่บางทำให้ง่ายต่อการติดป้ายที่ตัววัตถุ

ในส่วนของระบบ จะนำเอาคลื่นวิทยุมาเป็นคลื่นพาหะ เพื่อใช้ในการสื่อสารข้อมูลระหว่างป้าย และเครื่องอ่าน เขียนข้อมูล ซึ่งเป็นการสื่อสารแบบไร้สาย (Wireless) ทำการมอดูเลต (Modulation) ข้อมูลที่ต้องการส่งกับคลื่นวิทยุแล้วส่งออก ผ่านทางสายอากาศที่เครื่องรับข้อมูล หลักการทำงานเบื้องต้นของอาร์เอฟไอดี เริ่มจากเครื่องอ่าน เขียนข้อมูลจะปล่อยคลื่นแม่เหล็กไฟฟ้าคอยตรวจจับว่าพบป้ายในบริเวณสนามแม่เหล็กไฟฟ้าที่ปล่อยมา หรือเกิดการมอดูเลตสัญญาณ ถ้ามีป้ายอยู่ในบริเวณสนามแม่เหล็กไฟฟ้า ป้ายจะได้รับพลังงานไฟฟ้าที่เกิดจากการเหนี่ยวนำของคลื่นแม่เหล็กไฟฟ้า เพื่อให้ป้ายเริ่มทำงาน และส่งข้อมูลในหน่วยความจำที่ผ่านการมอดูเลตกับคลื่นพาห์ออกมาทางสายอากาศที่อยู่ภายในป้าย จากนั้นคลื่นพาห์ที่ถูกส่งออกมาจากป้าย จะเกิดการเปลี่ยนแปลงแอมปลิจูด, ความถี่ หรือเฟส ขึ้นอยู่กับวิธีการมอดูเลต สุดท้ายเครื่องอ่านเขียนข้อมูลจะตรวจจับความเปลี่ยนแปลงของคลื่นพาห์แปลงออกมาเป็นข้อมูล แล้วทำการถอดรหัสเพื่อนำข้อมูลไปใช้งานต่อไป อธิบายโดยสรุปได้ดังรูปที่ 2.3

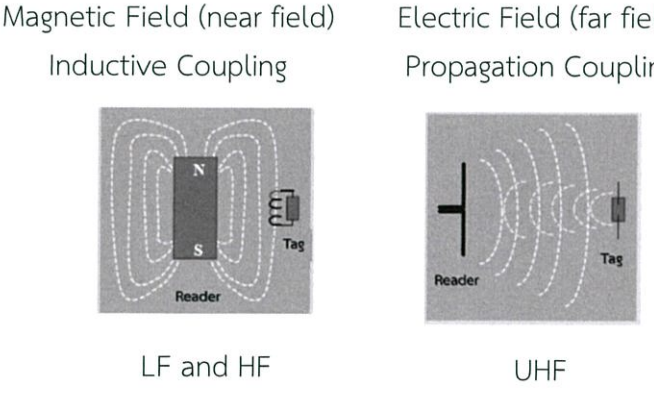


รูปที่ 2.3 การทำงานของระบบอาร์เอฟไอดี

2.1.3.1 หลักการทำงานของป้ายแบบแพ็สซีฟ (Passive Tag)

ในย่านความถี่ต่ำ (Low Frequency : LF) และความถี่สูง (High Frequency : HF) จะใช้หลักการคู่ควบแบบเหนี่ยวนำ (Inductive coupling) ซึ่งเกิดจากการอยู่ใกล้กันของขดลวดจากเครื่องอ่านที่กำลังทำงานและสายอากาศของป้าย ทำให้เกิดการถ่ายเทพลังงานจากเครื่องอ่านไปยังป้าย ผ่านสนามแม่เหล็กไฟฟ้าที่เกิดขึ้น เมื่อไมโครชิปได้รับพลังงานก็จะทำงานตามที่ได้ตั้งค่าไว้ โดยเครื่องอ่านจะรับรู้ได้จากสนามแม่เหล็กที่ส่งมาจากป้าย

ส่วนในย่านความถี่สูงยิ่ง (Ultra High Frequency : UHF) จะใช้หลักการคู่ควบแบบแผ่กระจาย (Propagation coupling) โดยที่สายอากาศของเครื่องอ่านจะทำการส่งพลังงานแม่เหล็กไฟฟ้า ในรูปคลื่นวิทยุออกมา เมื่อป้ายได้รับสัญญาณผ่านสายอากาศ จะสะท้อนกลับคลื่นที่ถูกปรับค่าตามรหัสประจำตัว ไปยังเครื่องอ่าน (backscattering)



รูปที่ 2.4 หลักการทำงานของ LF, HF และUHF [1]

จากรูปที่ 2.4 จะเป็นหลักการทำงานของป้ายแบบแพ็สซีฟ (Passive Tag) ในสามย่านความถี่ ซึ่งแบ่งเป็นย่านความถี่ต่ำ (Low Frequency : LF), ความถี่สูง (High Frequency : HF) ใช้หลักการควบคู่เดียวกัน และย่านความถี่สูงยิ่ง (Ultra High Frequency : UHF) จะใช้อีกแบบ

2.1.3.2 หลักการทำงานของป้ายแบบแอ็กทีฟ (Active Tag)

ป้ายแบบแอ็กทีฟจะทำการส่งข้อมูลก็ต่อเมื่อได้รับสัญญาณจากเครื่องอ่าน เขียนข้อมูล และเครื่องบอกตำแหน่ง หรือเบคอน (beacon) ซึ่งสัญญาณจะถูกปล่อยออกมาเป็นระยะๆ ตลอดเวลา การส่งข้อมูลของอาร์เอฟไอดี สามารถเข้ารหัสข้อมูลและมอดูเลชั่นได้ทั้งแบบ ASK, PSK, FSK รูปแบบการส่งข้อมูล แบบ Full Duplex, Half Duplex และSequential มีระบบการใช้งานได้พร้อมกัน แบบ TDMA, FDMA, CDMA และSDMA

2.1.4 มาตรฐานของอาร์เอฟไอดี

มาตรฐานระหว่างประเทศสำหรับการใช้งาน RFID มี 2 หน่วยงานหลัก ได้แก่ International Organization of Standard (ISO) และEPC Global

2.1.4.1 การกำหนดมาตรฐานอาร์เอฟไอดี

โดยที่มาตรฐานของ RFID มีการกำหนดไว้ 4 ด้านคือ

1. มาตรฐานด้านเทคโนโลยี (Technology)
2. มาตรฐานรูปแบบของข้อมูล (Data format)
3. มาตรฐานวิธีการทดสอบ (Conformance)
4. มาตรฐานการใช้งาน (Applications)

2.1.4.2 คลื่นความถี่ที่ใช้งานกับ RFID

ในปัจจุบันคลื่นพาห์ที่ใช้งานในระบบอาร์เอฟไอดี จะอยู่ในย่านความถี่พลเรือน ISM (Industrial-Scientific-Medical) ซึ่งเป็นย่านความถี่ที่กำหนดในการใช้งานในเชิงการแพทย์วิทยาศาสตร์ และอุตสาหกรรม สามารถใช้งานได้โดยไม่ตรงกับย่านความถี่ที่ใช้งานในการสื่อสารโดยทั่วไป โดยมี 4 ย่านความถี่ใช้งาน อาจแบ่งออกได้เป็น 4 ย่านใหญ่ๆ ได้แก่ ย่านความถี่ต่ำ (Low Frequency : LF) ต่ำกว่า 150 กิโลเฮิรตซ์ (kHz), ย่านความถี่สูง (High Frequency : HF) 13.56/27.125 เมกะเฮิรตซ์ (MHz), ย่านความถี่สูงยิ่ง (Ultra High Frequency : UHF) 433/868/915 เมกะเฮิรตซ์ (MHz) และย่านความถี่ไมโครเวฟ (Microwave frequency) 2.45/5.8 กิกะเฮิรตซ์ (GHz) เปรียบเทียบคลื่นความถี่ที่ใช้งาน ในแต่ละย่านความถี่ในด้านของระยะการอ่าน สามารถสรุปได้ ดังตารางที่ 2.2

ตารางที่ 2.2 คลื่นความถี่ที่ใช้งานกับระยะที่อ่านได้

ความถี่	ระยะที่อ่านได้
125 – 134 กิโลเฮิรตซ์ (kHz)	น้อยกว่า 1 เมตร (10 เซนติเมตร)
13.56 เมกะเฮิรตซ์ (MHz)	น้อยกว่า 1.5 เมตร (~1 เมตร)
860 – 960 เมกะเฮิรตซ์ (MHz)	2 – 5 เมตร 1 – 100 เมตร (ป้ายแบบแอ็กทีฟ)
2.45 กิกะเฮิรตซ์ (GHz)	น้อยกว่า 1 เมตร (ป้ายแบบแพ็สซีฟ) 1 – 15 เมตร (ป้ายแบบแอ็กทีฟ)

2.1.4.3 อัตราการรับส่งข้อมูลและแบนด์วิดธ์

อัตราการรับส่งข้อมูล (Data Transfer Rate) จะขึ้นอยู่กับความถี่ของคลื่นพาห์ ความถี่ของคลื่นพาห์ยิ่งสูง อัตราการรับส่งข้อมูลก็จะยิ่งสูงตามไปด้วย ส่วนการเลือกแบนด์วิดธ์ หรือย่านความถี่นั้นก็จะมีผลต่ออัตราการรับส่งข้อมูลเช่นกันโดยมีหลักว่า แบนด์วิดธ์ควรจะมีค่ามากกว่าอัตราการรับส่งข้อมูลที่ต้องการอย่างน้อยสองเท่า แต่การใช้แบนด์วิดธ์ที่กว้างเกินไปก็อาจทำให้เกิดปัญหาเกี่ยวกับสัญญาณรบกวนมาก หรือทำให้ S/N Ratio ต่ำลง ดังนั้นการเลือกใช้แบนด์วิดธ์ให้ถูกต้องก็เป็นส่วนสำคัญในการพิจารณา

2.1.4.4 ระยะการรับส่งข้อมูลและกำลังส่ง

ระยะการรับส่งข้อมูลในระบบอาร์เอฟไอดี ขึ้นอยู่กับปัจจัยสำคัญต่างๆ ได้แก่ กำลังส่งของเครื่องอ่านเขียนข้อมูล (Reader/Interrogator Power), กำลังส่งของป้าย (Tag Power), สภาพแวดล้อม และการออกแบบสายอากาศของตัวอ่านข้อมูล

2.2 รีเลย์

รีเลย์ (Relay) เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่คล้ายกับสวิตช์ โดยจะเป็นสวิตช์ที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานแม่เหล็ก ใช้ในการเปลี่ยนสภาวะทำงาน โดยการป้อนกระแสไฟฟ้าตามที่กำหนดให้กับขดลวด ทำให้หน้าสัมผัสติดกันเป็นวงจรปิด และทันทีที่ไม่ได้ป้อนกระแสไฟฟ้าให้กับรีเลย์จะกลายเป็นวงจรเปิด สามารถนำรีเลย์ไปประยุกต์ใช้ในการควบคุมวงจรตามที่ต้องการ เช่น การกด, การดึง, การเลื่อน และการเปิด-ปิดกลอนไฟฟ้า เป็นต้น

กระแสไฟฟ้าที่ใช้ป้อนให้กับตัวรีเลย์ จะเป็นกระแสไฟฟ้าที่มาจากแหล่งกำเนิดต่างๆ สำหรับรีเลย์ จะมีด้วยกันสองชนิด ได้แก่ รีเลย์ไฟฟ้ากระแสสลับ โดยแรงดันไฟป้อนขดลวดจะเป็นแรงดันไฟ AC เช่น 220 โวลต์ และ110 โวลต์ เป็นต้น อย่างที่สองคือรีเลย์ไฟฟ้ากระแสตรง แรงดันไฟป้อนขดลวดจะเป็นแรงดันไฟ DC เช่น 3 โวลต์, 5 โวลต์, 6 โวลต์, 9 โวลต์, 12 โวลต์, 24 โวลต์, 32 โวลต์ และ48 โวลต์ เป็นต้น

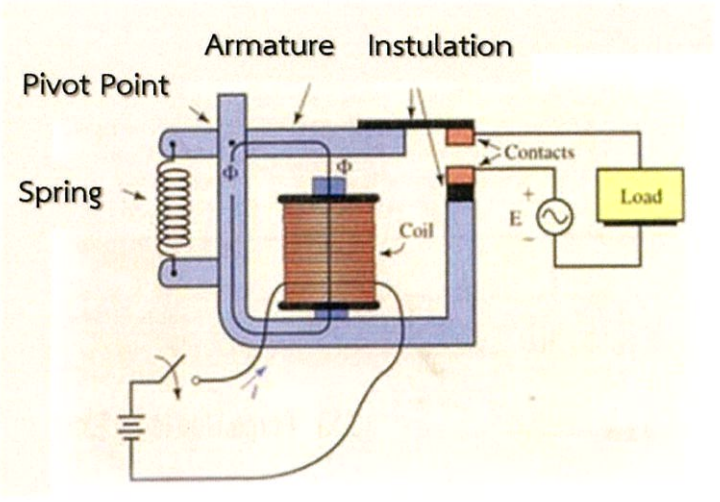
โดยสรุปแล้วรีเลย์เป็นอุปกรณ์ไฟฟ้าชนิดหนึ่ง ที่ใช้ตรวจสอบสภาพการณ์ของทุกส่วนในระบบกำลังไฟฟ้าอยู่ตลอดเวลาหากระบบมีการทำงานที่ผิดปกติ รีเลย์จะเป็นตัวสั่งการให้ตัดส่วนที่ลัดวงจรหรือส่วนที่ทำงานผิดปกติ ออกจากระบบทันทีโดยเซอร์กิตเบรกเกอร์จะเป็นตัวที่ตัดส่วนที่เกิดความผิดพลาดออกจากระบบจริงๆ

2.2.1 ส่วนประกอบของรีเลย์

รีเลย์ ประกอบด้วยส่วนสำคัญ 2 ส่วนหลักด้วยกัน ได้แก่

2.2.1.1 ส่วนของขดลวด (Coil)

โดยทั่วไปจะเป็นขดลวดพันรอบแกนเหล็ก เป็นส่วนที่ใช้ควบคุมการทำงานของรีเลย์ให้เปิด (Open) หรือปิด (Closed) โดยที่วงจรปิดคือการมีกระแสไหลผ่าน ทำงานโดยการรับแรงดันจากภายนอกต่อคร่อมที่ขดลวดเหนี่ยวนำนี้ ส่วนของขดลวดจะเหนี่ยวนำกระแสต่ำ ทำหน้าให้เกิดสนามแม่เหล็กไฟฟ้า เกิดแรงดูดโลหะอ่อนที่เรียกว่า อาร์เมเจอร์ (Armature) ที่ปลายของอาร์เมเจอร์แต่ละด้าน ด้านหนึ่งมักยึดติดกับสปริง และปลายอีกด้านหนึ่งจะยึดติดกับหน้าสัมผัส (Contact) การเคลื่อนที่ของอาร์เมเจอร์จากสนามแม่เหล็กไฟฟ้านั้นจะเป็นการควบคุมการเคลื่อนที่ของหน้าสัมผัสให้แยกออกหรือดึงเข้ากับหน้าสัมผัสอีกตัวนึง ซึ่งยึดอยู่กับที่ เราสามารถนำหลักการนี้ไปควบคุมโหลดได้

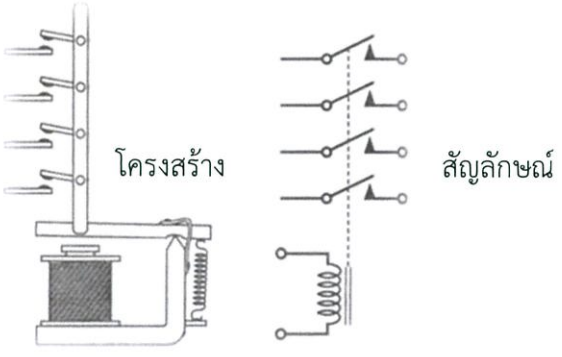


รูปที่ 2.5 การเชื่อมต่อของขดลวด [2]

จากรูปที่ 2.5 การทำงานและการเชื่อมต่อของขดลวด จะเห็นได้ว่า ตัวขดลวดเหนี่ยวนั้นจะเชื่อมต่ออยู่กับอาร์เมเจอร์ ซึ่งเป็นโลหะเชื่อมต่อกับสปริง และหน้าสัมผัส ซึ่งสามารถทำหน้าที่ควบคุมโหลด

2.2.1.2 ส่วนของหน้าสัมผัส (Contact)

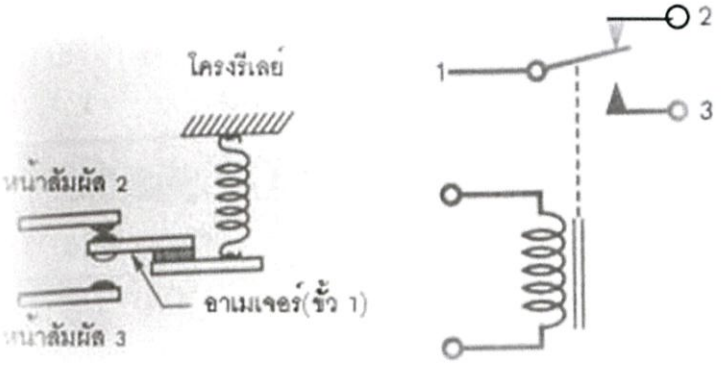
ทำหน้าที่เหมือนสวิตช์จ่ายกระแสไฟให้กับอุปกรณ์หรือโหลด ให้ทำงานตามที่ต้องการ ในตอนแรกรีเลย์มีหน้าสัมผัสเพียงชุดเดียว คือหนึ่งคู่หน้าสัมผัส แต่ในปัจจุบันรีเลย์ที่มีขดลวดชุดเดียวสามารถควบคุมหน้าสัมผัสได้หลายชุดโดยอาร์เมเจอร์ ซึ่งหนึ่งอาร์เมเจอร์จะยึดกับหน้าสัมผัส ที่เคลื่อนที่ได้ 4 ชุด รีเลย์ดังกล่าวจึงสามารถควบคุมการทำงานออกกรือเข้าหากันได้ถึง 4 ชุด แต่ละหน้าสัมผัสที่เคลื่อนที่ได้ เรียกว่าขั้ว (Pole)



รูปที่ 2.6 โครงสร้างและสัญลักษณ์ของชุดหน้าสัมผัสแบบ 4PST [2]

จากรูปที่ 2.6 ในรีเลย์นี้ มีหน้าสัมผัส 4 ชุด และแต่ละหน้าสัมผัสจะมีขั้วที่ทำให้หน้าสัมผัสนั้นเคลื่อนที่ได้มีทั้งหมด 4 ขั้ว จึงเรียกหน้าสัมผัสเช่นนี้ว่า ชุดหน้าสัมผัสแบบ 4PST (Four

Pole Single Throw) แต่อีกแบบหนึ่งคือถ้ามีขั้วที่เคลื่อนที่แยกออกจากหน้าสัมผัสอันใดอันหนึ่งสลับไปมา คล้ายกับการโยกสวิตช์ เลือกใช้เพียง 1 หน้าสัมผัสจาก 2 หน้าสัมผัส จะเรียกหน้าสัมผัสแบบนี้ว่า ชุดหน้าสัมผัสแบบ SPDT (Single Pole Double Throw) ดังรูปที่ 2.7 ส่วนรูปแบบอื่นๆ มาจากการแยกประเภทตามจำนวนของขั้ว และจำนวน Throw ซึ่งจำนวนของขั้ว (SP-Single Pole, DP-Double Pole, 3P-Triple Pole, etc.) บอกถึงจำนวนวงจรที่ทำการเปิด-ปิด หรือจำนวนของขา คอมมอนนั่นเอง และจำนวน Throw (ST,DT) จะบอกถึงจำนวนของตัวเลือกของขั้ว ตัวอย่างเช่น SPST (Single Pole Single Throw) สวิตช์จะสามารถเลือกได้เพียงอย่างเดียวโดยจะเป็นแบบปกติเปิด (NO-Normally Open) หรือแบบปกติปิด (NC-Normally Close) แต่ถ้าเป็น SPDT (Single Pole Double Throw) สวิตช์จะมีหนึ่งคู่เป็นแบบปกติเปิดและอีกหนึ่งคู่เป็นแบบปกติปิดเสมอ



รูปที่ 2.7 ชุดหน้าสัมผัสแบบ SPDT [2]

ซึ่งในชุดหน้าสัมผัส 1 ชุด จะประกอบไปด้วย

1) ขา C หรือ ขาคอมมอน (Common) เป็นขาที่ถูกใช้งานร่วม ต่อกันระหว่างหน้าสัมผัส NC และ หน้าสัมผัส NO ขึ้นอยู่กับว่าขณะนั้นมีกระแสไฟฟ้าไหลผ่านขดลวดเหนี่ยวนำหรือไม่ หน้าสัมผัสในรีเลย์ 1 ตัวอาจมีมากกว่า 1 ชุด ขึ้นอยู่กับผู้ผลิตและลักษณะของงานที่ถูกนำไปใช้

2) หน้าสัมผัส NO (Normally opened หรือปกติเปิด) หน้าสัมผัสแบบปกติเปิด โดยในสภาวะปกติจะลอยอยู่แบบเปิดไว้ ไม่ถูกต่อกับขาคอมมอน แต่จะเชื่อมต่อกันและทำงาน ต่อเมื่อป้อนกระแสไฟฟ้าให้กระแสไฟฟ้าไหลผ่านขดลวด

3) หน้าสัมผัส NC (Normally closed หรือปกติปิด) หน้าสัมผัสแบบปกติปิดโดยในสภาวะปกติ ในกรณีที่ไม่ได้จ่ายกระแสะไฟฟ้าให้ขดลวดเหนี่ยวนำ หน้าสัมผัส NC จะเชื่อมต่อเข้ากับขาคอมมอน นิยมใช้ต่อกับโหลดหรือเครื่องใช้ไฟฟ้าเพื่อให้ทำงานตลอดเวลา และจะลอยหรือไม่สัมผัสกันเมื่อมีกระแสไฟฟ้าไหลผ่านขดลวด

รูปที่ 2.8 การเชื่อมรีเลย์

จากรูปที่ 2.8 เป็นการใช้รีเลย์แบบ เชื่อม 2 ขาและ 3 ขา โดยแสดงผ่านสัญลักษณ์ และขาที่มีการเชื่อมต่อกันระหว่างหน้าสัมผัส NC, หน้าสัมผัส NO และขาคอมมอน ซึ่งรูปแบบชุดหน้าสัมผัส จะมีทั้งหมด 3 แบบ

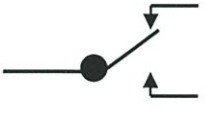
1. หน้าสัมผัสแบบ A (Form A) หมายถึง หน้าสัมผัสของรีเลย์ ในสภาพปกติเปิด (NO) และหน้าสัมผัสเป็นแบบ SPST (Single Pole Single Throw) ถ้าจะเขียนเป็นสัญลักษณ์ได้ ดังรูปที่ 2.9

รูปที่ 2.9 หน้าสัมผัสแบบ A

2. หน้าสัมผัสแบบ B (Form B) หมายถึง หน้าสัมผัสของ Relay ในสภาพปกติจะปิด (NC) และหน้าสัมผัสเป็นแบบ SPST (Single Pole Single Throw) เขียนเป็นสัญลักษณ์ได้ ดังรูปที่ 2.10

รูปที่ 2.10 หน้าสัมผัสแบบ B

3. หน้าสัมผัสแบบ C (Form C) กรณีแบบนี้เรียกว่า break, make หรือ transfer เป็นหน้าสัมผัสแบบ SPDT (Single Pole Double Throw) หน้าสัมผัสแบบ C จะมีอยู่ด้วยกัน 3 ขา ในขณะที่รีเลย์ยังไม่ทำงาน หน้าสัมผัส 1 และ2 จะต่อกันอยู่ เมื่อรีเลย์ทำงานหน้าสัมผัส 1 และ2 จะแยกกัน จากนั้นหน้าสัมผัส 1 จะมาต่อกับหน้าสัมผัส 3 แทน พอถึงเวลาที่รีเลย์หยุดทำงาน หน้าสัมผัส 1 กับ2 ก็จะกลับมาต่อกันตามเดิม เขียนเป็นสัญลักษณ์ได้ ดังรูปที่ 2.11



รูปที่ 2.11 หน้าสัมผัสแบบ C

2.2.2 ชนิดของรีเลย์

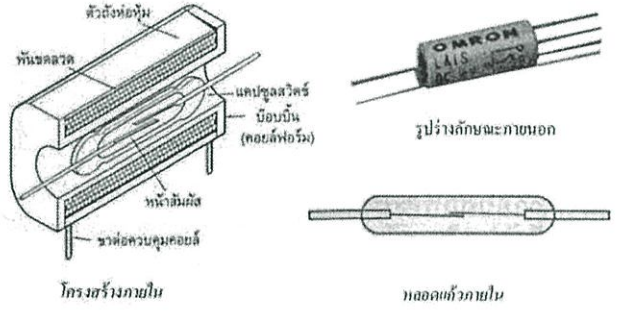
รีเลย์นั้นมีอยู่ด้วยกันหลายชนิดและมีลักษณะการทำงานที่คล้ายคลึงกัน จึงควรทำความรู้จักว่าแต่ละประเภทมีความเหมาะสมกับงานชนิดใด เพื่อเลือกใช้ได้อย่างถูกต้อง ชนิดของรีเลย์ที่นิยมใช้งานและรู้จักกันแพร่หลาย มีอยู่ด้วยกัน 4 ชนิด

2.2.2.1 อาร์เมเจอร์รีเลย์ (Armature Relay)

คือรีเลย์ที่นิยมใช้กันมากที่สุด บางครั้งเรียกรีเลย์แบบนี้ว่า รีเลย์ชนิดแคลปเปอร์ (Clapper Relay) จะถูกใช้ในไทม์เมอร์รีเลย์ (Timer Relay) และเจนเนอรัล รีเลย์ (General Relay)

2.2.2.2 รีดรีเลย์ (Reed Relay)

เป็นรีเลย์ไฟฟ้าที่มีลักษณะเป็นแคปซูลขนาดเล็ก ดังรูปที่ 2.12 แสดงภาพตัดขวางของรีเลย์ ที่ประกอบด้วยส่วนที่เรียกว่ารีดแคปซูล ซึ่งมีขดลวดเหนี่ยวนำพันบนแกนบ๊อบบิ้น รีดแคปซูลจะเป็นหลอดแก้ว ภายในบรรจุก๊าซเฉื่อย หน้าสัมผัสเป็นโลหะผสมแผ่นบางๆ ปลายตัด 2 แผ่น วางซ้อนแต่ไม่สัมผัสกัน เป็นสวิตช์ชุดเดียวทางเดียวหน้าสัมผัสปกติเปิดวงจร (SPST-NO)



รูปที่ 2.12 รีดรีเลย์ [2]

2.2.2.3 รีดสวิตช์ (Reed Switch)

Magnetic Sensors ที่มีลักษณะเป็นแบบหน้าสัมผัสซึ่งโดยปกติทั่วไปแล้ว จะเป็นหน้าสัมผัสแบบปกติเปิด (Normally Open : NO) สวิตช์นี้จะทำงานโดยอาศัยสนามแม่เหล็ก ซึ่งอาจจะเป็นแม่เหล็กถาวร หรือแม่เหล็กไฟฟ้า โดยแผ่นหน้าสัมผัสจะทำมาจากสารที่มีผลต่อสนามแม่เหล็ก (ferromagnetic) และ ติดตั้งอยู่ภายในกระเปาะแก้วเล็กๆที่มีการเติมก๊าซเฉื่อย เพื่อทำให้การตัดต่อการส่งกระแสไฟฟ้าได้เร็วยิ่งขึ้น

2.2.2.4 โซลิดสเตตรีเลย์ (Solid-State Relay)

ทำงานด้วยสัญญาณจากไมโครคอนโทรลเลอร์ เพื่อไปสั่งอุปกรณ์หรือโหลดที่มีกำลังไฟฟ้าสูงกว่าตัวควบคุม มีขั้วต่ออย่างละ 2 ขั้ว โดยเป็นขั้วสำหรับป้อนสัญญาณควบคุม เพื่อบังคับให้

ขั้วเอาต์พุตปิด หรือเปิดวงจร โดยจะมีการแยกกันทางไฟฟ้าระหว่างขั้วอินพุตและเอาต์พุต ทำงานได้ไว (Response time ต่ำ) และไม่มีเสียงดังรบกวนขณะเปิด-ปิดหน้าสัมผัส ใช้พลังงานน้อยกว่า ส่วนข้อเสียคือแพง เป็นรีเลย์ที่ไม่มีโครงสร้างทางกลอยู่ภายใน

2.2.3 ประโยชน์ของรีเลย์

ดังที่กล่าวมาว่ารีเลย์สามารถควบคุมการทำงานของอุปกรณ์อุปกรณ์ได้อย่างมากมายหลายกรณี จะอธิบายเป็นข้อๆ ดังนี้

1. ทำให้ระบบส่งกำลังมีเสถียรภาพ (Stability) ที่สูง โดยรีเลย์จะสั่งตัดวงจรเฉพาะส่วนที่สถานการณ์ผิดปกติหรือลัดวงจรออกจากระบบ ซึ่งจะเป็นการลดความเสียหายให้แก่ระบบน้อยที่สุด
2. ลดค่าใช้จ่ายในการซ่อมแซมโดยซ่อมเฉพาะส่วนที่เกิดสถานการณ์ผิดปกติ
3. ลดความเสียหายไม่เกิดลุกลามไปยังอุปกรณ์อื่นๆ
4. ทำให้ระบบไฟฟ้าไม่ดับทั้งระบบ เมื่อเกิดฟอลต์ขึ้นในระบบ เนื่องจากตัวรีเลย์จะตัดการลัดวงจรออกไป
5. หน้าสัมผัสที่แนบสนิทจะทำให้การไหลของกระแสไฟฟ้าราบรื่น

2.2.4 คุณสมบัติของรีเลย์

การจะนำรีเลย์ไปใช้นั้นควรจะรู้หลักการทำงานและคุณสมบัติของรีเลย์ เพื่อเลือกใช้ได้อย่างถูกต้อง และสามารถทราบได้ว่า รีเลย์นั้นทำอะไรได้บ้าง และควรนำไปใช้อย่างไร

1. ความว่องไว (Sensitivity) คือมีความสามารถในการตรวจพบ สิ่งที่มีความผิดปกติเพียงเล็กน้อยได้

2. ความเร็วในการทำงาน (Speed) คือความสามารถทำงานได้รวดเร็ว ไม่ทำให้เกิดความเสียหายแก่อุปกรณ์และไม่กระทบกระเทือนต่อระบบ โดยทั่วไปแล้วเวลาที่ใช้ในการตัดวงจรจะขึ้นอยู่กับระดับของแรงดันของระบบด้วยดังนี้ ระบบ 6-10 เควี จะต้องตัดวงจรออกจากระบบภายในเวลา 1.5-3.0 วินาที, ระบบ 100-220 เควี จะต้องตัดวงจรออกจากระบบภายในเวลา 0.15-0.3 วินาที และระบบ 300-500 เควี จะต้องตัดวงจรออกจากระบบภายในเวลา 0.1-0.12 วินาที

2.2.5 การประยุกต์ใช้รีเลย์

จากคุณสมบัติของรีเลย์ จะเห็นได้ว่ามีการทำงานที่สามารถควบคุมอุปกรณ์ที่ต้องการให้ทำหน้าที่คล้ายกับสวิตช์ สามารถอาศัยคุณสมบัตินี้ไปประยุกต์ใช้งานจริงได้ ในหัวข้อนี้จะกล่าวถึงวิธีการนำรีเลย์โมดูล ไปประยุกต์ใช้งานจริง

2.2.5.1 ข้อมูลบนรีเลย์

รีเลย์ที่มีใช้ในปัจจุบันจะมีลักษณะต่างกันออกไป แต่ในที่นี้จะยกตัวอย่างมาตราฐานที่ควรจะมีข้อมูลบนรีเลย์ เพื่อสะดวกต่อการอ่าน แสดงดังรูปที่ 2.13

รูปที่ 2.13 การอ่านข้อมูลบนรีเลย์

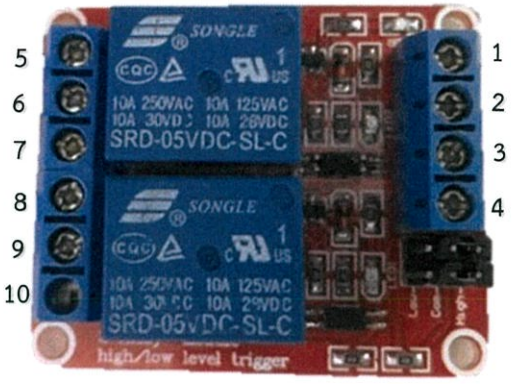
1. ยี่ห้อ รุ่นของผู้ผลิต (แบรนด์) รวมถึงสัญลักษณ์มาตรฐานต่างๆ
2. รายละเอียดของไฟฟ้ากระแสสลับที่รองรับการทำงานได้ (VAC)
3. รายละเอียดของไฟฟ้ากระแสตรงที่รองรับการทำงานได้ (VDC)
4. โมเดลระดับแรงดันฝั่งขดลวด ชนิดและโครงสร้าง และข้อมูลด้าน Coil Sensitivity

จากข้อมูลที่อ่านได้ สามารถสรุปว่าเป็นรีเลย์ ยี่ห้อ SONGLE โมเดล SRD รองรับการทำงานไฟฟ้ากระแสสลับที่ 250VAC@10A หรือ125VAC@10A รองรับไฟฟ้ากระแสตรงที่ 28VDC@10A ฝั่งขดลวดทำงานด้วยแรงดัน 5 โวลต์โดยตรงจากอาดุยโน่ รับกระแสได้สูงถึง 10 แอมป์ โครงสร้างรีเลย์เป็นแบบซีลด์ หน้าสัมผัสเป็นรูปแบบ 1 from C มีเม็ดแอลอีดีแสดงสถานะการทำงานของรีเลย์ ออกแบบให้ป้องกันวงจรด้านควบคุม ออกจากด้านกำลังโดยการใช้การส่งผ่านด้วยแสง (Optocoupler) ในรีเลย์ทุกตัว ถ้าต้องการให้รีเลย์ทำงานให้ส่งสัญญาณ 0 ไป ถ้าต้องการหยุดการทำงานส่งสัญญาณ 1 ไป แสดงรายละเอียดของรีเลย์ดังนี้

1. การเชื่อมต่อมาตรฐานที่สามารถใช้ควบคุมได้โดยตรงจากไมโครคอนโทรนเลอร์ เช่น Arduino, 8051, AVR, PIC, DSP, ARM, ARM, MSP430 และTTL logic
2. ใช้ไฟฟ้าที่ 5 โวลต์ สามารถรับจากบอร์ดอาดุยโน่ที่มีขา 5 โวลต์ได้
3. ใช้ควบคุมไฟฟ้าแรงดันสูงได้ ที่ไฟฟ้ากระแสตรง (DC) 30 โวลต์ 10 แอมป์ และไฟฟ้ากระแสสลับ (AC) 250 โวลต์ 10 แอมป์
4. เชื่อมต่อด้วยขั้วสกรู ทำให้ติดตั้งได้ง่ายและสะดวก
5. การส่งสัญญาณควบคุมรีเลย์เป็นแบบ Active low

6. วงจรขับรีเลย์เป็นแบบแยกกราวด์ (Opto isolated Relay) ปลอดภัยต่อวงจรไมโครคอนโทรลเลอร์

7. ขนาดรูยึดบอร์ด 3 มิลลิเมตร ขนาด (ยาว x กว้าง x สูง) : 77 x 55 x 20 mm



รูปที่ 2.14 โมดูลรีเลย์แบบ 2 ตัว

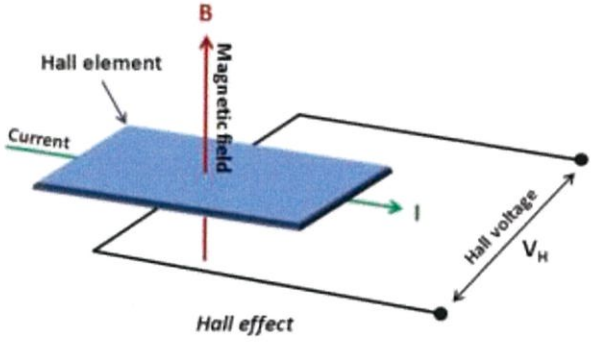
จากรูปที่ 2.14 เป็นการแสดงโมดูลรีเลย์แบบ 2 ตัว และกำหนดตัวเลขเพื่อที่จะอธิบายว่าพินไหน ทำหน้าที่อะไร ดังตารางที่ 2.3

ตารางที่ 2.3 ขาสัญญาณ (Pin Definition) และคำอธิบาย

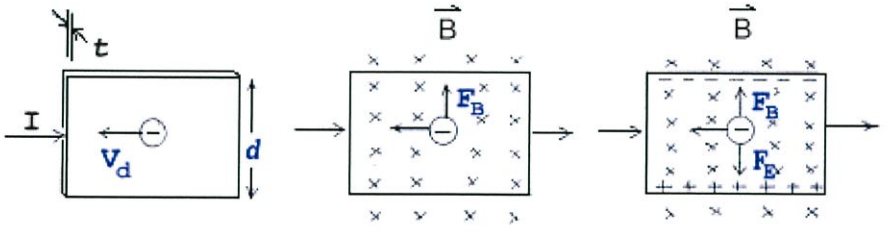
ขาที่	คำอธิบาย
1	ขาสัญญาณอินพุตรีเลย์ 2 (IN2)
2	ขาสัญญาณอินพุตรีเลย์ 1 (IN1)
3	GND
4	+VCC ขาไฟ 5VDC
5	NO2 (Normal Open) ซึ่งหมายถึงหน้าสัมผัสแบบปกติเปิด
6	COM2 (Common) จะตัดหรือต่อวงจร จากขา NC และ NO
7	NC2 (Normal Close) ซึ่งหมายถึงหน้าสัมผัสแบบปกติปิด
8	NO1 (Normal Open)
9	COM1 (Common)
10	NC1 (Normal Close)

2.3 ตัวรับรู้ฮอลล์

ฮอลล์เอฟเฟ็คเซนเซอร์ (Hall Effect Sensor) เป็นตัวแปลงสัญญาณที่แปรผันแรงดันไฟฟ้าเอาท์พุต ที่ตอบสนองกับสนามแม่เหล็ก โดย Hall Effect นั้นมาจากทฤษฎีทางไฟฟ้า ที่ถูกค้นพบโดย Edwin Hall (7 พฤศจิกายน ค.ศ.1885 – 20 พฤศจิกายน ค.ศ.1938) ในปี พ.ศ.2422 ซึ่งสรุปได้ดังนี้ นำแผ่นตัวนำเมื่อมีฟลักซ์แม่เหล็ก (Magnetic Flux) มากระทำในทิศทางที่ตั้งฉากกับแผ่นตัวนำ ทำให้เกิดสนามไฟฟ้าหรือแรงดัน เรียกแรงดันนั้นว่า แรงดันฮอลล์ (Hall Voltage) เกิดขึ้นที่ตัวนำในทิศทางตั้งฉากกับกระแส และฟลักซ์แม่เหล็ก เมื่อจ่ายกระแสคงที่ให้แผ่นตัวนำจะทำให้อิเล็กตรอนเคลื่อนที่จากขั้วลบไปขั้วบวก และจะทำให้ตัวนำเบี่ยงเบนไปด้านบนของตัวนำ เนื่องจากอิเล็กตรอนมีประจุเป็นลบ ทำให้ด้านบนของแผ่นตัวนำมีขั้วไฟฟ้าเป็นลบ ส่วนด้านล่างของแผ่นตัวนำจะมีขั้วตรงข้ามกับด้านบน เมื่อวัดความต่างศักย์ระหว่างด้านบนกับด้านล่างทำให้ได้แรงดันไฟฟ้าออกมาเป็นแรงดันลบ โดยขนาดของแรงดันที่วัดได้จะขึ้นอยู่กับ ความหนาแน่นของฟลักซ์แม่เหล็กที่มากระทำ หากความเข้มสนามแม่เหล็กมากก็จะทำให้เกิดแรงดันมาก ซึ่งจะเรียกปรากฏการณ์นี้ว่าปรากฏการณ์ฮอลล์ แสดงได้ดังรูปที่ 2.15 และรูปที่ 2.16



รูปที่ 2.15 การเกิดแรงดันฮอลล์ในปรากฏการณ์ฮอลล์ [3]



รูปที่ 2.16 การเปลี่ยนแปลงขั้วไฟฟ้าในปรากฏการณ์ฮอลล์ [4]

สำหรับฮอลล์เอฟเฟ็คเซนเซอร์ มีสมบัติดังนี้ Input Voltage 4.5 – 6 Volt, Offset Voltage 2.5 Volt เมื่อต่อแหล่งจ่ายไฟเข้าที่ขา 1 และ2 จะสามารถวัดค่าด้วยโวลต์มิเตอร์ที่ขา 2 และ 3 ได้ประมาณ 2.5 Volt แสดงได้ดังรูปที่ 2.17 ซึ่งค่าที่ได้นี้เป็นค่าความต่างศักย์ขณะที่ไม่มีสนามแม่เหล็ก เรียกค่านี้ว่า Offset Voltage จากนั้นเมื่อนำแม่เหล็กเข้าใกล้พื้นที่ที่ทำงานของฮอลล์เอฟเฟ็คเซนเซอร์ (Active Area) เพื่อสร้างสนามแม่เหล็ก จะทำให้ความต่างศักย์เกิดการเปลี่ยนแปลง โดยจะเพิ่มขึ้นเมื่อนำขั้วใต้เข้าใกล้ หรือลดลงเมื่อนำขั้วเหนือเข้าใกล้ ความต่างศักย์ที่เปลี่ยนไป มีความสัมพันธ์กับความเข้มของสนามแม่เหล็ก หรือความหนาแน่นฟลักซ์แม่เหล็ก (Magnetic Flux Density) ดังสมการที่ 2.1

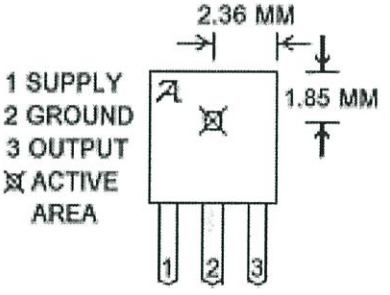
$$B = (V_{out\,(B)} - V_{out\,(O)})\,S^{-1} \qquad (2.1)$$

เมื่อ $V_{out\,(O)}$ เป็นความต่างศักย์ขณะไม่มีสนามแม่เหล็ก

$V_{out\,(B)}$ เป็นความต่างศักย์ขณะมีสนามแม่เหล็ก

S เป็นสัมประสิทธิ์ความไว มีหน่วยเป็นโวลต์ต่อเทสลา (V/T)

B เป็นความเข้มของสนามแม่เหล็ก หรือความหนาแน่นฟลักซ์แม่เหล็ก มีหน่วยเป็นเทสลา (T)

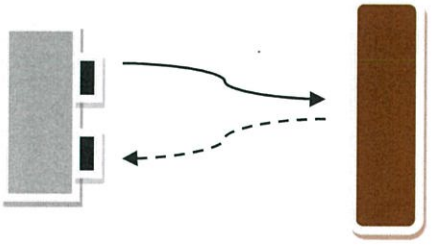


รูปที่ 2.17 Hall Effect Sensor [4]

ตัวรับรู้ฮอลล์สามารถวัดสนามแม่เหล็กในบริเวณใกล้แม่เหล็กถาวร สนามแม่เหล็กที่เกิดขึ้นบริเวณปลายโซเลนอยด์ และสนามแม่เหล็กใกล้เส้นลวดตัวนำที่มีกระแสไฟฟ้าผ่านได้

2.4 อัลตราโซนิคเซนเซอร์

เซนเซอร์ชนิดอัลตราโซนิค (Ultrasonic Sensor Module) หรือเซนเซอร์ชนิดใช้เสียงเป็นอุปกรณ์ที่สามารถวัดระยะทางไปยังวัตถุ โดยอาศัยคลื่นเสียงที่มีความถี่สูงกว่า 20 กิโลเฮิร์ต (kHz) ซึ่งเป็นย่านความถี่ที่มนุษย์ไม่สามารถได้ยินได้ เซนเซอร์จะวัดระยะทางได้โดยการส่งคลื่นเสียงออก ไป และฟังคลื่นเสียงที่เด้งกลับมา อาศัยการเคลื่อนที่และการกระจายของคลื่นไปกระทบกับตัวกลาง อาจเป็นได้ทั้งของแข็ง และของเหลว ส่วนที่สะท้อนกลับ เรียกว่า "Echo" ระยะเวลาระหว่างคลื่นที่ส่งออกไปและคลื่นเสียงที่สะท้อนกลับ แสดงดังรูปที่ 2.18 สามารถนำมาคำนวณหาระยะห่างระหว่างเซนเซอร์ และวัตถุได้

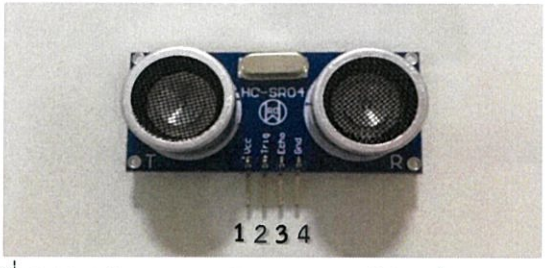


รูปที่ 2.18 การสะท้อนของคลื่นจาก Ultrasonic Sensor Module

เนื่องจากทราบเราทราบว่าความเร็วเสียงผ่านตัวกลางที่เป็นอากาศเท่ากับ 344 m/s สามารถคำนวณหาระยะทางโดยใช้เวลาที่คลื่นเสียงเดินทางกลับมา และคูณด้วย 344 เมตร จากนั้นหาร 2 เพราะ เวลาที่วัดได้จากเซนเซอร์ คือเวลาจากการเดินทางไปและกลับ ซึ่งสูตรของการคำนวณหาระยะทางเป็นดังนี้

$$\text{distance} = \frac{\text{speed of sound} \times \text{time taken}}{2} \qquad (2.2)$$

สิ่งสำคัญคือต้องเข้าใจว่าวัตถุบางอย่างอาจไม่สามารถตรวจวัดได้โดยเซนเซอร์ชนิดอัลตราโซนิก เนื่องจากวัตถุบางอย่างมีรูปร่าง หรือวางตำแหน่งในลักษณะที่คลื่นเสียงเบี่ยงเบนออกจากวัตถุ นอกจากนี้ยังมีความเป็นไปได้ที่วัตถุจะมีขนาดเล็กเกินไปที่จะสะท้อนให้เห็นถึงคลื่นเสียงที่ส่งกลับไปยังเซนเซอร์เพื่อตรวจจับได้ และวัตถุบางอย่างสามารถดูดซับคลื่นเสียงได้ทั้งหมด เช่น ผ้าปูพรม ซึ่งหมายความว่าเซนเซอร์อาจจะตรวจจับได้ไม่ถูกต้องถึง 100 เปอร์เซนต์



รูปที่ 2.19 Ultrasonic Sensor Module รุ่น HC-SR04

จากรูปที่ 2.19 แสดงตัวอย่างอัลตราโซนิคเซนเซอร์ และกำหนดตัวเลขเพื่อที่จะอธิบายการต่อเพื่อใช้งานทำได้โดยต่อขาพิน ว่าพินไหนทำหน้าที่อะไร ดังตารางที่ 2.4 และมีคุณสมบัติดังตารางที่ 2.5

ตารางที่ 2.4 ขาสัญญาณ (Pin Definition) และคำอธิบายของอัลตราโซนิคเซนเซอร์

ขาที่	คำอธิบาย
1	5V Supply
2	Trigger Pulse Input
3	Echo Pulse Output
4	GND

ตารางที่ 2.5 คุณสมบัติการทำงานของอัลตราโซนิคเซนเซอร์

การทำงาน	คำอธิบาย
แรงดัน	DC 5 V
กระแสไฟฟ้า	15 mA
ความถี่	40 Hz
ระยะสูงสุดที่อ่านได้	4 m
ระยะต่ำสุดที่อ่านได้	2 cm
Induction Angle	Less than 15
Trigger Input	10uS TTL Level signal and the range in propotion
ขนาด	45*20*15 mm
Output way	GPIO
ทำงานที่อุณหภูมิ	0 ~ + 70
ความแม่นยำในการตรวจจับ	0.3 cm + 1%

2.5 NodeMCU

NodeMCU/ESP8266 (ซึ่งในบางครั้ง อาจเรียกบอร์ดลักษณะนี้ว่า WiFi controller) เป็นโปรแกรมที่ใช้งานง่าย เขียนด้วยภาษา C และความที่เป็น Open Source ทำให้ใช้งานได้โดยไม่มีค่าใช้จ่าย ได้รับความนิยมสูง จึงทำให้มีแหล่งข้อมูลให้ศึกษาค้นคว้าเพิ่มเติมในเว็บบอร์ดหรือเว็บไซต์ในอินเทอร์เน็ตอีกมากมาย และในส่วนของบอร์ด NodeMCU/ESP8266 เองนั้น เป็นบอร์ดไมโครคอนโทลเลอร์ที่มี จำนวนขาพอร์ตอินพุตและเอาต์พุตมากพอสำหรับการนำไปใช้งานจริง สามารถต่อกับเซ็นเซอร์ได้ทั้งแบบดิจิตอลและแอนะล็อก และยังต่อเพื่อขับอุปกรณ์เอาต์พุตให้ทำงานโดยที่เราจะต้องเขียนโปรแกรมเพื่อสั่งงานให้บอร์ด NodeMCU/ESP8266 สามารถควบคุมอุปกรณ์ต่างๆ เช่น ควบคุมการเปิดปิดหลอดไฟ, ปลั๊กไฟฟ้า หรือเครื่องรดน้ำต้นไม้ เป็นต้น และเนื่องจากมีโมดูล WiFi ในตัว จึงสามารถเชื่อมต่อเพื่อส่งข้อมูลหรือสั่งงานผ่านทางอินเทอร์เน็ตได้โดยไม่ต้องหาซื้ออุปกรณ์มาต่อเพิ่ม นอกจากนั้นยังมีราคาถูก ซึ่งจะช่วยลดต้นทุนลงเป็นอย่างมากหากต้องการนำบอร์ดไปใช้ในการพัฒนาอุปกรณ์ Internet of Things (IoT) ไม่ว่าจะเป็นการทำ Web Server ขนาดเล็ก การควบคุมการเปิดปิดไฟผ่านWiFi และ อื่นๆอีกมากมาย

2.5.1 แพลตฟอร์มของ NodeMCU

ทางผู้พัฒนาตั้งใจจะออก NodeMCU ให้เป็นแพลตฟอร์มที่ออกแบบทุกอย่างเป็น Node การทำงานย่อยๆ และ ใช้ภาษา Lua ในการเขียนโปรแกรม แต่ด้วย platfrom ที่สะดวกในการใช้งาน ทางกลุ่มนักพัฒนาของ ESP8266 จึงนำ NodeMCU (ESP8266) มาบรรจุในเป็นบอร์ดหนึ่งของ ARDUINO IDE (ESP8266) ด้วย จึงได้มีการพัฒนาต่อให้สามารถเขียนในภาษา C++ ซึ่งหลังจากที่บอร์ด NodeMCU (ESP8266) นี้มีการพัฒนาบน ARDUINO IDE เรียบร้อยแล้ว หากเป็นผู้ที่นิยมเล่นไมโครคอนโทรลเลอร์อยู่ก่อนจะนิยมเล่นเป็นภาษา C/C++ ซึ่งภาษานี้สามารถไปได้กว้างเล่นได้หลายอย่างกว่า Lua ภาษาอาดุยโน่ จึงเป็นภาษาที่ใช้ในการเขียน เพื่อควบคุมไมโครคอนโทรลเลอร์ (MCU) โดยจะมีหลักการเขียนเหมือนกับภาษา C/C++ และโปรแกรม Arduino IDE เป็นโปรแกรมสำหรับใช้ในการเขียนภาษาอาดุยโน่, คอมไพล์โปรแกรม และอัพโหลดโปรแกรมลงบอร์ด

ความแตกต่างของลักษณะภาษาที่เขียนระหว่างภาษา Lua ใน Lua IDE และภาษา C/C++ บน Arduino IDE ในทางด้านของภาษา Lua จะออกแบบให้เป็น interpreter มีคำสั่งการใช้งาน พอเพียงสำหรับใช้งานไมโครคอนโทรลเลอร์ แต่ด้วยความที่เป็น Interpreter เป็นสิ่งที่ต้องใช้พื้นที่จำนวนมากใน flash rom ที่จะเอา firmware ตัวนี้ลงไปก่อน และ ภาษา Lua ในบางเวลาจะทำงานช้ากว่า C/C++ แต่ข้อดีของ Lua คือความเป็น Interpreter นี้ จึงสามารถเขียนโปรแกรม จากเครื่องใดก็ได้ ไม่ต้องพึ่ง compiler ต่อเข้ากับคอมได้ ก็จะสามารถเขียนโปรแกรมได้เลย ในภาษา Lua จะเขียนเป็นแบบ Event – Drive และ มองทุกอย่างเป็น Object ส่วนใน C/C++ มีกลุ่มพัฒนา นำ

SDK ของ ESP8266 มาพัฒนาต่อยอด ให้เข้ากับแพลตฟอร์ม ของ Arduino IDE จึงทำให้ ESP8266 ใช้ภาษา C/C++ ได้นี้เอง

2.5.2 ข้อดีในการเลือกใช้ NodeMCU/ESP8266

สำหรับเหตุผลในการเลือกใช้สามารถแจกแจงได้ตามนี้ คือ

1. ราคา : NodeMCU/ESP8266 มีราคาไม่แพงเมื่อเทียบกับแพลตฟอร์มไมโครคอนโทรลเลอร์อื่นๆ

2. แพลตฟอร์ม : สามารถทำงานข้ามแพลตฟอร์มได้ NodeMCU/ESP8266 สามารถทำงานบนระบบปฏิบัติการได้หลากหลาย Windows, Macintosh OSX และ Linux อีกทั้งยังเขียนได้หลายภาษา ตามความสะดวกของผู้ใช้

3. การเขียนโปรแกรม : การใช้งานมีความเรียบง่าย และชัดเจน สามารถใช้งานง่ายสำหรับผู้เริ่มต้น แต่ก็มีความยืดหยุ่นสำหรับผู้ใช้งานชั้นสูงที่สามารถนำไปใช้ประโยชน์ได้เช่นกัน

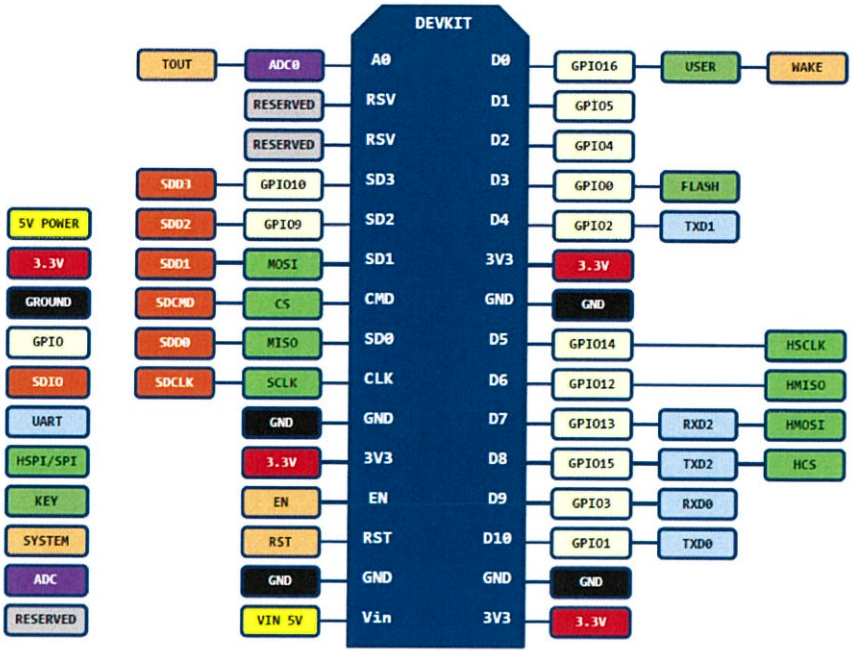
4. ซอฟต์แวร์ Open Source และส่วนขยาย : ซอฟต์แวร์ถูกเผยแพร่เป็นเครื่องมือ Open Source ซึ่งสามารถใช้ได้กับโปรแกรมเมอร์ที่มีประสบการณ์ ภาษาสามารถขยายได้ผ่าน C Library เป็นแบบ Open Source Project มี Source code ให้ได้เรียนรู้อยู่บน Github

5. ฮาร์ดแวร์ Open Source และส่วนขยาย : ชิบภายใน ESP 8266 มี CPU ขนาด 32 bit แตกต่างจากอาดุยโน่ ที่เป็น CPU 8 bit ถึงแม้ขา I/O จะไม่มากเท่าของอาดุยโน่ แต่เราสามารถเขียนโปรแกรมลงบนขา GPIO ได้ทุกขาพอๆกัน เป็นข้อดีที่เพิ่มมาจากความต้องการใช้ WIFI เชื่อมต่อเมื่อต้องการเล่นอาดุยโน่ ทำให้ต้องซื้อ Module wifi เพิ่ม นั่นคือ NodeMCU/ESP8266 มีต้นทุนต่ำกว่ามาก

6. มีอุปกรณ์หลายอย่างที่ใช้งานที่แรงดัน +3.3 V เป็นส่วนใหญ่ ดังนั้นเราสามารถนำ NodeMCU (ESP8266) มาใช้เชื่อมต่อได้โดยตรง

2.5.3 NodeMCU Devkit V1.0

NodeMCU Devkit V1.0 เป็นตัวที่พัฒนาจาก NodeMCU Version 0.9 เดิม โดยเป็นโมดูลที่ประกอบด้วย ESP8266-12E มีเสาอากาศแบบ PCB Antenna เชื่อมต่อเฮดเดอร์สำหรับขาสัญญาณต่างๆ ได้แก่ GPIO PWM I2C 1-wire ADC และ มี SPI เพิ่มขึ้นมาจาก Version เดิม มีส่วนของ USB-to-TTL และพอร์ต micro USB ใช้เชื่อมต่อเข้ากับเครื่องคอมพิวเตอร์เพื่อพัฒนาโปรแกรม สามารถติดตั้งเฟิร์มแวร์ NodeMCU ได้เป็นหนึ่งในชนิดที่ได้รับความนิยมมากที่สุด แม้ว่าจะไม่ได้เป็นบอร์ดรุ่นแรกที่ถูกผลิตออกมา แต่ก็ถูกใช้อย่างแพร่หลาย และยอมรับในวงกว้าง เนื่องจากถูกพัฒนาให้ใช้ง่ายขึ้นกว่ารุ่นก่อนๆ และเชื่อมต่อกับคอมพิวเตอร์ได้โดยตรงผ่ายสาย USB โดยรูปที่ 2.20 แสดงบอร์ดพินใน NodeMCU Devkit V1.0



รูปที่ 2.20 บอร์ดพินใน NodeMCU Devkit V1.0 [9]

2.5.4 วิธีเชื่อมต่อ NodeMCU เข้ากับระบบ Network

ในการใช้งานเครือข่ายด้วยโปรโตคอล TCP นั้นสามารถใช้งานได้ 2 โหมด ได้แก่ TCP Server และ TCP Client

1. TCP Server เป็นการกำหนดให้ ESP8266 ทำหน้าที่เป็นเครื่องแม่ข่าย ให้บริการอย่างใดอย่างหนึ่ง โดยการเปิดพอร์ต (Listening) เพื่อรอรับการเชื่อมต่อจากเครื่องลูกข่าย

2. TCP Client เป็นการกำหนดให้ ESP8266 ทำหน้าที่เป็นเครื่องลูกข่าย โดยเชื่อมต่อไปยังเครื่องแม่ข่ายด้วยไอพีแอดเดรส และพอร์ตที่กำหนดไว้เพื่อขอใช้บริการนั้นๆ

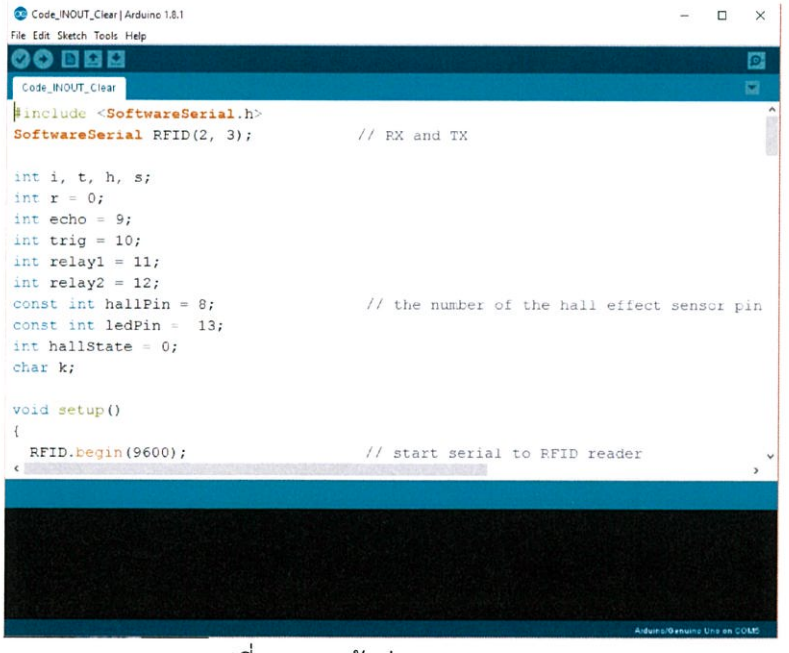
2.5.5 คุณลักษณะของ ESP8266

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity

- WiFi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IR Remote Control, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4s guard interval
- Deep sleep power <10uA, Power down leakage current < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20 dBm output power in 802.11b mode
- Operating temperature range -40C ~ 125C
- FCC, CE, TELEC, WiFi Alliance, and SRRC certified

2.6 Arduino IDE

Arduino IDE คำว่า IDE ย่อมากจาก Integrated Development Environment คือ เครื่องมือสำหรับการเขียนโปรแกรมภาษาอาดุยโน่ ที่ใช้งานได้กับบอร์ดอาดุยโน่ทุกรุ่น ประกอบ ด้วย โปรแกรมแก้ไขข้อความสำหรับเขียนโค้ดคำสั่ง, พื้นที่ข้อความ, คอนโซลข้อความ, แถบของเครื่องมือที่ มีปุ่มสำหนับฟังก์ชั่นทั่วไป และชุดของเมนู เชื่อมต่อกับฮาร์ดแวร์อาดุยโน่ เพื่ออัพโหลดโปรแกรม และ สื่อสารกับบอร์ด ตัวอย่างหน้าตาของโปรแกรม Arduino IDE แสดงดังรูปที่ 2.21



รูปที่ 2.21 หน้าต่าง Arduino IDE

2.7 RFID Reader Module

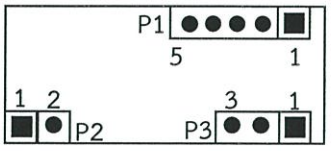
การอ่านป้ายแท็กอาร์เอฟไอดีนั้น จำเป็นต้องมีเครื่องอ่านผ่านสายอากาศดพื่อประมวลผลและส่งค่าข้อมูล ในปริญญานิพนธ์นี้จะใช้โมดูลการ์ด RDM6300 125kHz ซึ่งถูกออกแบบมาสำหรับการอ่านโค้ดจากแท็กแบบอ่านอย่างเดียวที่รองรับบัตร 125kHz และการ์ดอ่าน/เขียน สามารถใช้ในสำนักงานและบ้าน เพื่อการรักษาความปลอดภัยข้อมูลส่วนบุคคล ควบคุมการเข้าถึง การป้องกันการปลอมแปลง แบบโต้ตอบและระบบการควบคุมการผลิต เป็นต้น มีคุณสมบัติดังตารางที่ 2.6

ตารางที่ 2.6 คุณสมบัติของโมดูลการ์ด RDM6300 125kHz

คุณสมบัติ	คำอธิบาย
ความถี่	125kHz
Baud Rate	9600 (TTL Electricity Level RS232 format)
Interface	Weigang 26 Or TTL Electricity Level RS232 format
Power supply	DC 5V (±5%)
Current	<50 mA
Operating range	>50mm (Depend on Card/Tag shape, manufacturer)
Expand I/O port	N/A
Indication light	N/A
ทำงานที่อุณหภูมิ	-10℃~ +70℃
เก็บที่อุณหภูมิ	-20℃~ +80℃
ความชื้นสัมพัทธ์	0 ~ 95%
ขนาด	38.5mm ×19mm×9mm

2.7.1 รายละเอียด RFID Reader Module RDM6300 125kHz

รายละเอียดการทำงานของ RFID Reader Module แสดงดังรูปที่ 2.22



รูปที่ 2.22 RFID Reader Module

การทำงานของขาพินที่จำเป็นต้องใช้งานมี 3 ส่วน ได้แก่

ส่วน P1 : PIN1-TX, PIN2- RX, PIN3 PIN4-GND และPIN5 ทำหน้าที่เป็น +5V(DC)

ส่วน P2 : PIN1-ANT1 และ PIN2-ANT2

ส่วน P3 : PIN1-LED, PIN2 ทำหน้าที่เป็น +5V(DC) และPIN3-GND

2.8 ความรู้ทั่วไปเกี่ยวกับสายอากาศ

สายอากาศ (Antenna) เป็นอุปกรณ์ไฟฟ้าชนิดหนึ่งที่เปลี่ยนพลังงานไฟฟ้าให้เป็นคลื่นแม่เหล็กไฟฟ้าหรือในทางตรงกันข้าม โดยปกติแล้วสายอากาศจะถูกใช้กับเครื่องส่งและเครื่องรับวิทยุ ในด้านการส่ง เครื่องส่งวิทยุจะป้อนคลื่นกระแสไฟฟ้า ที่ความถี่วิทยุไปยังขั้วไฟฟ้าทั้งสองของสายอากาศ จากนั้นสายอากาศจะแผ่รังสีพลังงาน จากกระแสในรูปของคลื่นแม่เหล็กไฟฟ้าในการรับสายอากาศจะดักจับพลังงานของคลื่นแม่เหล็กไฟฟ้า เพื่อที่จะสร้างแรงดันไฟฟ้าขนาดเล็กที่ขั้วไฟฟ้าของมัน แรงดันไฟฟ้านี้จะถูกส่งต่อไปให้เครื่องรับเพื่อทำการขยายสัญญาณต่อไป

สายอากาศเป็นชิ้นส่วนที่สำคัญของอุปกรณ์ทุกชนิดที่ใช้คลื่นวิทยุในการสื่อสาร เช่น สถานีวิทยุกระจายเสียง, สถานีโทรทัศน์, วิทยุสองทาง, เครื่องรับสื่อสาร, เรดาร์, โทรศัพท์เคลื่อนที่, และการสื่อสารดาวเทียม นอกจากนี้ ยังใช้กับอุปกรณ์อื่นๆที่ต้องการใช้การสื่อสารไร้สาย เช่น ประตูโรงรถอัตโนมัติ, ไมโครโฟนไร้สาย, บลูทูธ, แลนไร้สาย, ฉลากRFID และ ของเล่นวิทยุบังคับต่างๆ

โดยทั่วไปสายอากาศจะประกอบด้วยโครงสร้างของตัวนำโลหะ ที่เรียกว่าอุปกรณ์ขับ (driven element) ที่ต่อทางไฟฟ้าเข้ากับเครื่องส่งหรือเครื่องรับ เครื่องส่งจะบังคับให้กระแสไฟฟ้าที่เป็นคลื่นของอิเล็กตรอนไหลผ่านสายอากาศ กระแสไฟฟ้าดังกล่าวจะสร้างสนามไฟฟ้าที่เป็นคลื่นไปตามอีลิเมนท์นั้น สนามพลังงานไฟฟ้าที่เปลี่ยนแปลงไปตามเวลาเหล่านี้จะถูกแผ่กระจายออกไปจากสายอากาศเข้าสู่อากาศในรูปของคลื่นสนามแม่เหล็กไฟฟ้าเคลื่อนที่ตามขวาง ทางด้านรับคลื่นเหล่านี้เข้ามารวมกันที่สายอากาศ สนามแม่เหล็กไฟฟ้าที่เป็นคลื่นจะสร้างแรงขึ้นบนอิเล็กตรอนในอุปกรณ์ขับของสายอากาศ ทำให้อิเล็กตรอนต้องเคลื่อนที่กลับไปกลับมา เป็นการสร้างกระแสที่เป็นคลื่นในสายอากาศ

2.8.1 ชนิดของสายอากาศ

เราสามารถแบ่งชนิดของสายอากาศในแบบต่างๆได้ด้วยกันหลายปัจจัย

2.8.1.1 สายอากาศที่แบ่งตามประเภท ได้แก่

1) สายอากาศแบบเส้นลวด รูปแบบจะมีหลายแบบแต่ส่วนใหญ่จะเป็น แบบเส้นตรง แบบวงกลม หรือแบบสปริง เป็นส่วนใหญ่มักพบเจอได้มาก กับอุปกรณ์เคลื่อนที่ เช่น เสาอากาศวิทยุ

2) สายอากาศแบบโครงโลหะ ถูกใช้งานในวงแคบเฉพาะด้านเท่านั้น ลักษณะเด่นคือสายอากาศประเภทนี้ บังคับทิศทางได้แน่นอน เพราะโครงโลหะจะหันปลายไปทางทิศทศต้องการ โดยทั่วไปแล้วมักพบอยู่สามแบบคือแบบพีระมิด แบบทรงกรวย และแบบสี่เหลี่ยม ขนาดของสายอากาศชนิดนี้ ยาวประมาณ 1-2 ฟุต

3) สายอากาศแบบระนาบ หรือเรียกอีกอย่างว่าสายอากาศแบบไมโครสตริป เป็นสายอากาศบนแผ่นคล้ายแผ่นวงจรไฟฟ้าทั่วไป

4) สายอากาศแบบอาเรย์ เป็นการนำสายอากาศแบบเส้นลวดมาเพิ่มประสิทธิภาพขึ้น ทำให้รับสัญญาณได้ดีขึ้น แต่ข้อเสียคือทิศทางต้องตรงกันพอดี จึงถูกใช้งานกับอุปกรณ์ที่ไม่เคลื่อนที่ เช่น เสาอากาศโทรทัศน์ ซึ่งยาวประมาณ 1-2เมตร

5) แบบจานสายอากาศ ใช้งานกับดาวเทียมหรือการสื่อสารที่มีระยะทางไกลมาก เนื่องจากมีประสิทธิภาพสูงที่สุด แต่ข้อเสียคือ ทิศทางแคบมากที่สุดเช่นกัน จึงไม่ควรคลาดเคลื่อน และ สายอากาศชนิดนี้ยังมีขนาดใหญ่ที่สุดอีกด้วย

2.8.1.2 สิ่งที่มีความสำคัญมากในเรื่องของการพิจารณาเลือกใช้สายอากาศคือ การโพลาไรซ์ของคลื่น (Wave Polarization) ในทางธรรมชาติของคลื่นแม่เหล็กไฟฟ้าหรือคลื่นวิทยุที่ถูกส่งออกไปจากสายอากาศของสถานีวิทยุกระจายเสียง ไม่ว่าจะเป็นระบบอะไรก็ตาม หากต้องการให้สายอากาศของเครื่องรับวิทยุสามารถรับสัญญาณที่มีประสิทธิภาพสูงสุดจะต้องให้การโพลาไรซ์ของสายอากาศของเครื่องรับนี้ อยู่ในทิศทางเดียวกับโพลาไรซ์ของคลื่น ที่ออกมาจากสายอากาศของสถานีส่ง ชนิดของสายอากาศที่แบ่งตามการโพลาไรซ์หรือการจัดขั้วคลื่น ได้แก่

1) สายอากาศที่มีการโพลาไรซ์ในแนวตั้ง (Vertical Polarization)
2) สายอากาศที่มีการโพลาไรซ์ในแนวนอน (Horizontal Polarization)
3) สายอากาศที่มีการโพลาไรซ์แบบวงกลม (Circular Polarization)

2.8.1.3 แบ่งตามแบบรูปการแพร่กระจายคลื่น (Radiation Pattern)

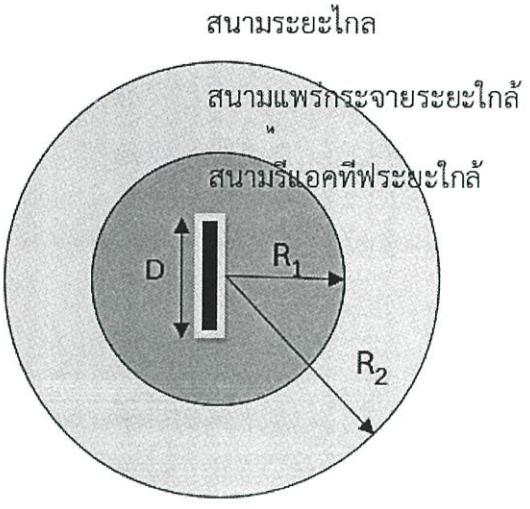
สายอากาศสามารถออกแบบให้ส่งหรือรับคลื่นวิทยุ ได้หลายแบบด้วยกัน ได้แก่ สายอากาศที่ส่งหรือรับคลื่นได้ทุกทิศทาง แนวราบเท่าๆกัน ที่เรียกว่า สายอากาศทุกทิศทาง (Omnidirectional antenna) สายอากาศที่ส่งหรือรับคลื่นวิทยุได้ในทิศทางเฉพาะ ที่เรียกว่า สายอากาศเฉพาะทิศทาง (Directional antenna) หรือสายอากาศเกนสูง (อังกฤษ: High gain antenna) ซึ่งสายอากาศลักษณะนี้อาจต้องมีอุปกรณ์หรือตัวประกอบอื่นๆเพิ่มเติม ที่ไม่มีการต่อถึงกันทางไฟฟ้าเข้ากับเครื่องส่งหรือเครื่องรับแต่อย่างใด อุปกรณ์ดังกล่าว ได่แก่ อุปกรณ์กาฝาก (Parasitic elements), ตัวสะท้อนแบบโค้ง (Parabolic reflectors) หรือสายอากาศปากแตร (Horn antenna) ซึ่งมีหน้าที่นำทางคลื่นวิทยุให้อยู่ในรูปลำแสงหรือรูปแบบการแผ่กระจายคลื่นที่ต้องการ

2.8.1.4 ตามการแบ่งขอบเขตสนามรอบสายอากาศ

1) บริเวณสนามรีแอคทีฟระยะใกล้ (Reactive Near-Field Region) หมายถึง ที่บริเวณสนามรีแอคทีฟระยะใกล้ เป็นสนามที่ล้อมรอบใกล้ชิดสายอากาศมากที่สุด ซึ่งจะมีการแผ่คลื่นพลังงานมากท่สุดเช่นกัน หรือจุดที่กระจายสัญญาณได้รอบทิศทาง

2) บริเวณสนามแพร่กระจายระยะใกล้ (Radiating Near-Field Region) ในสนามบริเวณนี้จะมีสนามที่แผ่กระจายอยู่เป็นส่วนใหญ่ และ การกระจายของสนามตามมุมต่างๆนั้น แปรผันตามระยะทางจากสายอากาศ เมื่อสายอากาศมีขนาดเล็กมากๆเมื่อเทียบกับความยาวคลื่น สนามในบริเวณนี้อาจไม่เกิดขึ้น เนื่องจากไม่มีลักษณะความเป็นคลื่นระนาบอย่างเพียงพอ และมีโครงสร้างซับซ้อน

3) บริเวณสนามแพร่กระจายระยะไกล (Radiating Far-Field Region) เป็นบริเวณที่ใช้จัดวางสายอากาศเพื่อทำการวัดแบบรูปการแผ่กระจายกำลังงานหรือทำการวัดคุณลักษณะต่างๆ ของสายอากาศ ในส่วนของการกระจายของสนามจะเป็นอิสระจากระยะห่างสายอากาศ และมีลักษณะความเป็นคลื่นระนาบอย่างเด่นชัด คือรูปแบบการกระจายแบบปกติของสนามแม่เหล็กไฟฟ้าในระนาบอยู่ในแนวขวางกับทิศทางการแผ่กระจาย



รูปที่ 2.23 การแบ่งขอบเขตสนาม

จากรูปที่ 2.23 การแบ่งขอบเขตของสนามแม่เหล็กไฟฟ้ารอบสายอากาศ สามารถแบ่งได้เป็น 3 บริเวณ โดยที่ระยะ R_1 และ R_2 เป็นดังสมการที่ 2.3 และ 2.4

$$R_1=0.62\sqrt{\frac{D^3}{\lambda}} \tag{2.3}$$

$$R_2=\frac{2D^2}{\lambda} \tag{2.4}$$

เมื่อ R_1 คือ ระยะจากสายอากาศถึงบริเวณสนามระยะใกล้

R_2 คือ ระยะจากสายอากาศถึงบริเวณสนามระยะไกล

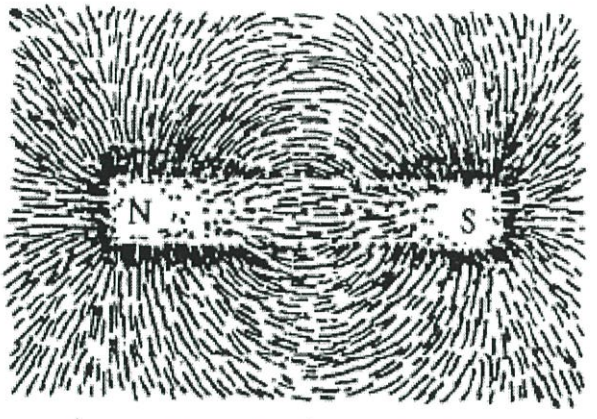
D คือ ขนาดของสายอากาศ

λ คือ ความยาวคลื่น

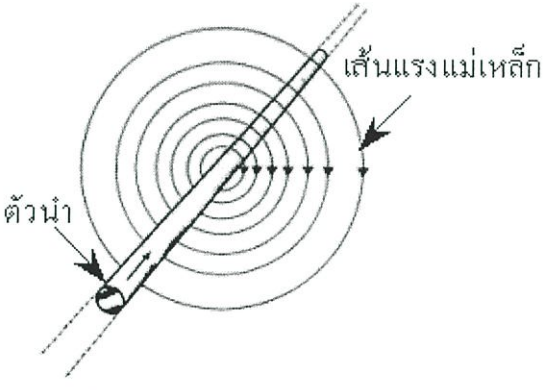
2.9 สนามแม่เหล็กไฟฟ้า

เส้นสมมุติที่เขียนขึ้นเพื่อแสดงอาณาเขตและความเข้มของเส้นแรงที่เกิดขึ้น ระหว่างวัตถุที่มีความแตกต่างของศักย์ไฟฟ้าหรือแรงดันไฟฟ้า เรียกว่า สนามไฟฟ้า ส่วนสนามที่เกิดขึ้นโดยรอบวัตถุที่มีกระแสไฟฟ้าไหล เรียกว่า สนามแม่เหล็ก ในกรณีที่กล่าวถึงทั้งสนามไฟฟ้า และสนามแม่เหล็กพร้อมกันมักจะเรียกรวมว่า สนามแม่เหล็กไฟฟ้า (Electromagnetic Field: EMF) หรือ คลื่นแม่เหล็กไฟฟ้า

สนามแม่เหล็กไฟฟ้ายังสามารถแบ่งออกเป็น สนามแม่เหล็กไฟฟ้าสถิตที่ไม่มีการเปลี่ยนตามเวลา (Static Field หรือ DC Field) เช่น สนามไฟฟ้าระหว่างก้อนเมฆกับพื้นโลก สนามแม่เหล็กจากแม่เหล็กถาวร และสนามแม่เหล็กโลก ส่วนอีกประเภทคือสนามแม่เหล็กไฟฟ้าที่มีการเปลี่ยนแปลงตามเวลา (Dynamic Field หรือAC Field) สนามแม่เหล็กเป็นปริมาณเวกเตอร์ ซึ่ง ณ ตำแหน่งใดๆ มีทิศตามทิศขั้วเหนือของเข็มทิศ ทำให้เกิด เส้นแรงแม่เหล็ก ดังรูปที่ 2.24



รูปที่ 2.24 เส้นแรงแม่เหล็กของแท่งแม่เหล็ก [6]



รูปที่ 2.25 เส้นแรงแม่เหล็กรอบตัวนำ [6]

จากรูปที่ 2.25 แสดงเส้นแรงแม่เหล็กที่เกิดขึ้นรอบตัวนำที่มีกระแสไฟฟ้าไหลผ่าน ซึ่งมีลักษณะเป็นรูปวงกลม โดยเส้นแรงแม่เหล็กมีทิศทางไปในทิศของการขันสกรูเกลียวขวาเมื่อกระแสมีทิศทางพุ่งเข้าและจะไปในทิศการขันสกรูเกลียวซ้ายเมื่อกระแสพุ่งออก

ของสายอากาศ ทำให้อิเล็กตรอนต้องเคลื่อนที่กลับไปกลับมา เป็นการสร้างกระแสที่เป็นคลื่นในสายอากาศ

2.9.1 ความเข้มของสนามแม่เหล็ก

ความเข้มความเข้มของสนามแม่เหล็ก (B): จำนวนเส้นแรงแม่เหล็กที่พุ่งตั้ง ฉากกับพื้นที่หนึ่งตารางหน่วย ดังสมการ 2.5

$$B = \frac{\Phi}{A} \tag{2.5}$$

Φ คือ จำนวนเส้นแรงแม่เหล็ก หรือเรียกว่า ฟลักซ์แม่เหล็ก หน่วย เวเบอร์ (W)

A คือ พื้นที่ หน่วย ตร.ม. (m^2)

B คือ ความเข้มของสนามแม่เหล็ก หน่วย เวเบอร์ต่อตร.ม. (Wb/m^2)

กระแสไฟฟ้าเป็นพารามิเตอร์นึง ที่ทำให้เกิดสนามแม่เหล็ก โดย Hans Christian Oersted พบว่ากระแสไฟฟ้าผ่านขดลวดตัวนำแบบต่างๆ สามารถทำให้เกิดสนามแม่เหล็กรูปร่างต่างๆกันด้วย ได้แก่

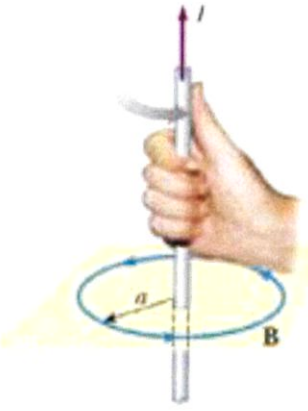
2.9.1.1 สนามแม่เหล็กจากกระแสไฟฟ้าไหลในเส้นลวดตรงยาว

พบว่าสนามแม่เหล็กจากกระแสไฟฟ้าไหลในเส้นลวดตรงยาว จะมีความเข้มของสนามแม่เหล็กแปรผันตรง กับปริมาณกระแส โดยทิศทางของสนามแม่เหล็กเป็นไปตามกฎมือขวา ดังสมการ 2.6 และดังรูปที่ 2.26

$$B = \frac{\mu_0 I}{2\pi a} \tag{2.6}$$

$\mu_0 = 4\pi \times 10^{-7}$ H/m คือสภาพซึมซาบได้ของสนามแม่เหล็กในสญญากาศ

I = กระแสไฟฟ้าที่ไหลผ่าน



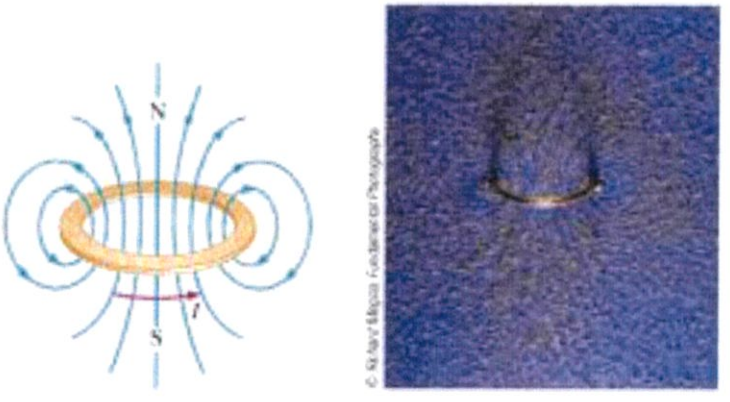
รูปที่ 2.26 สนามแม่เหล็กจากกระแสไฟฟ้าไหลในเส้นลวดตรงยาว [7]

2.9.1.2 สนามแม่เหล็กจากกระแสไฟฟ้าไหลในขดลวด

สนามแม่เหล็กจะเกิดรอบๆขดลวด โดยขนาดของสนามที่จุดศูนย์กลางของขดลวดแปรผกผันกับรัศมีของขดลวด แต่จะแปรผันตามจำนวนรอบของขดลวดและปริมาณกระแส ดังสมการที่ 2.7 และดังรูปที่ 2.27

$$B = \frac{\mu_0 nI}{2a} \qquad (2.7)$$

n = จำนวนรอบของขดลวด

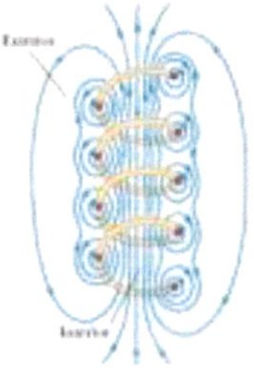


รูปที่ 2.27 สนามแม่เหล็กจากกระแสไฟฟ้าไหลในขดลวดสนามแม่เหล็ก [7]

2.9.1.3 สนามแม่เหล็กจากกระแสไฟฟ้าไหลในขดลวดโซลินอยด์

ความเข้มของสนามแม่เหล็กในขดลวดโซลินอยด์แปรผันตามจำนวนของขดลวด และปริมาณกระแส ดังสมการที่ 2.8 และรูปที่ 2.28

$$B = \mu_0 nI \tag{2.8}$$



รูปที่ 2.28 สนามแม่เหล็กจากกระแสไฟฟ้าไหลในขดลวดโซลินอยด์ [7]

2.9.1.4 ความเข้มของสนามแม่เหล็กไฟฟ้า จะขึ้นอยู่กับส่วนประกอบต่างๆ ดังนี้

1) จำนวนรอบของการพันเส้นลวดตัวนำ การพันจำนวนรอบของเส้นลวดตัวนำมากเกิดสนามแม่เหล็กมาก ในทางกลับกันถ้าพันจำนวนรอบน้อยการเกิดสนามแม่เหล็กก็น้อยตามไปด้วย

2) ปริมาณการไหลของกระแสไฟฟ้า ผ่านเส้นลวดตัวนำ กระแสไฟฟ้าไหลผ่านมากสนาม แม่เหล็กเกิดขึ้นมาก และถ้ากระแสไฟฟ้าไหลผ่านน้อยสนามแม่เหล็กเกิดน้อย

3) ชนิดของวัสดุที่ใช้ทำแกน ของแท่งแม่เหล็กไฟฟ้า วัสดุต่างชนิดกันจะให้ความเข็มของสนามแม่เหล็กต่างกัน เช่น แกนอากาศจะให้ความเข็มของสนามแม่เหล็กน้อยกว่าแกนที่ทำจากสารเฟอโรแมกเนติก (Ferromagnetic) หรือสารที่สามารถเกิดอำนาจแม่เหล็กได้ เช่น เหล็ก เฟอร์ไรท์ เป็นต้น สารเหล่านี้จะช่วยเสริมอำนาจแม่เหล็กในขดลวดทำให้มีความเข็มของสนามแม่เหล็กมากขึ้น

4) ขนาดของแกนแท่งแม่เหล็กไฟฟ้า แกนที่มีขนาดใหญ่จะให้สนามแม่เหล็กมาก ส่วนแกนที่มีขนาดเล็กจะให้สนามแม่เหล็กน้อย

2.9.2 ความเหนี่ยวนำไฟฟ้า

การเหนี่ยวนำแม่เหล็กไฟฟ้า ไมเคิล ฟาราเดย์ พบว่า เมื่อกระแสไฟฟ้าไหลผ่านสนามแม่เหล็กจะมีผลให้เกิดการเคลื่อนที่ โดยการเคลื่อนที่ของตัวนำในสนามแม่เหล็กจะก่อให้เกิดแรงเคลื่อนไฟฟ้าในตัวนำนั้น เรียกว่า การเหนี่ยวนำแม่เหล็กไฟฟ้าซึ่งจะเกิดขึ้นเสมอในตัวนำที่วางอยู่ในสนามแม่เหล็กที่เปลี่ยนแปลง

กฎการเหนี่ยวนำของฟาราเดย์ กล่าวว่า ขนาดของแรงเคลื่อนไฟฟ้าเหนี่ยวนำในตัวนำเป็นสัดส่วนกับอัตราการเปลี่ยนแปลงสนามแม่เหล็ก

กฎของเลนซ์ กล่าวว่า แรงเคลื่อนไฟฟ้าเหนี่ยวนำจะเกิดขึ้นเพื่อขัดขวางสาเหตุที่ทำให้เกิดการเปลี่ยนแปลง

กฎมือขวาของเฟรมมิง หรือกฎไดนาโม กล่าวว่าทิศของกระแสไฟฟ้าเหนี่ยวนำหาได้จากทิศของสนามแม่เหล็กและทิศการเคลื่อนที่โดยใช้มือขวา

2.9.2.1 การคำนวณค่าความเหนี่ยวนำ L แกนอากาศ

การคำนวณค่าความเหนี่ยวนำนั้น มีด้วยกันหลายวิธี และขึ้นอยู่กับปัจจัยหลายอย่าง ทั้งขนาด, รูปร่าง และชนิดของสายอากาศ

ค่าความเหนี่ยวนำในขดลวดลูปจตุรัส N รอบ หลายชั้น ดังสมการ 2.9 และอ้างอิงการคำนวณดังรูปที่ 2.29

$$L = 0.008aN^2\left\{2.303\log_{10}\left(\frac{a}{b+c}\right)+0.2235\left(\frac{b+c}{a}\right)+0.726\right\}\ (\mu H) \qquad (2.9)$$

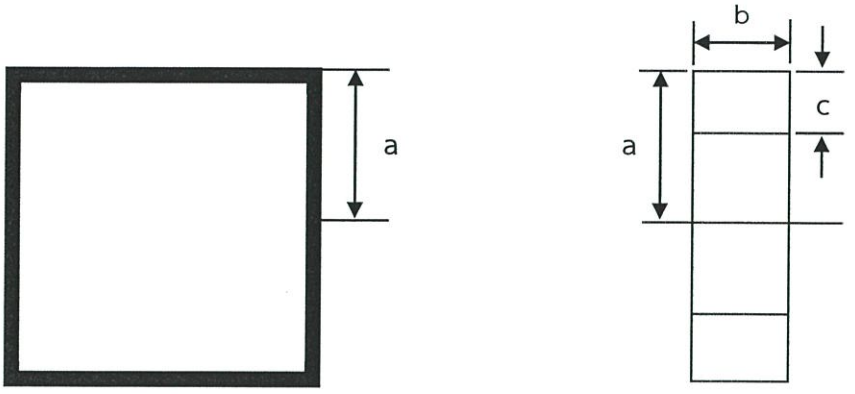
เมื่อ N = จำนวนรอบ

a = ความยาวถึงกึ่งกลางวัดจากด้านข้างของรอบ

b = ความยาวรอบ

c = ความลึกรอบ

หมายเหตุ : ความยาวทั้งหมดหน่วยเซนติเมตร (cm)



รูปที่ 2.29 ภาพมุมบนและมุมข้าง สายอากาศจตุรัส อ้างอิงเพื่อใช้ในการคำนวณ

ค่าความเหนี่ยวนำในขดลวดลูปผืนผ้า N รอบ หลายชั้นดังสมการที่ 2.10 และอ้างอิงการคำนวณดังรูปที่ 2.30

$$L = \frac{0.0276\,(CN)^2}{1.908C + 9b + 10h}\ (\mu H) \qquad (2.10)$$

เมื่อ N = จำนวนรอบ

C = x+y+2h

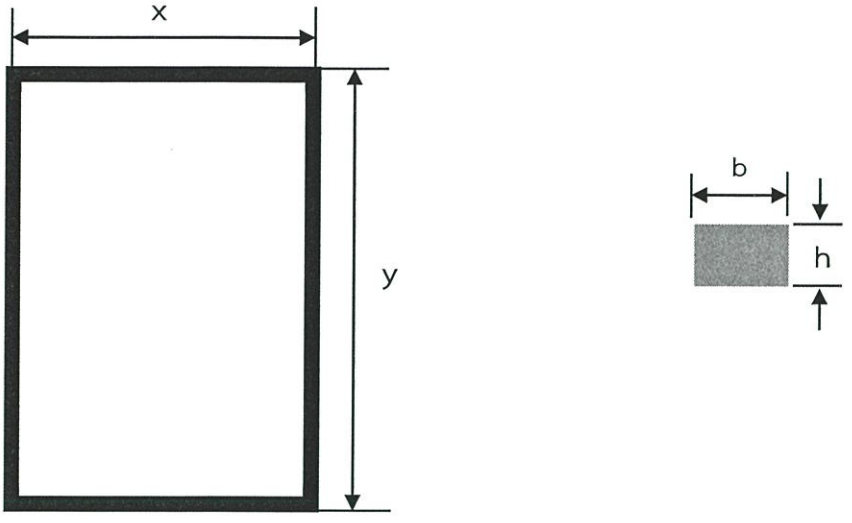
x = ความกว้างของขดลวด

y = ความยาวของขดลวด

b = ความกว้างของขอบ

h = ความสูงของขอบ

หมายเหตุ : ความยาวทั้งหมดหน่วยเซนติเมตร (cm)



รูปที่ 2.30 ภาพมุมบนและมุมข้าง สายอากาศผืนผ้า อ้างอิงเพื่อใช้ในการคำนวณ

2.9.3 วงจรเรโซแนนซ์

วงจรเรโซแนนซ์ คือ การใช้ตัวต้านทาน ตัวเหนี่ยวนำ และตัวเก็บประจุ ทำหน้าที่เป็นวงจรกรองความถี่ โดยที่ความถี่เรโซแนนซ์จะกำจัดฮาร์โมนิกอันดับสูง ๆ ออกไปเหลือเฉพาะสัญญาณรูปไซน์ ความต้านทานเชิงซ้อนรวมของวงจรมีคาเฉพาะจำนวนจริง หรือกลาวอีกนัยหนึ่งวงจร RLC จะเสมือนมีแต R อยางเดียวเทานั้น คาที่เปนจำนวนจินตภาพเปนศูนย์ ข้อดีที่สำคัญของวงจรเรโซแนนซ์คือ ลดการสูญเสียในสวิตช์ ข้อเสียที่สำคัญของวงจรเรโซแนนซ์คือ ค่าที่ใกล้ความถี่เรโซแนนซ์กระแสไฟฟ้าไหลเวียน หรือแรงดันไฟฟ้าตกคร่อมวงจรอาจมีค่าสูง ทำให้เกิดการสูญเสียในองค์ประกอบวงจร และเกิดความเค้นที่สวิตช์สูงขึ้น พลังงานที่สะสมในตัวเหนี่ยวนำ และตัวเก็บประจุมีค่าสูงด้วย กรณีที่กระแสไฟฟ้าไหลเวียนมีค่าสูง จะทำให้เกิดการสูญเสียในช่วงการนำกระแสไฟฟ้าก็มีค่าสูงด้วย วงจรเรโซแนนซ์แบ่งได้เป็น 2 ชนิด

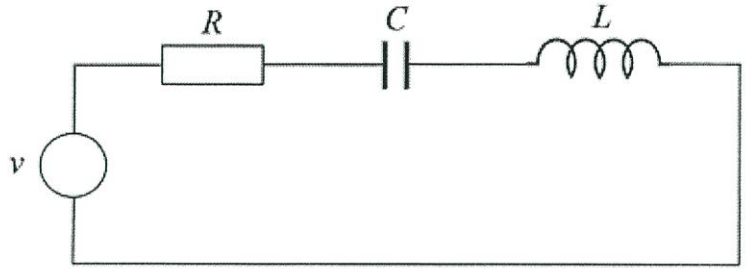
2.9.3.1 วงจรเรโซแนนซ์แบบอนุกรม

คุณสมบัติของวงจร คือ มีค่าอิมพีแดนซ์ต่ำที่ความถี่เรโซแนนท์ ทำให้กระแสไฟฟ้าไหลได้สูงสุด กรณีตัวต้านทาน ตัวเหนี่ยวนำ และตัวเก็บประจุต่อกันแบบอนุกรม เมื่อวงจรเกิดสภาวะเรโซแนนซ์จะทำให้ค่า $X_L = X_C$ เมื่อค่าของ $X_L = X_C$ แล้วจะทำให้เกิดการหักล้างเป็นศูนย์เนื่องจากค่าของ X_L เป็นบวก ส่วนค่าของ X_C เป็นลบ เมื่อ $X_L - X_C = 0$ ดังนั้นในวงจรจึงเหลือค่าความต้านทานของตัวต้านทานเท่านั้น ความต้านทานเชิงซ้อนของวงจรจึงมีค่าเท่ากับค่าความต้านทานของตัวต้านทานในวงจร $Z = R$ ดังรูปที่ 2.31

โดยที่ X_L คือ อินดัคทีฟรีแอคแตนซ์ : [Ω]

X_C คือ คาปาซีทีฟรีแอคแตนซ์ : [Ω]

Z คือ อิมพีแดนซ์ : [Ω]



รูปที่ 2.31 วงจรเรโซแนนซ์แบบอนุกรม [8]

จากวงจรเรโซแนนซ์แบบอนุกรม ดังรูปที่ 2.40 จะได้สมการที่ 2.11

$$Z=R+jX_L-jX_C$$
$$Z=\sqrt{R^2+(X_L-X_C)^2} \quad (2.11)$$

เมื่อ $X_L = X_C$ และ $X_L - X_C = 0$ ดังนั้นจะได้สมการที่ 2.12

$$Z=\sqrt{R^2+(0)^2}$$
$$Z=R \quad (2.12)$$

เมื่อ $X_L=2\pi f_0 L$ และ $X_C=\frac{1}{2\pi f_0 C}$ จะได้สมการที่ 2.13

$$2\pi f_0 L=\frac{1}{2\pi f_0 C} \quad (2.13)$$

$$f_0=\frac{1}{2\pi\sqrt{LC}} \quad (2.14)$$

เมื่อ f_0 คือความถี่เรโซแนนซ์ดังสมการที่ 2.14 และค่ากระแสไฟฟ้าที่ไหลในวงจรขณะเกิดเรโซแนนซ์ เรียกว่ากระแสไฟฟ้าเรโซแนนซ์ ค่ากระแสไฟฟ้าเรโซแนนซ์ I_0 จะหาได้ดังสมการที่ 2.15

$$I_0=\frac{V}{Z_{SO}}=\frac{V}{R} \quad (2.15)$$

ค่าโวลเตจ V_R, V_C และ V_Lที่ตกคร่อมขั้วตัวต้านทาน ตัวเหนี่ยวนำ และตัวเก็บประจุ จะมีค่าดังสมการที่ 2.16, 2.17 และ2.18 ตามลำดับ

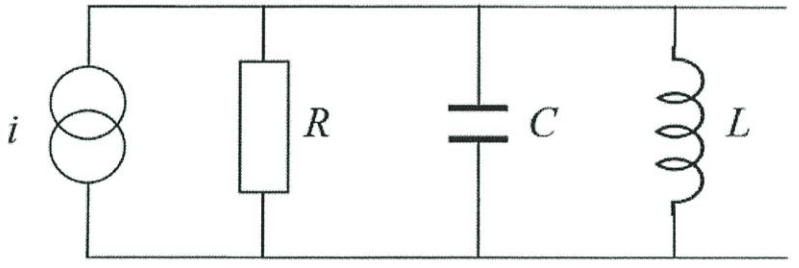
$$V_R=RI_0=V \tag{2.16}$$

$$V_C=\frac{1}{\omega_0 C}I_0=\frac{V}{\omega_0 CR} \tag{2.17}$$

$$V_L=\omega_0 LI_0=\frac{\omega_0 L}{R}V \tag{2.18}$$

2.9.3.2 วงจรเรโซแนนซ์แบบขนาน

เมื่อนำตัวต้านทาน ขดลวดเหนี่ยวนำ และตัวเก็บประจุมาต่อขนานกันวงจรจะแสดงคุณลักษณะของวงจรออกมา คือ ถ้า $X_L > X_C$ กระแสไฟฟ้า I_C จะมากกว่า I_Lแต่ถ้า $X_L < X_C$ กระแสไฟฟ้า I_L จะมากกว่า I_C และที่กระแสไฟฟ้า I_L เท่ากับ I_C เรียกคุณลักษณะของวงจรนี้ ว่า สภาวะเรโซแนนซ์ กระแสสามารถแยกไหลได้สามเส้นทาง โดยมีหลักการที่สำคัญคือ ความต่างศักย์คร่อมอุปกรณ์แต่ละชนิดมีค่าเท่ากัน ดังรูปที่ 2.32



รูปที่ 2.32 วงจรเรโซแนนซ์แบบขนาน [8]

วงจรตัวต้านทาน ขดลวดเหนี่ยวนำ และตัวเก็บประจุต่อขนานกันเมื่อเกิดสภาวะเรโซแนนซ์ค่า $X_L = X_C$ เป็นผลให้กระแสไฟฟ้าที่ไหลผ่านขดลวดเหนี่ยวนำ มีค่าเท่ากับกระแสไฟฟ้าที่ไหลผ่านตัวเก็บประจุ $I_L = I_C$ กระแสไฟฟ้าที่ไหลผ่านขดลวดเหนี่ยวนำ ตัวเก็บประจุนี้จะสมดุลกันและมีทิศทางตรงกันข้ามกันจึงหักล้างกันไปหมด $I_L - I_C = 0$ ในวงจรจึงเหลือแต่กระแสไฟฟ้าที่ไหลผ่านตัวต้านทาน ในวงจรแบบขนานความต้านทานเชิงซ้อนของวงจร ดังสมการที่ 2.19

$$Z=\frac{RX_LX_C}{\sqrt{X_L{}^2X_C{}^2+R^2(X_L-X_C)^2}} \tag{2.19}$$

เมื่อวงจรอยู่ในสภาวะเรโซแนนซ์ $X_L = X_C$ ดังนั้น $X_C - X_L = 0$ และจะเป็นดังสมการ 2.12, 2.13 และ 2.14 เช่นเดียวกันกับวงจรเรโซแนนซ์แบบอนุกรม

2.10 NETPIE

NETPIE (Network Platform for Internet of Everything) เป็นระบบคราวด์แพลตฟอร์มที่ให้บริการลักษณะของเครื่องคอมพิวเตอร์ ซึ่งให้บริการที่เก็บเว็บไซต์ให้บริการสำหรับ Internet of Things (IoT) คือ การที่อุปกรณ์ต่างๆ สิ่งต่างๆ ได้ถูกเชื่อมโยงทุกสิ่งทุกอย่างสู่โลกอินเตอร์เน็ต โดยเปิดให้บุคคลทั่วไปใช้งานโดยมีเว็บท่าหรือศูนย์รวมลิ้งค์ เพื่อให้สามารถเชื่อมโยงไปยังส่วนต่างๆ ที่เกี่ยวข้องกันในองค์กร ผู้ใช้งานสามารถลงทะเบียนและจัดการตัวตนและสิทธิ์ของแอปพลิเคชั่นและอุปกรณ์ได้ แพลตฟอร์มนี้ยังช่วยให้นักพัฒนาซอฟต์แวร์เข้าถึงอุปกรณ์อิเล็กทรอนิกส์ต่างๆได้ง่ายขึ้น ผ่านไลบรารี่สาเร็จรูปที่แพลตฟอร์มเตรียมไว้ให้ ดังนั้นบริการแพลต ฟอร์ม NETPIE จึงเป็นเสมือนสะพานเชื่อมระหว่างนักพัฒนาฮาร์ดแวร์และนักพัฒนาซอฟต์แวร์

NETPIE เป็น Middleware (software computer) ที่คอยช่วยเหลือดูแลแอปพลิแคชั่นที่รันอยู่บนตัวเชื่อมระหว่าง APP และ OS ซึ่งช่วยให้นักพัฒนา สามารถเชื่อมต่อสื่อสารกับภายนอกได้ง่ายขึ้น ทำให้ลดภาระในการดูแลรายละเอียดรอบข้างและเน้นแต่งานหลักที่ต้องการได้ มีหัวใจหลักเป็น Distributed MQTT Brokers โดยมีการติดต่อสื่อสารและทางานร่วมกันผ่านวิธีการส่งข้อความแบบ Publish/Subscribe โครงสร้างสถาปัตยกรรมของ NETPIE ทุกๆองค์ประกอบเป็นคลาวด์ จึงสามารถขยายตัวได้อย่างอัตโนมัติ สามารถดูแลและซ่อมแซมตัวเองได้อัตโนมัติเมื่อระบบมีปัญหาและมีการบริหารจัดการระบบแบบ Plug-and-Play

การสร้าง, ดูแล และรักษาความปลอดภัยของข้อมูลในช่องทางสื่อสารระหว่างอุปกรณ์กับ NETPIE ใช้ Microgear (Client Library) โดยในการส่งข้อมูลของ Microgear เป็น Open Source

2.10.1 ประโยชน์ของ NETPIE

1. ลดการใช้ทรัพยากรของการเชื่อมต่อ Microgear Library ในอุปกรณ์ NETPIE มีหน้าที่ดูแลเชื่อมต่อของอุปกรณ์ให้สามารถติดต่อสื่อสารกันได้ จะทำการลดความยืดหยุ่นของระบบและออกแบบให้อุปกรณ์ถูกค้นพบและเข้าสู่บริการโดยอัตโนมัติ

2. ลดภาระด้านความปลอดภัยของข้อมูล การออกแบบการเข้าถึงของ NETPIE ออกแบบให้ผู้ใช้สามารถออกแบบได้เองทั้งหมด (ระดับ Fine Grain)

3. ยืดหยุ่นต่อการขยายระบบ ในสถาปัตยกรรมของ NETPIE มีความยืดหยุ่นและคล่องตัวสูงในการขยายตัว ในการทำงานของโมดูลต่างๆยังมีการทำงานแยกจากกัน มีสื่อสารกันด้วยวิธี Asynchronous Messaging ช่วยให้แพลตฟอร์มมีความน่าเชื่อถือ

2.10.2 MICROGEAR

ซอฟต์แวร์ไลบรารี่ของ NETPIE คือ Microgear ทำตัวเสมือนตัวกลางและผู้ช่วยในการสร้างและดูแลการเชื่อมต่อ ให้มีความเสถียร ให้การสื่อสารแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์เป็นไป

อย่างราบรื่น โดยทำการติดตั้งอยู่บนอุปกรณ์ที่ต้องการเชื่อมต่อสื่อสารผ่านคลาวด์ บทบาทหน้าที่ของ Microgear สามารถแบ่งออกเป็น 4 ด้านคือ

1) ด้านการสื่อสาร (Communication) ในการแลกเปลี่ยนข้อมูลจะใช้โพรโทคอล MQTT ในการสื่อสาร โดยที่ Microgear จะทำหน้าที่เป็นผู้ช่วยในการสร้างการเชื่อมต่อไปยังคลาวด์ของ NETPIE และคอยตรวจสอบสถานะของการเชื่อมต่อ เมื่อเชื่อมต่อมีปัญหา Microgear สามารถช่วยเชื่อมต่อให้ใหม่เพื่อให้การสื่อสารเป็นไปได้อย่างราบรื่น และช่วยอำนวยความสะดวกในการสร้างช่องทางการสื่อสารแบบเข้ารหัส

2) ด้านการยืนยันตัวตน (Authentication) Microgear จะช่วยยืนยันตัวตนของอุปกรณ์กับคลาวด์ โดยใช้ข้อมูลประกอบกันสามส่วนคือ AppID, App Key และToken สำหรับการสร้างการเชื่อมต่อกับ NETPIE

3) ด้านการขออนุญาตสิทธิ์ (Authorization) การสร้างการเชื่อมต่อและการยืนยันตัวตนคลาวด์ของ NETPIE จะเป็นผู้ออกใบอนุญาต (Token) ที่ระบุว่าอุปกรณ์ สามารถสื่อสารได้กับอุปกรณ์ตัวใดบ้าง โดยต้องอยู่ภายใต้กลุ่ม AppID เดียวกันจึงจะมีสิทธิ์สื่อสารกันได้

4) ด้านการประสานงาน (Coordination) Microgear มีฟังก์ชั่นที่ช่วยให้อุปกรณ์ต่างๆ ภายในกลุ่ม AppID เดียวกันทราบสถานะของกันและกัน รวมถึงทราบการเปลี่ยนแปลงสถานะของอุปกรณ์ที่สนใจติดตาม จากข้อมูลดังกล่าวผู้ใช้สามารถกำหนดบทบาทหน้าที่ให้อุปกรณ์ในกลุ่มตามสถานะของอุปกรณ์อื่นๆ ในกลุ่ม เช่น หากเป็นอุปกรณ์ตัวแรกในกลุ่ม ให้ทำหน้าที่เป็นหัวหน้ากลุ่ม เป็นต้น

Microgear ถูกพัฒนาให้ทำงานได้กับอุปกรณ์ที่หลากหลาย ส่วนของซอฟต์แวร์มี Microgear ให้เลือกใช้กับ Programming Language ได้แก่ Node.js, Python, HTML5, Java Android และC# สำหรับอุปกรณ์ฮาร์ดแวร์ประเภทไมโครคอนโทรลเลอร์ Microgear เปรียบเสมือน Firmware ซึ่งรองรับ Arduino with Ethernet Shield (ใช้ได้กับ Arduino Mega) และ Microgear สำหรับ WiFi ไมโครคอนโทรลเลอร์ ESP8266 Microgear

2.10.3 Events ของ Microgear

การทำงานของ Microgear เป็นการทำงานโดยให้โปรแกรมจัดการกับเหตุการที่เกิดขึ้นเอง (Event-Driven) ซึ่งนั่นหมายถึงจะต้องมาดูเหตุการณ์ใดเกิดขึ้นกับแอปพลิเคชันที่เราจะเขียนบ้าง จำแนกออกมาทีละเหตุการณ์ หลังจากนั้นทำการเขียนโค้ดเข้าไปจัดการกับแต่ละเหตุการณ์เหล่านั้น โดยตอบสนองต่อเหตุการณ์ต่างๆ ด้วยการเขียน Callback Function ซึ่งชนิดของเหตุการณ์ที่สามารถเกิดขึ้น มีดังนี้

1) connected เกิดขึ้นเมื่อ Microgear เชื่อมต่อกับ NETPIE สาเร็จ
2) closed เกิดขึ้นเมื่อ Microgear ปิดการเชื่อมต่อกับ NETPIE
3) error เกิดขึ้นเมื่อมีความผิดพลาดเกิดขึ้นกับ Microgear

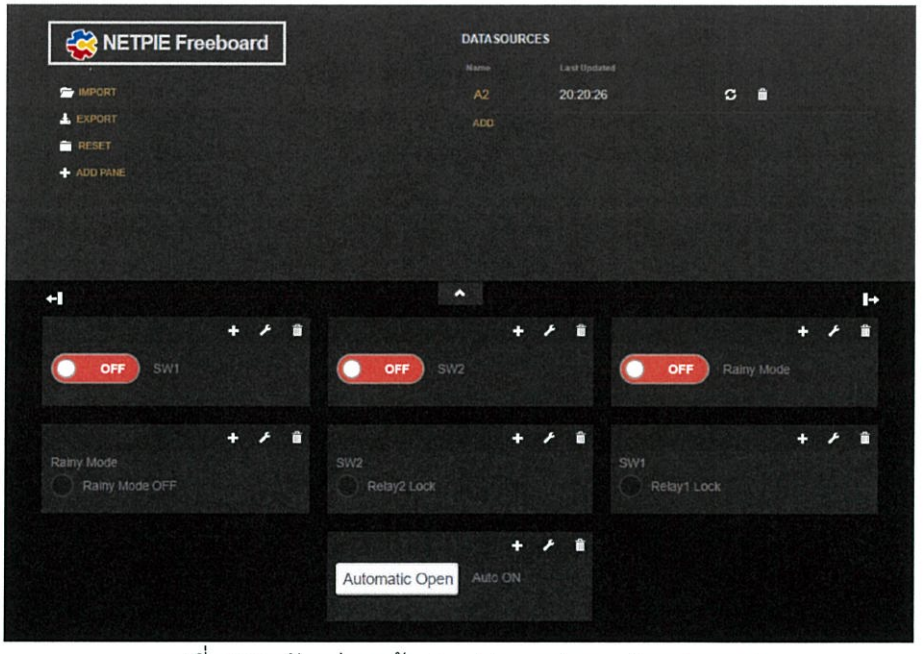
4) message เกิดขึ้นเมื่อมีข้อความเข้ามาที่อุปกรณ์

5) present เกิดขึ้นเมื่อมีอุปกรณ์ใน AppID เดียวกันเชื่อมต่อเข้ามาบน NETPIE

6) absent เกิดขึ้นเมื่อมีอุปกรณ์ใน AppID เดียวกันหายไปจากการเชื่อมต่อกับ NETPIE

2.10.4 FREEBOARD

Freeboard เป็นเว็บแอปพลิเคชั่น (Web Application) โดยเป็นแอปพลิเคชั่นที่ถูกเขียนขึ้นมาเป็นโปรแกรมหรือซอฟต์แวร์ที่ใช้ท่องเว็บหรือใช้ดูข้อมูลที่อยู่ในเว็บไซต์ มีการปรับแต่งให้แสดงผลแต่ส่วนที่จำเป็น เพื่อเป็นการลดทรัพยากรในการประมวลผล ของตัวเครื่องสมาร์ทโฟน หรือแท็บเล็ต โดยผู้ใช้งานยังสามารถใช้งานผ่านอินเทอร์เน็ตและอินทราเน็ตในความเร็วต่ำได้ Freeboard สามารถสร้าง Dashboard เพื่อแสดงผลสาหรับ IoT แอปพลิเคชั่นโดยสามารถใช้เป็นกระดานส่วนตัว สามารถวางปุ่มกดสวิตช์ไว้ใช้สาหรับควบคุมอุปกรณ์ หรือวางหน้าปัดเพื่อแสดงผลข้อมูลต่างๆ หรือแสดงผลเป็นกราฟจากอุปกรณ์ ในการปรับแต่งหน้าทำได้โดยป้อนข้อมูลเข้าหรือกำหนดคำสั่ง โดยที่ผู้ใช้ไม่จาเป็นต้องเขียน HTML Web Page เอง และการอัพเดทข้อมูลแบบ Real-time มีความเสถียรและเชื่อถือได้ และเป็น Open-Source ที่นักพัฒนาสามารถต่อยอดให้ดียิ่งขึ้นได้ ตัวอย่าง Freeboard แสดงดังรูปที่ 2.33



รูปที่ 2.33 ตัวอย่างหน้า Dashboard ของ Freeboard

NETPIE Freeboard ใช้สำหรับการควบคุมและการแสดงผล (Visualization) ข้อมูลที่ดึงมาจากอุปกรณ์ที่ต่อกับ NETPIE การใช้งาน NETPIE Freeboard นั้นสามารถใช้ซอฟต์แวร์ที่ท่อง

เว็บโดยเปิดไฟล์ index.html ที่ได้จากการติดตั้ง NETPIE Freeboard หรือสามารถใช้ Freeboard ผ่านทางหน้าเว็บของ NETPIE

2.10.5 Internet of Things

Internet of Things หรือIoT คือ สภาพแวดล้อมอันประกอบด้วยสรรพสิ่งที่สามารถสื่อสารและเชื่อมต่อกันได้ผ่านโพรโทคอลการสื่อสารทั้งแบบใช้สายและไร้สาย โดยสรรพสิ่งต่างๆ มีวิธีการระบุตัวตนได้ รับรู้บริบทของสภาพแวดล้อมได้ และมีปฏิสัมพันธ์โต้ตอบและทำงานร่วมกันได้ ความสามารถในการสื่อสารของสรรพสิ่งนี้จะนำไปสู่นวัตกรรมและบริการใหม่อีกมากมา ตัวอย่างเช่น เซ็นเซอร์ภายในบ้านตรวจจับการเคลื่อนไหวของผู้อยู่อาศัย และส่งสัญญาณไปสั่งเปิด-ปิดสวิตซ์ไฟตามห้องต่างๆ ที่มีคนหรือไม่มีคนอยู่ อุปกรณ์วัดสัญญาณชีพของผู้ป่วย, ผู้สูงอายุ และส่งข้อมูลไปยังบุคลากรทางการแพทย์ หรือส่งข้อความเรียกหน่วยกู้ชีพ หรือรถฉุกเฉิน เป็นต้น

นอกจากนี้ IoT จะเปลี่ยนรูปแบบและกระบวนการผลิตในภาคอุตสาหกรรมไปสู่ยุคใหม่ หรือที่เรียกว่า Industry 4.0 ที่จะอาศัยการเชื่อมต่อสื่อสาร และทำงานร่วมกันระหว่างเครื่องจักร มนุษย์ และข้อมูล เพื่อเพิ่มอำนาจในการตัดสินใจที่รวดเร็ว และมีความถูกต้องแม่นยำสูง โดยที่ข้อมูลทั้งหลายที่เก็บจากเซ็นเซอร์ที่ใช้ตรวจวัดตัวอุปกรณ์ และสภาพแวดล้อมจะถูกนำมาวิเคราะห์ ให้ได้ผลลัพธ์เพื่อนำไปปรับปรุงกระบวนการผลิตได้อย่างทันที นอกจากการข้ามขีดจำกัดเรื่องเวลาแล้ว ระบบควบคุม หรือระบบวิเคราะห์ข้อมูล อาจไม่ได้อยู่ในที่เดียวกันกับเครื่องจักร แต่สามารถควบคุมสั่งการได้โดยไร้ขีดจำกัดเรื่องสถานที่

เทคโนโลยีที่ทำให้ IoT เกิดขึ้นได้จริงและสร้างผลกระทบในวงกว้างได้ แบ่งออกเป็นสามกลุ่มได้แก่ เทคโนโลยีที่ช่วยให้สรรพสิ่งรับรู้ข้อมูลในบริบทที่เกี่ยวข้อง, เทคโนโลยีที่ช่วยให้สรรพสิ่งมีความสามารถในการสื่อสาร และเทคโนโลยีที่ช่วยให้สรรพสิ่งประมวลผลข้อมูลในบริบทของตน เช่น เทคโนโลยีการประมวลผลแบบคลาวด์ และเทคโนโลยีการวิเคราะห์ข้อมูลขนาดใหญ่ หรือBig Data Analytics

2.11 ANDROID STUDIO

Android Studio คือ ระบบปฏิบัติการของอุปกรณ์เคลื่อนที่ เป็นโปรแกรมสำหรับพัฒนาแอปพลิเคชั่นแอนดรอยด์ โดยเป็น IDE (Integrated Development Environment) Tool จาก Google เป็นการพัฒนาจากแนวคิดพื้นฐานมาจาก InteliJ IDEA โดยมีลักษณะการทำงานคล้ายกับ Eclipse และAndroid ADT Plugin เป็นโปรแกรมที่ออกแบบมาพัฒนาให้แอปพลิเคชั่นมีประสิทธิภาพมากขึ้น ทั้งด้านการออกแบบ GUI ที่ช่วยให้สามารถ Preview มุมมองที่แตกต่างกันบน Smart Phone แต่ละรุ่น สามารถแสดงผลบางอย่างได้ทันทีโดนไม่ต้องทำการรันแอปพลิเคชั่น บน Emulator

รูปที่ 2.34 Android Studio

ในส่วนของ Android Project จะมีไฟล์ 2 ส่วนแนบมาด้วย คือ

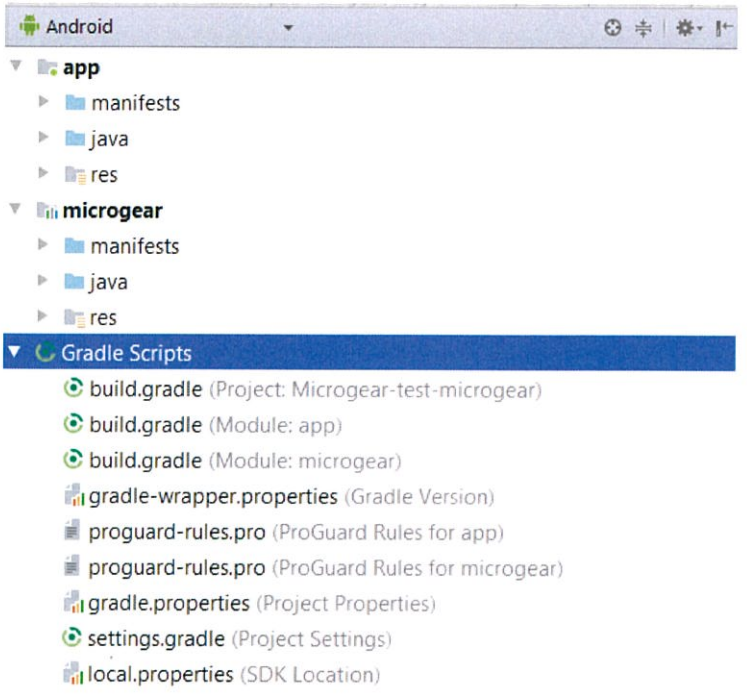
1. Folder app: จัดเก็บ Source Code โดยภายในจะแบ่งเป็น พื้นที่จัดเก็บ 3 ส่วน ดังนี้

    1.1 manifests: จัดเก็บ AndroidManifest.xml file.

    1.2 java: จัดเก็บ Java source code files, including JUnit test code.

    1.3 res: จัดเก็บ non-code resources เช่น XML layouts, UI strings, and bitmap images

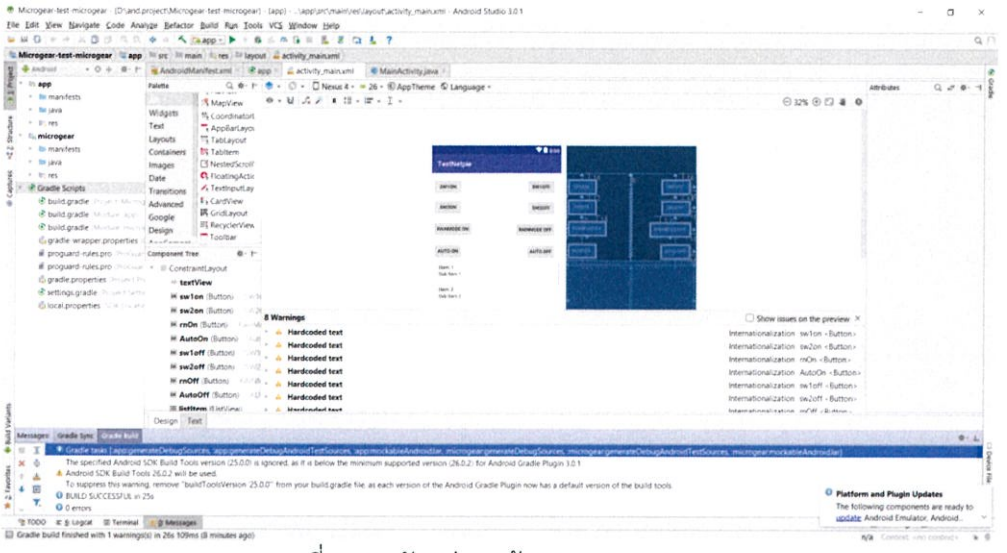
2. Gradle Scripts: จัดการเรื่องการ Build Android App



รูปที่ 2.35 ไฟล์เพิ่มเติมใน Android Project

The User Interface

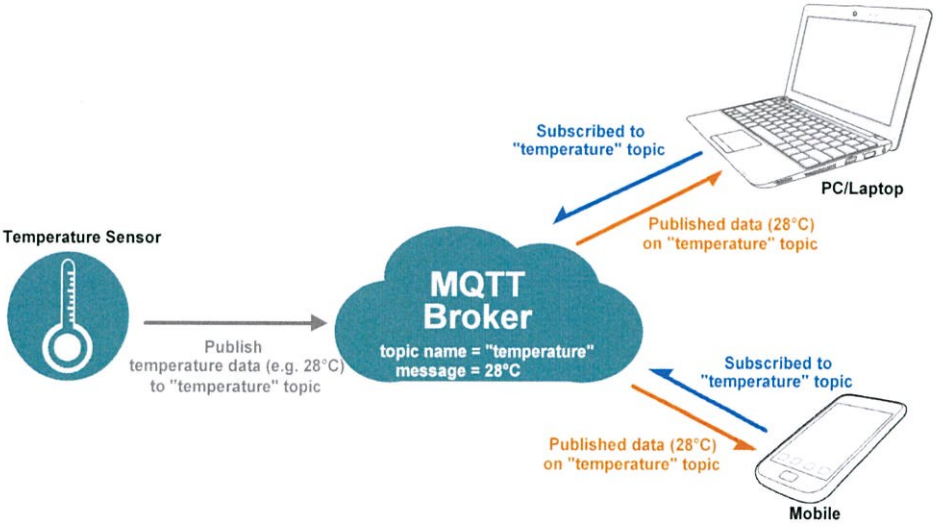
1. toolbar: แถบเครื่องมือที่ใช้บ่อยๆ
2. navigation bar: แถบเครื่องมือที่บอก location ของไฟล์ต่างๆ และทำให้เข้าถึงไฟล์ได้ง่ายขึ้น
3. editor window: พื้นที่แสดงโค้ด และกราฟฟิกต่างๆ
4. tool window bar: แถบหน้าต่างอื่นๆ ที่เกี่ยวข้อง ซึ่งจะวางอยู่รอบๆ IDE
5. tool windows: หน้าต่างที่ใช้จัดการ project เช่น แสดงไฟล์ในโปรเจ็ค แสดงการ debug แสดง console ต่างๆ ซึ่งจะวางอยู่รอบๆ editor
6. status bar: แสดงสถานะของ project เช่น error หรือ warning



รูปที่ 2.36 ตัวอย่างหน้า User Interface

2.12 MESSAGE QUEUING TELEMETRY TRANSPORT

Message Queuing Telemetry Transport (MQTT) เป็นโปรโตคอลสำหรับการเชื่อมต่อแบบอุปกรณ์กับอุปกรณ์ โดยใช้ร่วมกับเทโนโลยีอินเตอร์เน็ตเชื่อมต่อกับอุปกรณ์ต่างๆ เช่น โทรศัพท์มือถือ หลอกไฟ ตู้เย็น เข้ากับอินเตอร์เน็ตโดยที่ทำให้สามารถสื่อสารกับอุปกรณ์ต่างๆได้ โดยผ่านเครือข่ายอินเตอร์เน็ตจึงทำให้สะดวกสะบายสำหรับผู้ใช้งานที่สามารถควบคุมจากที่อื่นได้ ในการออกแบบโปรโตคอลนี้ออกแบบมาเพื่อใช้งานกับอุปกรณ์อิเล็กทรอนิกส์ขนาดเล็ก รับส่งข้อมูลในเครือข่ายขนาดเล็ก แบนร์วิธต่ำ โดยใช้หลักการแบบ publisher/subscriber [10]



รูปที่ 2.37 การทำงานของ MQTT [11]

MQTT เป็นโปรโตคอลที่รันอยู่บน Application Layer ตาม OSI Model ที่รันบน TCP/IP สถาปัตยกรรมของ MQTT เป็นแบบ Client/Server มี Topology เป็นแบบ hub-and-spoke ในการสื่อสารของโปรโตคอลนี้ประกอบด้วย 3ส่วน คือ Server (Broker), Client และ Topic

1. Broker มีหน้าที่เป็นเสมือนท่อส่งข้อมูลในการรับส่ง 'Message' ระหว่าง Client โดยทำการสร้างเส้นทาง (Routing) ด้วยหัวข้อ (Topic) โดยที่ Client ต้องทำการ Subscribe ใน Topic ที่ตัวเองต้องการ จากนั้น Broker ก็จะส่งข้อมูลทั้งหมดที่ถูก Publish ใน Topic นั้นๆให้ โดยที่ Client สื่อสารกันโดยที่ไม่รู้จักกัน ซึ่งถือเป็นข้อดีเมื่อต้องการขยายเครือข่ายก็สามารถดำเนินการได้ง่ายและหน้าที่ที่สำคัญมากอีกอย่างของ broker คือรักษาความปลอดภัยของเครือข่าย เช่น การทำ Authorization และ Authentication ของ Client ครับ ปัจจุบัน MQTT Broker เปิดให้ใช้งานหรือดัดแปลงอยู่หลายเจ้าด้วยกันตามด้านล่างนี้เลยครับ ในที่นี้จะเน้นเจ้าที่เป็นของคนไทย นั่นก็คือ NETPIE นั้นเองครับ
2. Client สามารถเป็นได้ทั้ง Publisher หรือ Subscriber ที่เชื่อมต่อแบบรวมศูนย์ไปยัง Broker ซึ่งสามารถเชื่อมต่อได้ทั้งแบบ persistent ที่ทำการสร้าง session ค้างไว้เปิดตลอดเวลาเพื่อติดต่อกับ Broker ซึ่งตรงกันข้ามกับ client ที่เชื่อมต่อแบบ transient ซึ่ง Broker ไม่สามารถติดตามสถานะได้
3. Topic เปรียบเสมือน address (endpoint) ที่ client ทำการเชื่อมต่อเพื่อรับส่งข้อความระหว่างกันบน Broker

ข้อดีของการใช้ MQTT Protocol เหมาะสำหรับนำไปใช้กับการให้บริการแบบรวมศูนย์ เพราะถูกออกแบบให้เหมาะกับการกระจายข้อมูลแบบ อย่างเช่นแอปพลิเคชั่น IoT Platform และระบบยังสามารถทำการสร้าง session แลกเปลี่ยนข้อมูลกันได้โดยไม่ต้องทำการตั้งค่า NAT ให้วุ่นวาย รวมไปถึงนักพัฒนาสามารถนำไปใช้กับร่วมกับ TLS/SSL เพื่อเพิ่มความปลอดภัยในการรับส่งข้อมูล แต่ยังมีข้อเสียสำหรับอุปกรณ์ที่มีทรัพยากรจำกัดเนื่องจาก client ทุกตัวต้องรองรับ TCP และทำการสร้างการเชื่อมต่อกับ broker ไว้ตลอดเวลา ซึ่งอาจเกิดปัญหาได้หากอยู่ในเครือข่ายที่ไม่เสถียร

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

ในส่วนของการออกแบบและการจัดทำปริญญานิพนธ์จะนำทฤษฎีจากบทที่ 2 มาใช้เป็นความรู้พื้นฐาน โดยเนื้อหาในบทนี้จะกล่าวถึงภาพรวมของระบบและอธิบายรายละเอียดต่างๆ ของการออกแบบ รวมถึงขั้นตอนการทดลองเพื่อทดสอบว่าระบบทำงานได้ตามต้องการ

3.1 การออกแบบ

3.1.1 การควบคุมกลอนไฟฟ้า

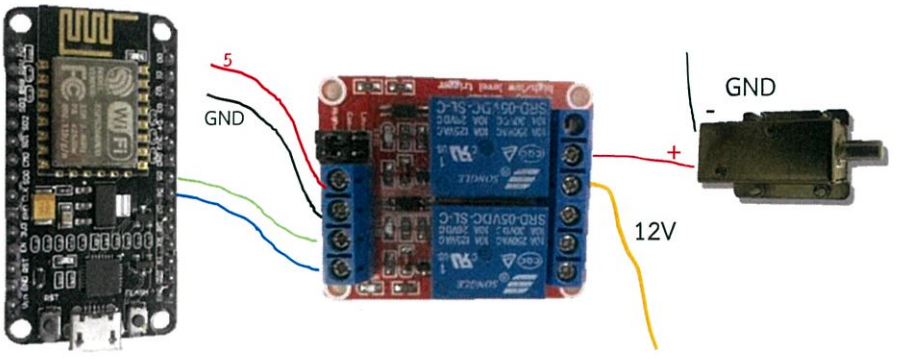
3.1.1.1 การควบคุมกลอนไฟฟ้าผ่านโมดูลรีเลย์

จากการศึกษาการทำงานของ NodeMCU และรีเลย์เรียบร้อยแล้ว ต่อมาจะเป็นการเชื่อมรีเลย์ เข้ากับกลอนไฟฟ้าผ่านบ NodeMCU นั่นเอง การเชื่อมต่อ NodeMCU, โมดูลรีเลย์ และกลอนไฟฟ้าเข้าด้วยกัน เพื่อควบคุมการเปิด-ปิดของกลอนไฟฟ้าผ่าน NodeMCU แสดงดังรูปที่ 3.1 โดยจะอธิบายเป็นขั้นตอนการเชื่อมต่อโมดูลรีเลย์และกลอนไฟฟ้า ดังนี้

1. นำขั้วบวกของกลอนไฟฟ้าต่อเข้ากับขาคอมมอนของรีเลย์ตัวที่ 1
2. นำขั้วลบของกลอนไฟฟ้าต่อเข้ากับไฟลบ (GND) ต่อเข้ากับไฟลบที่เป็นกราวน์ของไฟ 12 โวลต์
3. นำขา NO ของรีเลย์ตัวที่ 1 ต่อเข้ากับไฟบวก 12 โวลต์ เพื่อจ่ายแรงดันให้กับกลอนไฟฟ้า

ส่วนด้าน NodeMCU กับโมดูลรีเลย์ ทำได้โดย

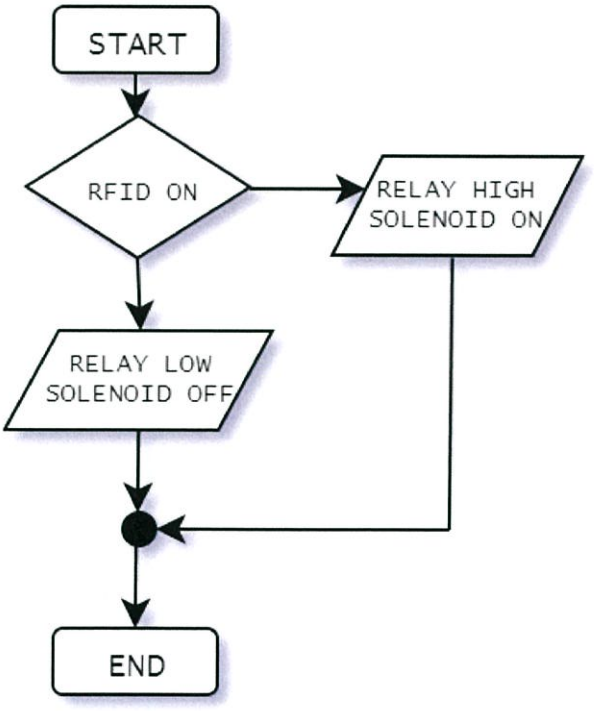
1. นำพิน D5 และD6 ที่ใช้สำหรับอินพุต และเอาท์พุตแบบดิจิตอล เชื่อมกับ IN1 และ IN2 ของโมดูลรีเลย์ ตามลำดับ ตามเส้นสีเขียวและสีฟ้า
2. นำแรงดัน 5 โวลต์ เชื่อมเข้ากับไฟบวกของโมดูลรีเลย์
3. นำ GND ของแรงดัน 5 โวลต์ เชื่อมเข้ากับไฟลบของโมดูลรีเลย์



รูปที่ 3.1 การเชื่อมต่อ NodeMCU โมดูลรีเลย์ และกลอนไฟฟ้า

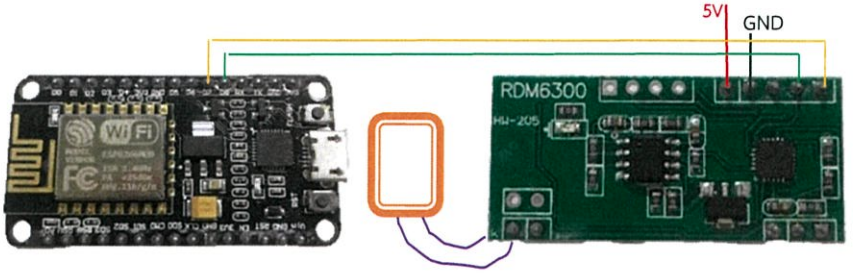
3.1.1.2 ควบคุมการทำงานของกลอนไฟฟ้าผ่านอาร์เอฟไอดี

การออกแบบการควบคุมการทำงานของกลอนไฟฟ้าผ่านอาร์เอฟไอดีนั้น จะเป็นการเชื่อมต่อเพิ่มเติม จากการควบคุมกลอนไฟฟ้าผ่านโมดูลรีเลย์ โดยกำหนดว่าเมื่ออาร์เอฟไอดีทำงานจากการที่ป้ายแท็กอยู่ในระยะ แสดงการทำงานผ่านแผนภาพ Flowchart ดังรูปที่ 3.2



รูปที่ 3.2 แผนผัง Flowchart ของการควบคุมกลอนไฟฟ้าผ่านอาร์เอฟไอดี

ส่วนการเชื่อมต่อใช้งานกับ NodeMCU นั้นจะสามารถทำได้ดังรูปที่ 3.3 เริ่มจากการนำพินที่ D7 และD8 ที่ใช้สำหรับอินพุต และเอาท์พุตแบบดิจิตอล เชื่อมกับ Tx และRx ของโมดูล RDM6300 ตามเส้นสีเหลืองกับเขียว ต่อแรงดัน 5 โวลต์, GND และสุดท้ายเชื่อมสายอากาศเข้าที่เส้นสีม่วง ซึ่งทำหน้าที่เป็นตัวอ่านป้ายแท็กอาร์เอฟไอดี



รูปที่ 3.3 การเชื่อมต่อ RDM6300 กับNodeMCU

3.1.1.3 การควบคุมการทำงานของกลอนไฟฟ้า กับHall Effect

ในส่วนของการเพิ่มการควบคุมการทำงานของกลอนไฟฟ้า กับHall Effect จะเป็นการทำต่อเนื่องมาจากการออกแบบการควบคุมการทำงานของกลอนไฟฟ้าผ่านอาร์เอฟไอดี และการควบคุมกลอนไฟฟ้าผ่านโมดูลรีเลย์ อย่างที่ทราบว่า Hall Effect จะเป็นการทำงานโดยสนามแม่เหล็กและเปลี่ยนสถานะตามความหนาแน่นของฟลักซ์แม่เหล็กที่มากระทำ หากความเข้มสนามแม่เหล็กมากก็จะทำให้เกิดแรงดันมาก จากนั้นเมื่อนำแม่เหล็กเข้าใกล้พื้นที่ที่ทำงานของฮอลล์เอฟเฟ็คเซนเซอร์ (Active Area) เพื่อสร้างสนามแม่เหล็ก จะทำให้ความต่างศักย์เกิดการเปลี่ยนแปลง โดยจะเพิ่มขึ้นเมื่อนำขั้วใต้ (S) เข้าใกล้ หรือลดลงเมื่อนำขั้วเหนือ (N) เข้าใกล้

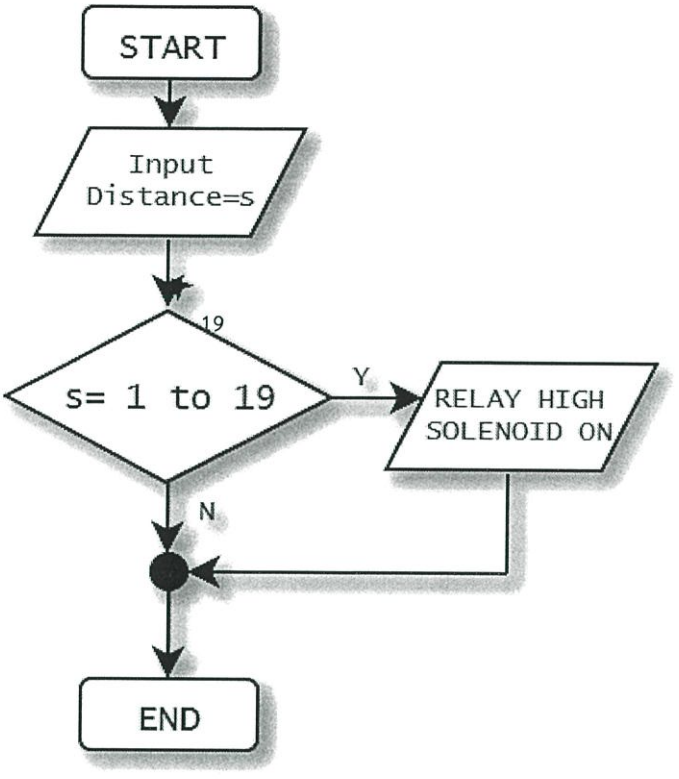
ดังนั้นเราจะนำ Hall Effect มาแสดงผลว่าประตูมีการเปิดจริง โดยติดแม่เหล็กไว้ที่บานพับของประตู จากการดันเข้าของสัตว์เลี้ยงเมื่ออาร์เอฟไอดีทำงานแล้วประตูมีการเปิด แสดงว่าสถานะของฮอลล์ได้มีการเปลี่ยนแปลง เนื่องจากแม่เหล็กจะออกห่างไปจากตัวรับรู้ฮอลล์ และเมื่อประตูปิดกลับแสดงว่าสัตว์เลี้ยงได้ผ่านเข้าไปในประตูแล้ว จึงทำการปิดกลอนไฟฟ้าได้

การเชื่อมต่อ Hall Effect Sensor เข้ากับ NodeMCU เราจะใช้ทรานซิสเตอร์ US5881 หรือ US1881 เชื่อมต่อกับตัวต้านทาน 10k จากนั้นจ่ายแรงดัน 5 โวลต์ และกราวน์ของ NodeMCU เข้ากับขาของทรานซิสเตอร์ให้เรียบร้อย แล้วจึงนำขาดิจิทัลเอาท์พุตของทรานซิสเตอร์ เข้าพิน D0 ของ NodeMCU ที่ใช้สำหรับอินพุต และเอาท์พุตแบบดิจิตอล

3.1.1.4 การควบคุมการทำงานของกลอนไฟฟ้ากับอัลตราโซนิคเซนเซอร์

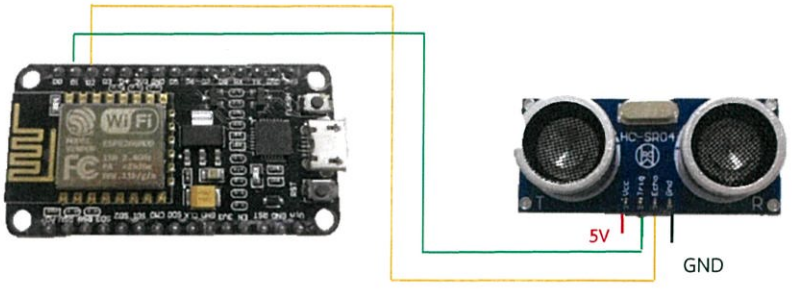
การทดลองปฏิบัติเกี่ยวกับอัลตราโซนิคเซนเซอร์ ซึ่งเป็นการวัดระยะทางโดยไม่ต้องมีการสัมผัสกับตำแหน่งที่ต้องการวัด ส่งสัญญาณอัลตร้าโซนิคความถี่ 40 กิโลเฮิรตซ์ ไปที่วัตถุที่

ต้องการวัด และรับสัญญาณที่สะท้อนกลับมา ซึ่งจะถูกนำมาใช้ในประตูฝั่งขาออก โดยถ้าสัตว์เลี้ยงเข้ามาใกล้ระยะที่กำหนด ประมาณ 1 ถึง 19 เซนติเมตร จะสั่งให้กลอนไฟฟ้าอีกด้านเปิดออก เพื่อที่จะออกมาข้างนอกได้ ซึ่งการทำงานก็จะทำต่อมาจากการเชื่อมต่อจากการควบคุมกลอนไฟฟ้าผ่านโมดูลรีเลย์ที่ควบคุมกลอนไฟฟ้าอยู่ แสดงผ่านแผนภาพ Flowchart ได้ดังรูปที่ 3.5



รูปที่ 3.4 แผนผัง Flowchart ของการควบคุมกลอนไฟฟ้าอัลตราโซนิคเซนเซอร์

การเชื่อมต่ออัลตราโซนิคเซนเซอร์ กับNodeMCU เริ่มจากต่อแรงดัน 5 โวลต์ และกราวน์เข้ากับขาของอัลตราโซนิคให้เรียบร้อย และจากนั้นนำขาส่งสัญญาณ (Trig) และรับสัญญาณ (Echo) ต่อเข้าพิน D1 และD2 ของ NodeMCU ที่ใช้สำหรับอินพุตเอาท์พุตแบบดิจิทัล ตามลำดับแสดงดังรูปที่ 3.5



รูปที่ 3.5 การเชื่อมต่ออัลตราโซนิคเซนเซอร์ กับNodeMCU

3.1.2 วิเคราะห์การเข้าออกของสัตว์เลี้ยง

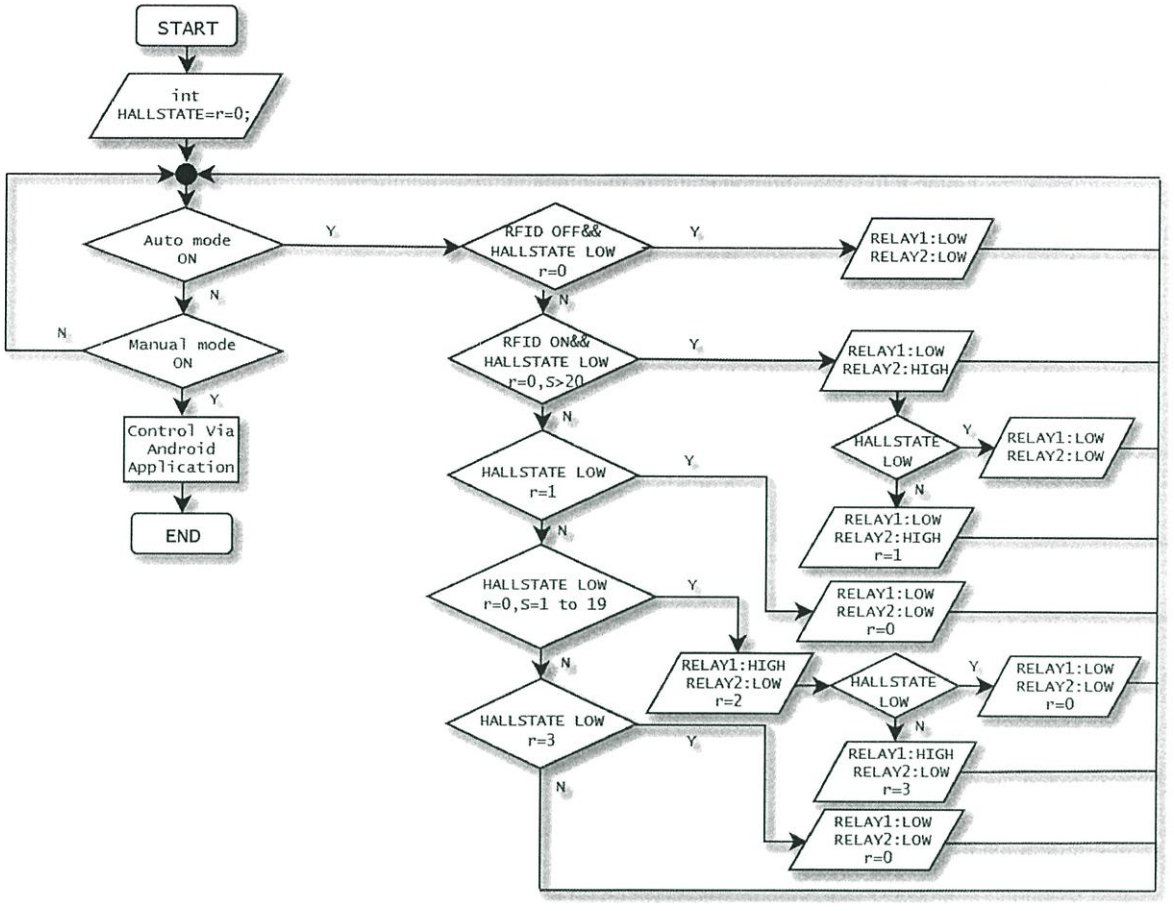
การออกแบบเปิด-ปิดประตูของสัตว์เลี้ยง ต้องคำนึงถึงการเข้าออกของสัตว์เลี้ยงเป็นหลักในการทำงาน เพราะถ้ามีการวิเคราะห์ที่ผิดพลาดอาจจะเป็นอันตรายต่อสัตว์ได้ จึงได้ออกแบบการเข้า-ออกของสัตว์เลี้ยงด้วยกันทั้งหมด 4 กรณี

1. สัตว์เลี้ยงไม่เข้าประตู จะเกิดเหตุการณ์ด้วยกันสองเหตุการณ์ หนึ่งคือ อาร์เอฟไอดีไม่ทำงานประตูไม่มีการเปิด-ปิด และอีกเหตุการณ์หนึ่งคือ อาร์เอฟไอดีทำงานแต่ตัวรับรู้ฮอลล์ไม่เปลี่ยนสถานะ แสดงว่าประตูไม่มีการเปิด จึงออกคำสั่งให้ประตูปิดทั้งสองเหตุการณ์

2. สัตว์เลี้ยงเข้าประตู อาร์เอฟไอดีจะทำงาน ตัวรับรู้ฮอลล์เปลี่ยนสถานะจากการที่ประตูถูกดันเข้า ประตูก็จะถูกเปิดและถูกปิดอีกครั้งเมื่อสถานะฮอลล์กลับมาที่จุดเดิม

3. สัตว์เลี้ยงไม่ออกจากประตู จะเกิดเหตุการณ์ด้วยกันสองเหตุการณ์ หนึ่งคือ อัลตราโซนิคไม่ทำงานประตูไม่มีการเปิด-ปิด อีกเหตุการณ์หนึ่งคืออัลตราโซนิคเซนเซอร์ส่งค่ามาว่าอ่านค่าได้ 1-19 กลอนไฟฟ้าจึงจะทำงาน แต่จะถูกปิดลงเมื่อตรวจสอบแล้วว่าตัวรับรู้ฮอลล์ไม่มีการเปลี่ยนสถานะ

4. สัตว์เลี้ยงออกจากประตู คืออัลตราโซนิคเซนเซอร์ส่งค่ามาว่าอ่านค่าได้ 1-19 กลอนไฟฟ้าจะทำงาน และรับรู้ได้ถึงค่าตัวรับรู้ฮอลล์ที่เปลี่ยนสถานะเมื่อประตูถูกเปิดออก และจะปิดอีกครั้งเมื่อสถานะฮอลล์กลับมาที่จุดเดิม

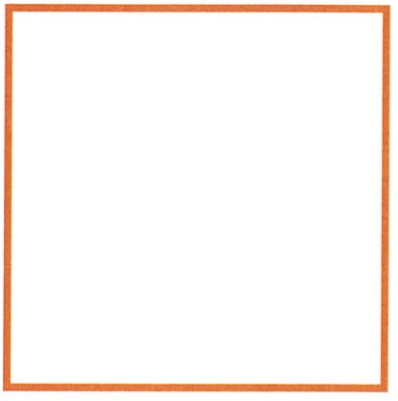


รูปที่ 3.6 แผนผัง Flowchart ของทั้งระบบ

จากรูปที่ 3.6 เป็นแผนผัง Flowchart ของทั้งระบบ ซึ่งจะแสดงการตัดสินใจในการทำงานของกลอนไฟฟ้า โดยให้รีเลย์ตัวที่ 1 เป็นตัวควบคุมกลอนไฟฟ้าตัวที่ 1 ด้านนอกประตูขาเข้า และรีเลย์ตัวที่ 2 เป็นตัวควบคุมกลอนไฟฟ้าตัวที่ 2 ด้านในประตูขาออก เมื่อรีเลย์มีสถานะ LOW กลอนไฟฟ้าจะไม่ทำงานประตูไม่สามารถเปิดได้ เมื่อรีเลย์มีสถานะ HIGH กลอนไฟฟ้าจะทำงานประตูสามารถเปิดได้

3.1.3 การออกแบบพารามิเตอร์สายอากาศ

เริ่มด้วยการออกแบบขนาดสายอากาศที่ต้องการ ในที่นี้จะออกแบบขนาดที่เหมาะสม คือ 25 เซนติเมตร x 25 เซนติเมตร ดังรูปที่ 3.7

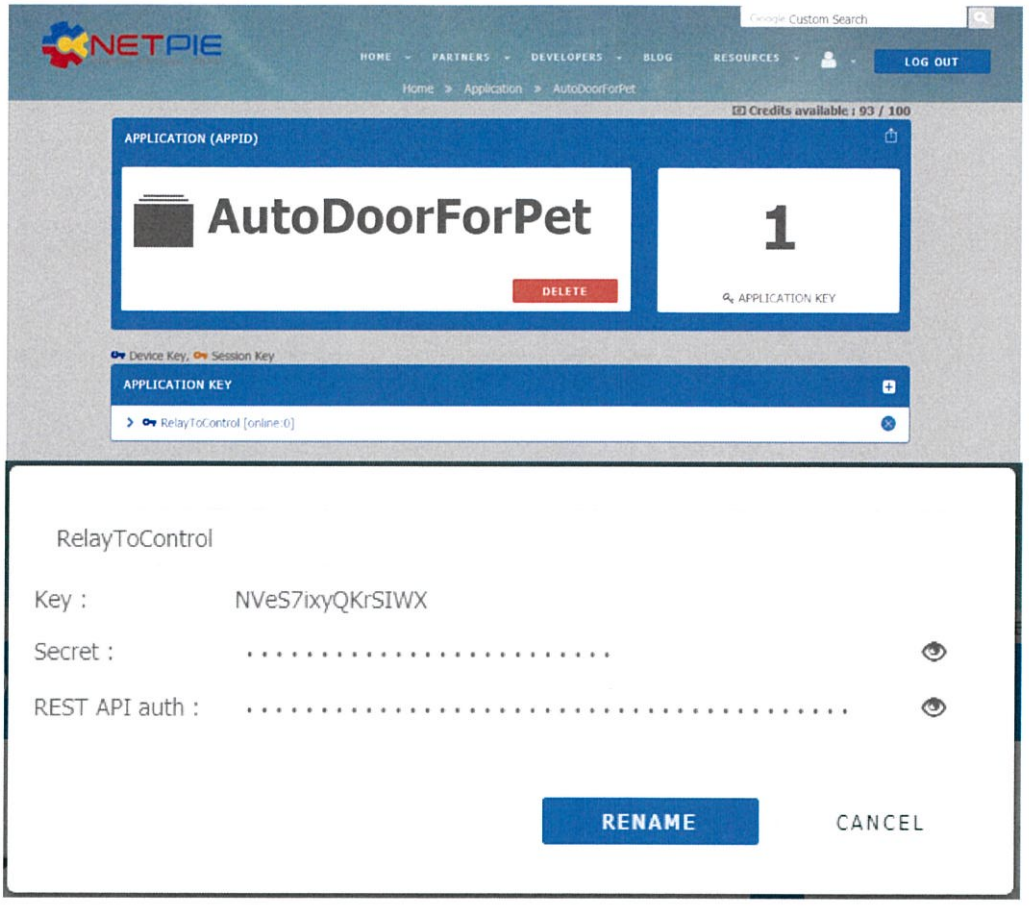


รูปที่ 3.7 ขนาดสายอากาศที่ต้องการ

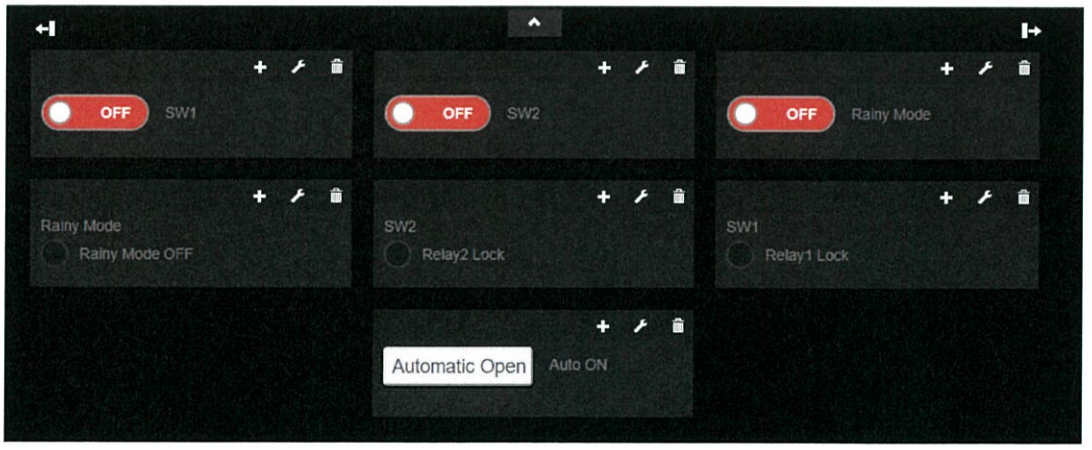
การคำนวณพารามิเตอร์ต่างๆของสายอากาศนั้นจะเป็นไปตามทฤษฎีเรโซแนนซ์ เพื่อทำการ Impededance Matching ค่าความเหนี่ยวนำ และตัวเก็บประจุ ให้สามารถอ่านค่ากันได้ แล้วจึงนำไปเชื่อมต่อกับ RFID Reader Module เพื่ออ่านค่า RFID Tag ความถี่ 125 kHz ต่อไป

3.1.4 การออกแบบ NETPIE SERVER

การออกแบบให้ระบบ สามารถเชื่อมต่อกับ Android Application ผ่านอินเตอร์เน็ตไวไฟได้ ขั้นตอนแรกให้สมัครสมาชิก NETPIE โดยสร้าง Application ใหม่บนเว็บ netpie.io เพื่อให้ได้ APPID, KEY และ SECRET ภายใต้ Application ID เดียวกัน ดังรูปที่ 3.8 และเริ่มต้นการออกแบบ Freeboard ใน NETPIE เพื่อใช้ควบคุมรีเลย์ โดยจะออกแบบให้สามารถควบคุมกลอนประตูโดยตรงผ่าน NETPIE ซึ่งจะมี Rainy Mode ไว้ปิดกลอนประตูทุกกรณี ไม่ต้องการให้สัตว์เลี้ยงออก หรือ Auto Open ผ่าน Ultrasonic และRFID ดังรูปที่ 3.9



รูปที่ 3.8 หน้าแรกของ Application ใน NETPIE



รูปที่ 3.9 สร้างหน้า Freeboards สำหรับไปควบคุมรีเลย์

ทำการออกแบบแอปพลิเคชั่นแอนดรอย์เพื่อเชื่อมต่อกับ NETPIE ผ่าน Microgear-android ซึ่งก็คือ client library สำหรับ Android Studio ที่ทำหน้าที่เป็นตัวกลางในการเชื่อมโยง application code เข้ากับบริการของ NETPIE platform เพื่อการพัฒนา IOT application

รายละเอียดเกี่ยวกับ NETPIE Platform MicroGear เป็น Library ที่วิ่งบน MQTT Protocol ซึ่งติดต่อกับ NETPIE Broker จะใช้ OAuth เข้ามาช่วยในการในการสร้างข้อมูลสำคัญที่ใช้เชื่อมต่อกับ MQTT Broker ดังนี้ MQTT Username, MQTT Password และMQTT ClientId แล้วจึงทดสอบการติดต่อ กับNETPIE โดยใช้ ID, KEY และSECRET จาก NETPIE ที่ทำการสร้าง Freebroadไว้ ดังรูปที่ 3.10

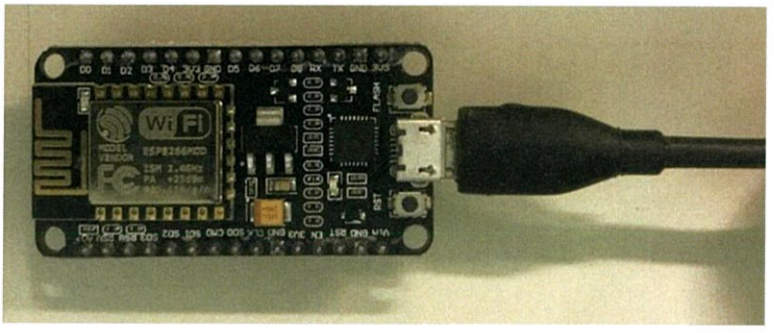
```
public class MainActivity extends AppCompatActivity {

    private Microgear microgear = new Microgear(this);
    private String appid = "AutoDoorForPet" ; //APP_ID
    private String key = "NVeS7ixyQKrSIWX" ; //KEY
    private String secret = "3KvF6nvfnFhvecPNVmBaVgjKb" ; //SECRET
    private String alias = "android";
```

รูปที่ 3.10 ID, KEY และSECRET จาก Netpies

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 NodeMCU/ESP8266

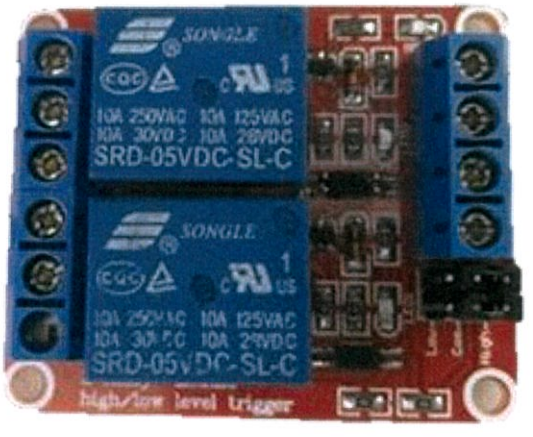
NodeMCU/ESP8266 ใช้ในการรับ-ส่งข้อมูล เชื่อมต่อไวไฟ และอุปกรณ์เข้าด้วยกัน เพื่อนำไปประมวลผล พร้อมสายยูเอสบีในการเชื่อมต่อกับคอมพิวเตอร์ ภาพอุปกรณ์แสดงดังรูปที่ 3.11



รูปที่ 3.11 NodeMCU/ESP8266

3.2.2 2-Channel Relay Module

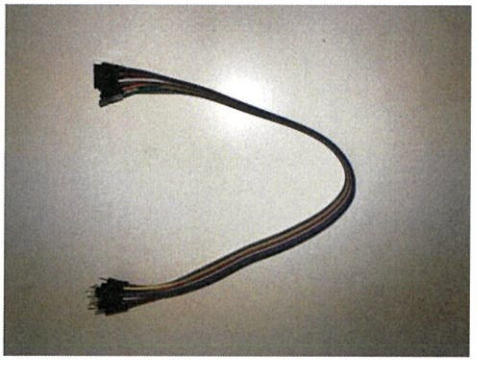
โมดูลรีเลย์ 2 ตัว ใช้งานในการควบคุมกลอนประตูไฟฟ้า รับแรงดัน 5 โวลต์ต่อตรงเข้ากับอาดุยโน่ ภาพอุปกรณ์แสดงดังรูปที่ 3.12



รูปที่ 3.12 2-Channel Relay Module

3.2.3 สาย jumper

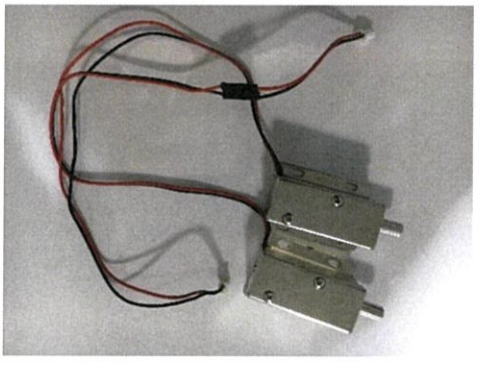
สาย jumper ใช้เชื่อมต่ออุปกรณ์แต่ละตัวเข้าด้วยกัน มีหลากหลายสี เพื่อง่ายต่อการจำ และทำให้ไม่สับสนในการเชื่อมต่อ ภาพอุปกรณ์แสดงดังรูปที่ 3.13



รูปที่ 3.13 สาย jumper

3.2.4 กลอนไฟฟ้า

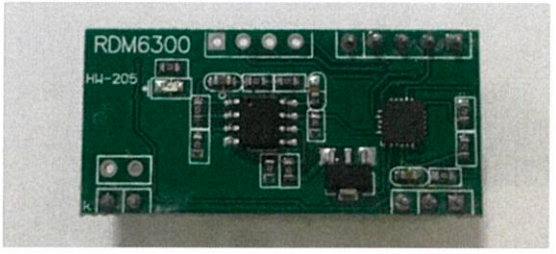
กลอนไฟฟ้าสามารถควบคุมผ่านรีเลย์ และสั่งผ่านไมโครคอนโทรลเลอร์ เช่น อาดุยโน่ ทำหน้าที่เป็นกลอนเปิด-ปิดประตู ภาพอุปกรณ์แสดงดังรูปที่ 3.14



รูปที่ 3.14 กลอนไฟฟ้า

3.2.5 Arduino 125kHz RFID Reader Module (RDM6300)

โมดูล RDM6300 125kHz ใช้ในการอ่านป้ายแท็กอาร์เอฟไอดีความถี่ 125 kHz ภาพอุปกรณ์แสดงดังรูปที่ 3.15



รูปที่ 3.15 RFID Reader Module

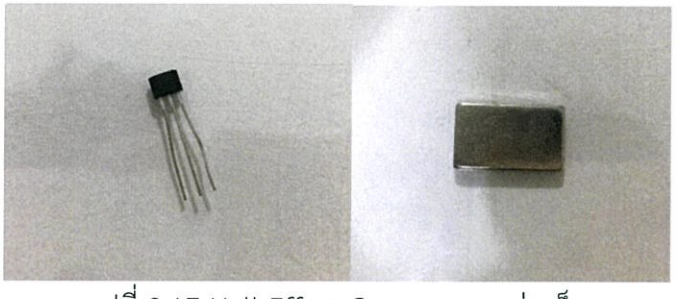
3.2.6 RFID 125 kHz Proximity Token Key Tags

ป้ายแท็กอาร์เอฟไอดี ความถี่ 125KHz มาตรฐานการ์ด TK4100 เข้ารหัสแบบ Manchester 40 bits ทำมาจากพีวีซีขนาด 85.6 × 54 × 0.86 (mm) ภาพอุปกรณ์แสดงดังรูปที่ 3.16

รูปที่ 3.16 RFID Key Tag

3.2.7 Hall Effect Sensor และแม่เหล็ก

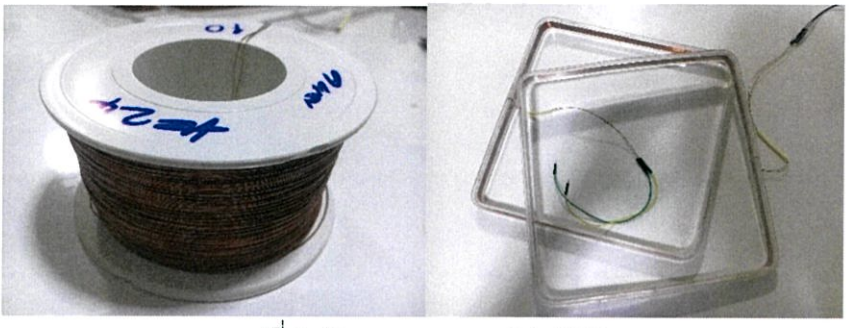
เราจะใช้ทรานซิสเตอร์ US5881 หรือUS1881 กับแม่เหล็ก ในการออกแบบควบคุมการทำงานของกลอนไฟฟ้ากับ Hall Effect ภาพอุปกรณ์แสดงดังรูปที่ 3.17



รูปที่ 3.17 Hall Effect Sensor และแม่เหล็ก

3.2.8 ลวดทองแดง 24 AWG

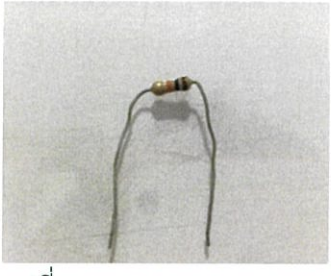
นำลวดทองแดง 24 AWG มาใช้ในการออกแบบสายอากาศ ให้เครื่องอ่านโมดูล RDM6300 125kHz ภาพอุปกรณ์แสดงดังรูปที่ 3.18



รูปที่ 3.18 ลวดทองแดง 24 AWG

3.2.9 R 10K 1/4W 5%

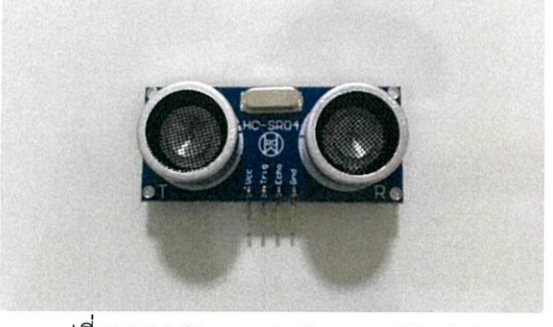
ใช้ในการเชื่อมต่อทรานซิสเตอร์ US5881 หรือ US1881กับตัวต้านทาน 10k เพื่อสร้างตัวรับรู้ฮอลล์ ภาพอุปกรณ์แสดงดังรูปที่ 3.19



รูปที่ 3.19 R 10K 1/4W 5%

3.2.10 Ultrasonic Sensor Module

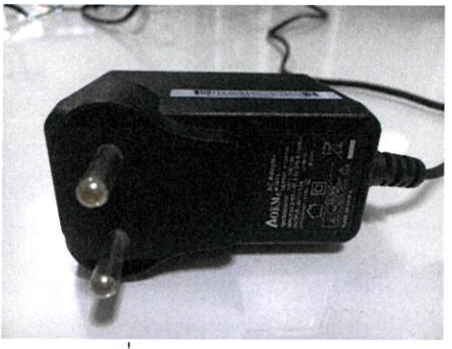
อัลตราโซนิคเซนเซอร์ ใช้ในการวัดระยะทางเมื่อมีวัตถุเข้ามาใกล้ โดยมีภาคส่งและรับสัญญาณในการวัดเพื่อเก็บข้อมูล ภาพอุปกรณ์แสดงดังรูปที่ 3.20



รูปที่ 3.20 Ultrasonic Sensor Module

3.2.11 Adapter 12V

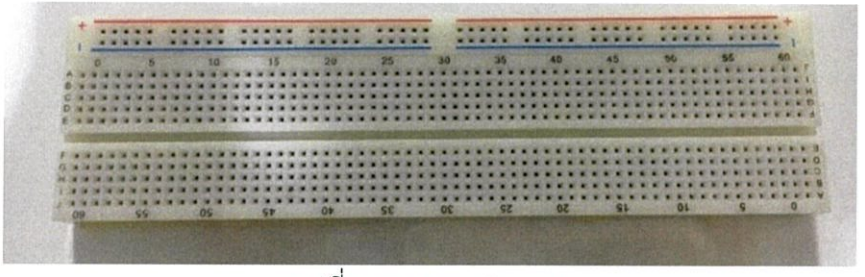
ทำหน้าที่จ่ายแรงดันให้กับกลอนไฟฟ้า ภาพอุปกรณ์แสดงดังรูปที่ 3.21



รูปที่ 3.21 Adapter 12V

3.2.12 Breadboard

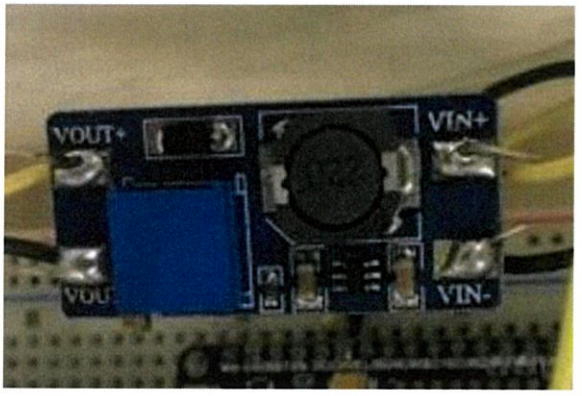
ใช้เชื่อมอุปกรณ์ อาทิเช่น กราวน์ และแรงดัน 5 โวลต์ ภาพอุปกรณ์แสดงดังรูปที่ 3.22



รูปที่ 3.22 Breadboard

3.2.13 MT3608 2A DC-DC Step Up Power Module

โมดูลแปลงแรงดันต่ำเป็นแรงดันสูง รองรับแรงดันอินพุตได้ตั้งแต่ 2V ถึง 24V แรงดันเอาต์พุตสามารถปรับได้ตั้งแต่ 5V ถึง 28V จ่ายกระแสได้สูงสุดถึง 2A ภาพอุปกรณ์แสดงดังรูปที่ 3.23



รูปที่ 3.23 MT3608 2A DC-DC Step Up Power Module

3.3 การจัดเก็บผลการทดลอง

การจัดเก็บผลการทดลองเพื่อสังเกตุการทำงานของอุปกรณ์ในการรับ-ส่งข้อมูล จะแบ่งออกเป็น 5 ส่วน

3.3.1 การจัดเก็บผลทดสอบระยะของอัลตราโซนิคเซนเซอร์

การจัดเก็บผลทดสอบระยะของอัลตราโซนิคเซนเซอร์จะเป็นการเก็บข้อมูลที่ระยะทางต่างๆ สังเกตุข้อมูลที่ได้รับมาจากการส่งของอัลตราโซนิคเซนเซอร์ ว่าส่งค่าอะไรมาที่ระยะทางต่างกัน

3.3.2 ผลการทดลองการอ่านข้อมูลใน RFID Tag จาก RFID Reader Module

ทำการทดสอบการทำงานของระบบเปิด-ปิดประตู ว่าเป็นตามที่ต้องการหรือไม่ เมื่อการทำงานของอุปกรณ์เกิดเหตุการณ์ในรูปแบบต่างๆกัน

3.3.3 การคำนวณพารามิเตอร์สายอากาศ

ทำการวัดและคำนวณผลที่ต้องการ สำหรับการออกแบบสายอากาศ เพื่อนำไปเชื่อมต่อกับRFID Reader จึงต้องออกแบบให้เหมาะสมกัน เพื่อให้อ่านค่ากันได้

3.3.4 การเก็บค่าจากการสั่งงานผ่าน NETPIE

เมื่อทำการออกแบบระบบประตูเปิดแล้ว จึงนำมาพัฒนาให้สามารถสั่งงานจาก NETPIE Freeboard ผ่านอินเตอร์เน็ตได้ และมีปุ่มสั่งงานตามที่ได้ทำการออกแบบไว้

3.3.5 ผลการทดลองภาพรวมของระบบทั้งหมด

หลังจากทำการทดลองต่างๆ ในขั้นตอนข้างต้นจนเสร็จสิ้นแล้ว จากนั้นทำการประกอบส่วนทั้งหมดเข้าด้วยกัน และทดลองภาพรวมในการทำงานของระบบทั้งหมด

3.3.6 ผลการทดลองผ่านหน้าแอปพลิเคชั่นแอนดรอยด์

ทำการทดลองการใช้แอปพลิเคชั่นที่ได้ทำการสร้างโดยโปรแกรม Android Studio ในทุกๆ คำสั่งที่ควบคุมประตู โดยผ่าน NETPIE

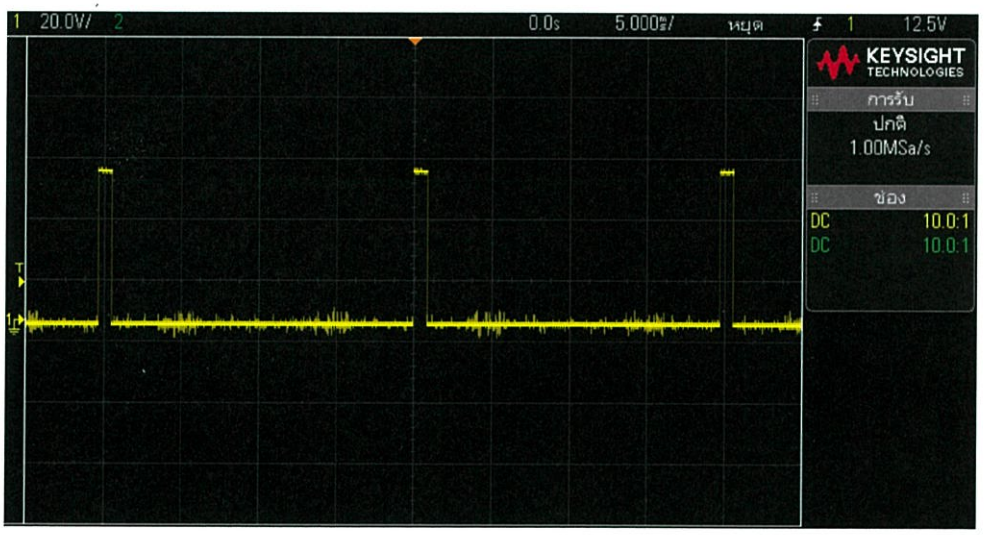
บทที่ 4

ผลการทดลอง

ในบทนี้จะกล่าวถึงผลการทดลองการทดสอบการอ่านค่าที่ได้จาก Ultrasonic Sensor Module, RFID Reader Module และการทำงานภาพรวมทั้งหมดของระบบ โดยแสดงผลการทดลองการทำงานของระบบในส่วนต่างๆ ดังนี้

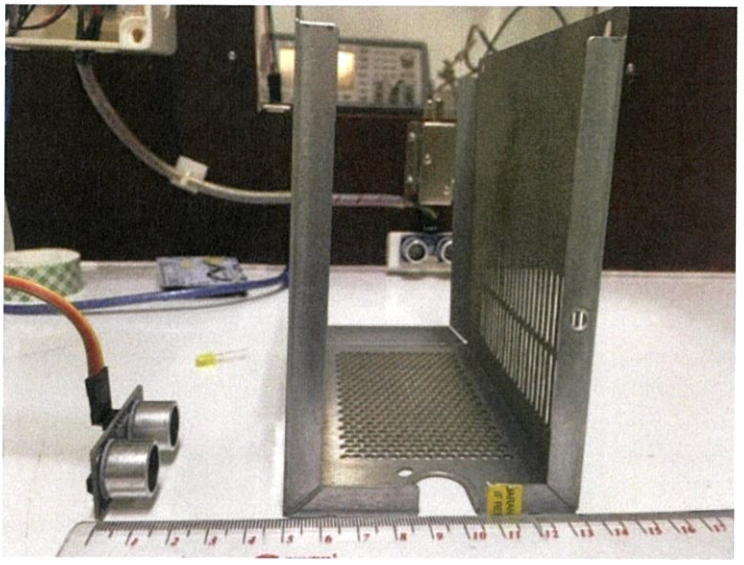
4.1 ผลการทดลองการอ่านค่าของ Ultrasonic Sensor Module

Ultrasonic Sensor Module เป็นอุปกรณ์ที่จะใช้ในระบบเปิดปิดประตูอัตโนมัติด้านใน ขาออกของสัตว์เลี้ยง โดยระยะทางที่วัดออกมาได้จะอยู่ในรูปของพัลส์ จึงเก็บผลการทดลองในรูปของคาบเวลา ผ่านออสซิโลสโคป แสดงดังรูปที่ 4.1



รูปที่ 4.1 พัลส์ที่รับมาจาก Ultrasonic Sensor Module

การทดลองการรับสัญญาณพัลส์คาบเวลาที่ส่งมาจาก Ultrasonic Sensor Module ในแต่ละระยะโดยทำการเพิ่มระยะขึ้นครั้งละ 5 เซนติเมตร ทำการทดลองทั้งหมด 3 ครั้ง แสดงการวัด ดังรูปที่ 4.2



รูปที่ 4.2 การวัดสัญญาณพัลส์คาบเวลา Ultrasonic Sensor Module

ตารางที่ 4.1 ความสัมพันธ์ระหว่างระยะทาง (cm) กับคาบเวลา (ms) ของ Ultrasonic Sensor Module จากการทดลองครั้งที่ 1

ระยะทาง (cm)	คาบเวลา (ms)
5	3.12
10	3.44
15	4.38
20	5.40
25	6.02
30	6.58
35	8.32
40	10.68
45	12.51
50	16.47
55	17.04
60	20.38

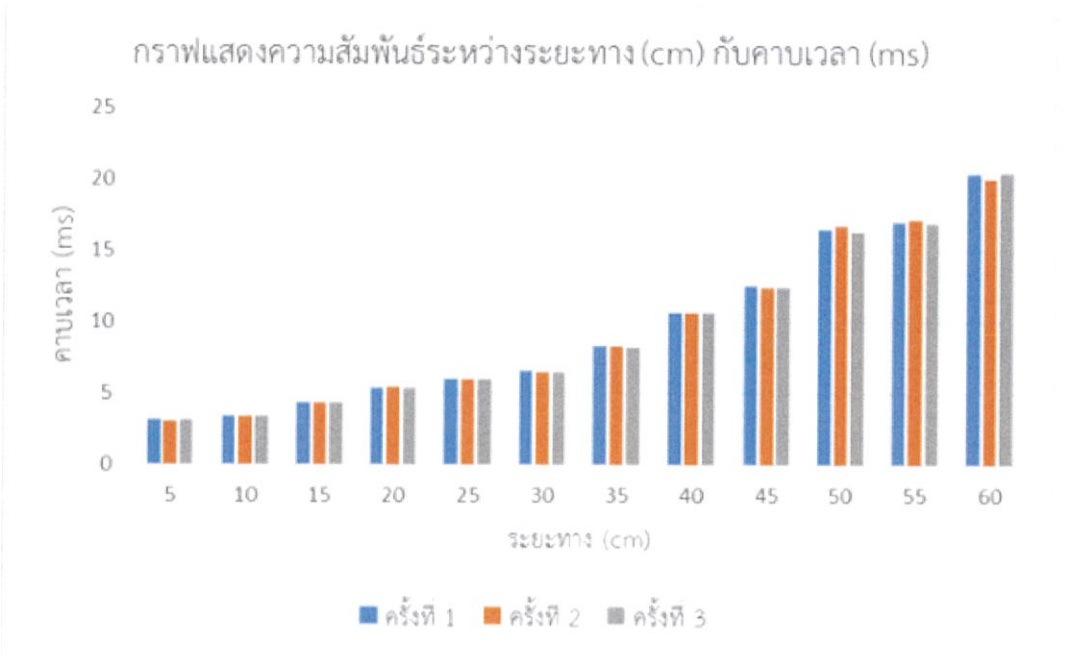
ตารางที่ 4.2 ความสัมพันธ์ระหว่างระยะทาง (cm) กับคาบเวลา (ms) ของ Ultrasonic Sensor Module จากการทดลองครั้งที่ 2

ระยะทาง (cm)	คาบเวลา (ms)
5	3.09
10	3.45
15	4.39
20	5.48
25	6.00
30	6.51
35	8.35
40	10.66
45	12.48
50	16.76
55	17.15
60	20.02

ตารางที่ 4.3 ความสัมพันธ์ระหว่างระยะทาง (cm) กับคาบเวลา (ms) ของ Ultrasonic Sensor Module จากการทดลองครั้งที่ 3

ระยะทาง (cm)	คาบเวลา (ms)
5	3.12
10	3.45
15	4.40
20	5.40
25	5.97
30	6.55
35	8.24
40	10.64
45	12.45
50	16.28
55	16.91
60	20.45

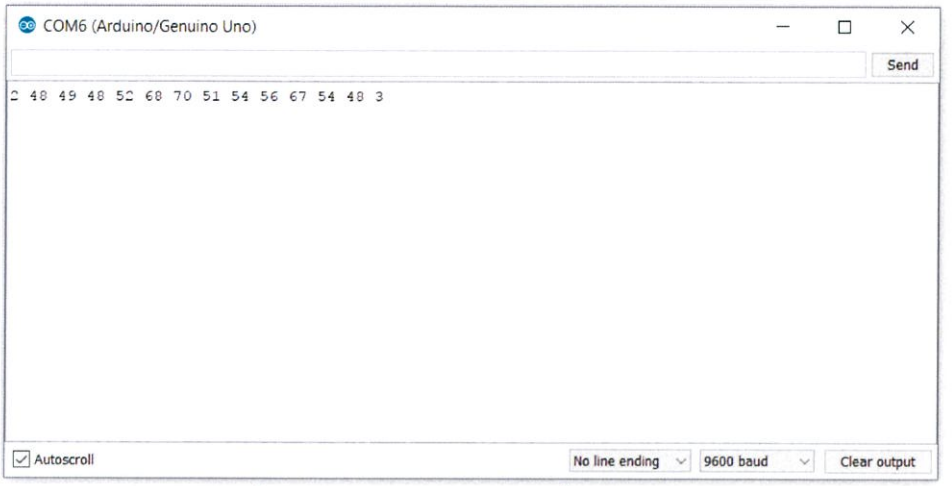
จากตารางที่ 4.1, 4.2 และ4.3 แสดงความสัมพันธ์ระหว่างระยะทาง (cm) กับคาบเวลา (ms) ของ Ultrasonic Sensor Module สามารถทำการวิเคราะห์ผลเป็นไปตามสมการเส้นตรง โดยที่คาบแปรผันตรงตามระยะทางดังแสดงในรูปที่ 4.3



รูปที่ 4.3 กราฟแสดงความสัมพันธ์ระหว่างระยะทาง (cm) กับคาบเวลา (ms) ของ Ultrasonic Sensor Module จากการทดลองทั้ง 3 ครั้ง

4.2 ผลการทดลองอ่านข้อมูลใน RFID Tag จาก RFID Reader Module

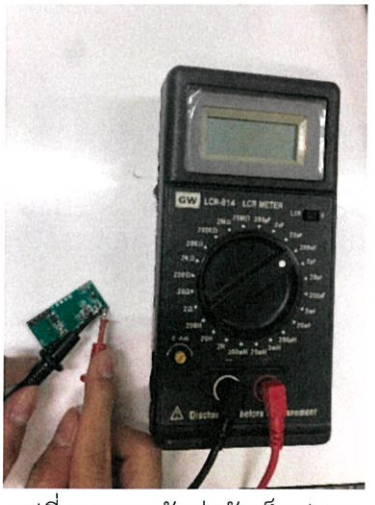
ในการอ่านข้อมูลจาก RFID Tag จาก RFID Reader Module เป็นการอ่านข้อมูลผ่านการเหนี่ยวนำจากคลื่นวิทยุด้วยการกรองคลื่นพาห์ออก วัดได้ดังรูปที่ 4.4 การแสดงค่าของ RFID Tag ที่ RFID Reader Module ส่งมาให้ไมโครคอนโทรลเลอร์ผ่าน Serial Monitor ของ Arduino IDE โดยหมายเลข RFID Tag ที่ได้ถูกกำหนดให้อยู่ในรูปเลขฐาน 10



รูปที่ 4.4 หมายเลข RFID Tag ที่ทำการทดสอบ

4.3 ผลการทดลองของสายอากาศ

เมื่อออกแบบขนาดสายอากาศที่ต้องการ คือ 25 เซนติเมตร x 25 เซนติเมตร แล้วใช้มิเตอร์ที่สามารถวัดค่าตัวเหนี่ยวนำ และตัวเก็บประจุได้ นำมาวัดที่ตัว RFID Reader เพื่อสังเกตุผลว่ามีผลเป็นค่าความเหนี่ยวนำ หรือค่าตัวเก็บประจุเพื่อใช้ในการคำนวณ ดังรูปที่ 4.5



รูปที่ 4.5 การวัดค่าตัวเก็บประจุ

นำทฤษฎีที่เกี่ยวข้องจากบทที่ 2 มาใช้ในการคำนวณพารามิเตอร์ต่างๆ ในที่นี้ใช้ขนาดเป็นสี่เหลี่ยมจตุรัสขนาด 25 เซนติเมตร x 25 เซนติเมตร และลวดทองแดง 24 AWG ดูค่าได้ดังตารางที่ 4.4 และ ค่าตัวเก็บประจุที่วัดได้จากตัวอ่านอาร์เอฟไอดีเป็น 7.72 nF จึงคำนวณค่าตัวเหนี่ยวนำออกมาได้เป็น 210 μH และเนื่องจากเรามี RFID Reader Module ที่ใช้ค่าความถี่ 125 kHz ส่วนค่า a วัดได้ 12.5 เซนติเมตร ค่า b วัดได้ 1 เซนติเมตร แสดงการคำนวณดังต่อไปนี้

จากการแมตช์อิมพิแดนซ์ ของวงจรเรโซแนนซ์ จะได้

$$X_L = X_C$$

$$2\pi f_0 L = \frac{1}{2\pi f_0 C}$$

$$2\pi(125\times10^3)L = \frac{1}{2\pi(125\times10^3)(7.72\times10^{-9})}$$

$$L = 210\ \mu H$$

เมื่อค่า a = 12.5 cm, b = 1 cm และค่า c = $21.6\times2.54\times10^{-3}\times2$ = 0.11 cm

$$L = 0.008aN^2\left\{2.303\log_{10}\left(\frac{a}{b+c}\right)+0.2235\left(\frac{b+c}{a}\right)+0.726\right\}\ (\mu H)$$

$$210 = 0.008(12.5)N^2\left\{2.303\log_{10}\left(\frac{12.5}{1+0.055}\right)+0.2235\left(\frac{1+0.055}{12.5}\right)+0.726\right\}$$

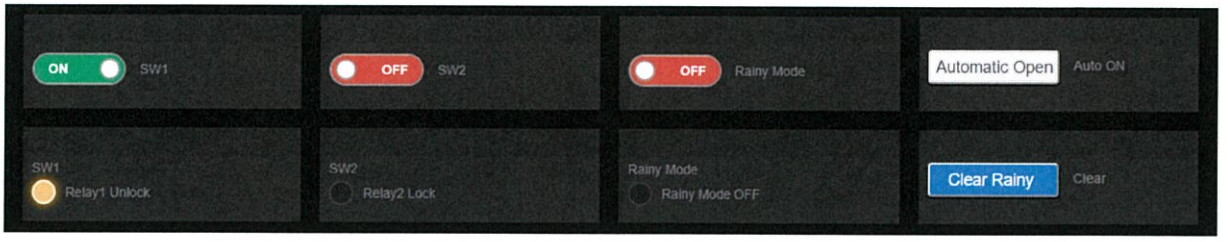
N = 25.7 ประมาณ 26 รอบ

ตารางที่ 4.4 AWG Wire Chart

Wire Size (AWG)	Dia. in Mils (bare)	Dia. in Mils (coated)	Ohms/ 1000 ft.
20	32.0	34.0	10.1
21	28.5	30.2	12.8
22	25.3	28.0	16.2
23	22.6	24.2	20.3
24	20.1	21.6	25.7
25	17.9	19.3	32.4
Note : mil = 2.54×10^{-3}cm			

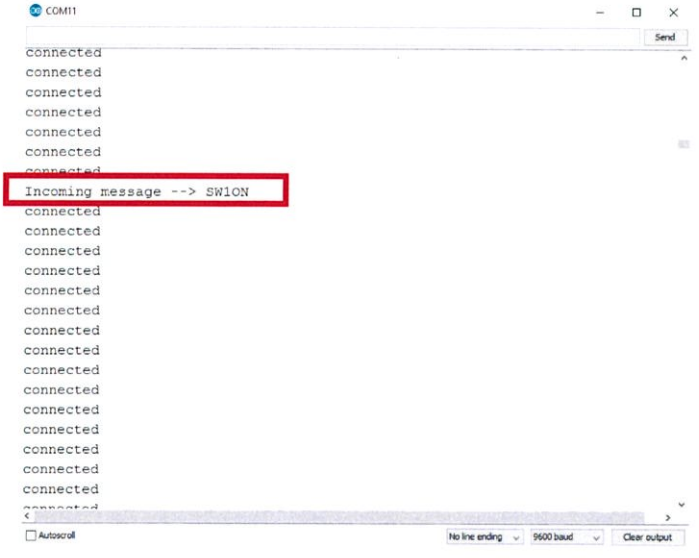
4.4 การเก็บค่าจากการสั่งงานผ่าน NETPIE

เก็บผลการทดลองจากการกดปุ่มสั่งงานบน NETPIE ตามที่กำหนดไว้ สังเกตุผลการทดลองผ่าน Serial Monitor ที่ส่งค่าจาก NodeMCU ดังรูปต่อไปนี้ เพื่อแสดงว่ามีการเชื่อมต่อกับ NETPIE และเปลี่ยนแปลงสถานะการทำงานของกลอนไฟฟ้า

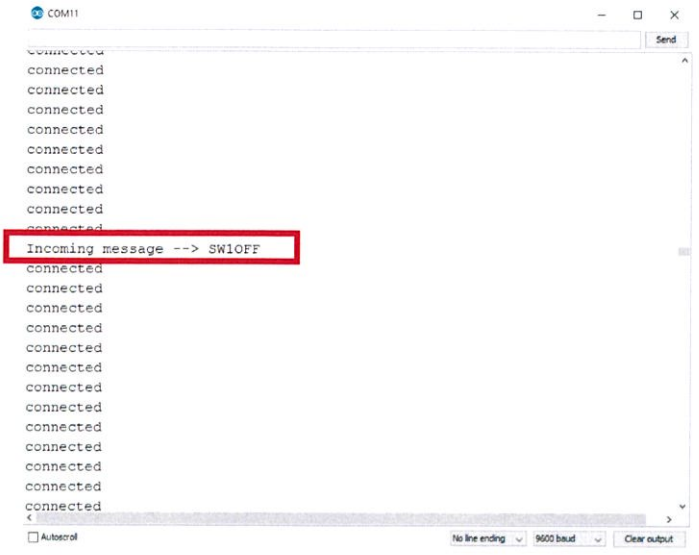


รูปที่ 4.6 NETPIE ขณะกดเปิดสวิตช์ปุ่มที่ 1

จากรูปที่ 4.6 แสดงหน้า NETPIE Freeboard ซึ่งผู้ใช้สามารถควบคุมได้ผ่านอินเตอร์เน็ตเว็บบราวเซอร์ ในภาพเป็นการควบคุมสวิตช์ปุ่มที่ 1 ให้มีสถานะ ON

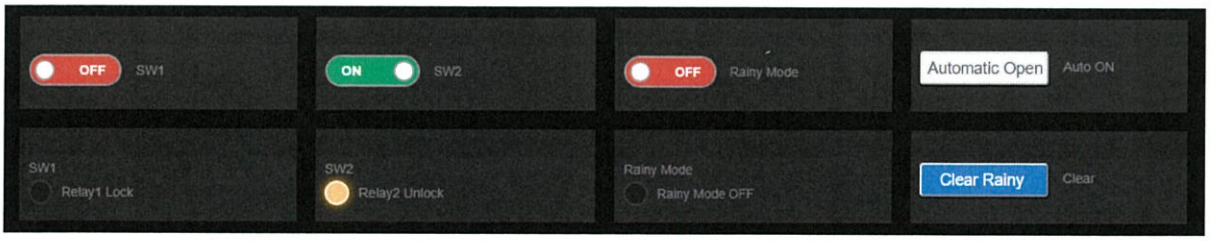


รูปที่ 4.7 Serial Monitor ขณะกดเปิดสวิตช์ปุ่มที่ 1



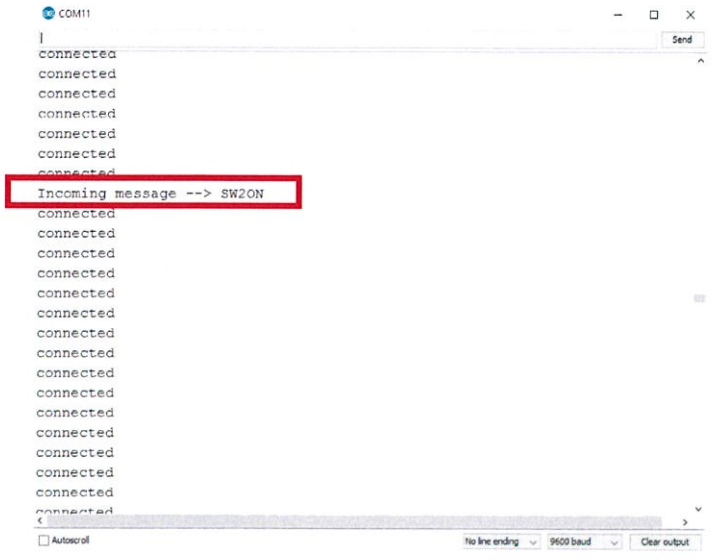
รูปที่ 4.8 Serial Monitor ขณะกดปิดสวิตช์ปุ่มที่ 1

จากรูปที่ 4.7 เป็นการแสดงข้อความขณะทำการควบคุมสวิตช์ปุ่มที่ 1 ให้มีสถานะ ON ที่ถูกส่งเข้ามาจาก NETPIE Freeboard แสดงผ่าน Serial Monitor บนโปรแกรม Arduino IDE และจากรูปที่ 4.8 เป็นการแสดงข้อความขณะทำการควบคุมสวิตช์ปุ่มที่ 1 ให้มีสถานะ OFF ที่ถูกส่งเข้ามาจาก NETPIE Freeboard แสดงผ่าน Serial Monitor บนโปรแกรม Arduino IDE

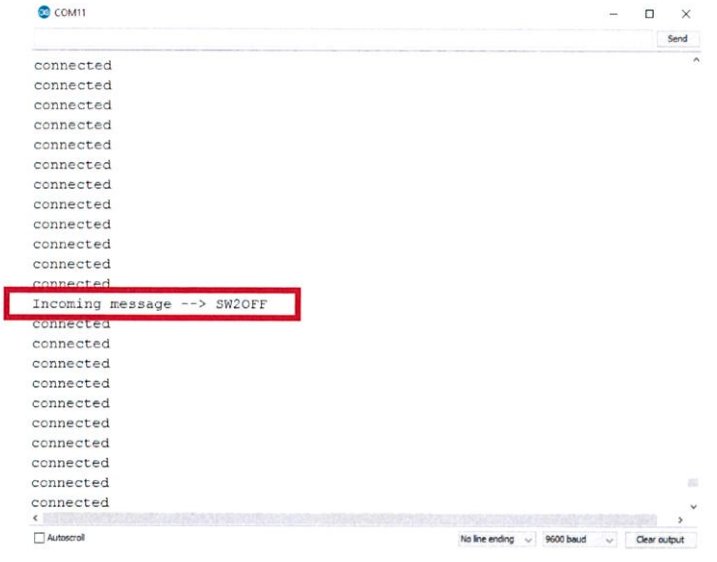


รูปที่ 4.9 NETPIE ขณะกดเปิดสวิตช์ปุ่มที่ 2

จากรูปที่ 4.9 แสดงหน้า NETPIE Freeboard ซึ่งผู้ใช้สามารถควบคุมได้ผ่านอินเตอร์เน็ตเว็บบราวเซอร์ ในภาพเป็นการควบคุมสวิตช์ปุ่มที่ 2 ให้มีสถานะ ON

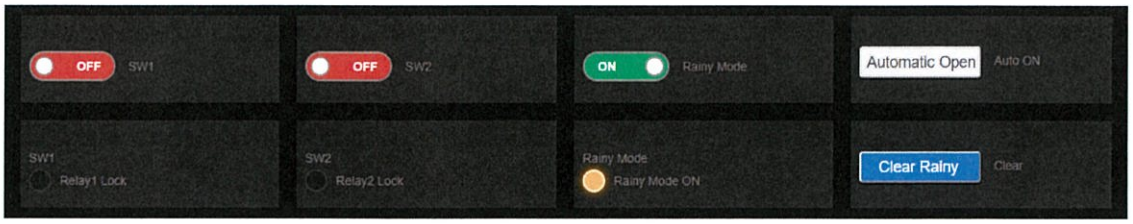


รูปที่ 4.10 Serial Monitor ขณะกดเปิดสวิตช์ปุ่มที่ 2



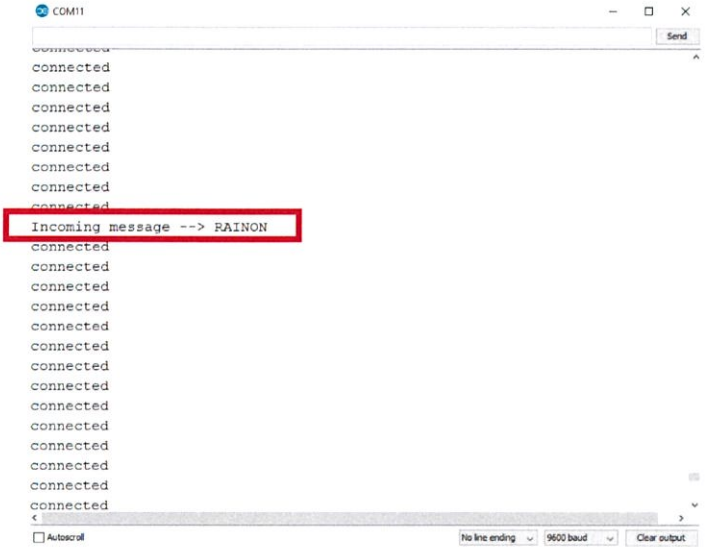
รูปที่ 4.11 Serial Monitor ขณะกดปิดสวิตช์ปุ่มที่ 2

จากรูปที่ 4.10 เป็นการแสดงข้อความขณะทำการควบคุมสวิตช์ปุ่มที่ 2 ให้มีสถานะ ON ที่ถูกส่งเข้ามาจาก NETPIE Freeboard แสดงผ่าน Serial Monitor บนโปรแกรม Arduino IDE และจากรูปที่ 4.11 เป็นการแสดงข้อความขณะทำการควบคุมสวิตช์ปุ่มที่ 2 ให้มีสถานะ OFF ที่ถูกส่งเข้ามาจาก NETPIE Freeboard แสดงผ่าน Serial Monitor บนโปรแกรม Arduino IDE

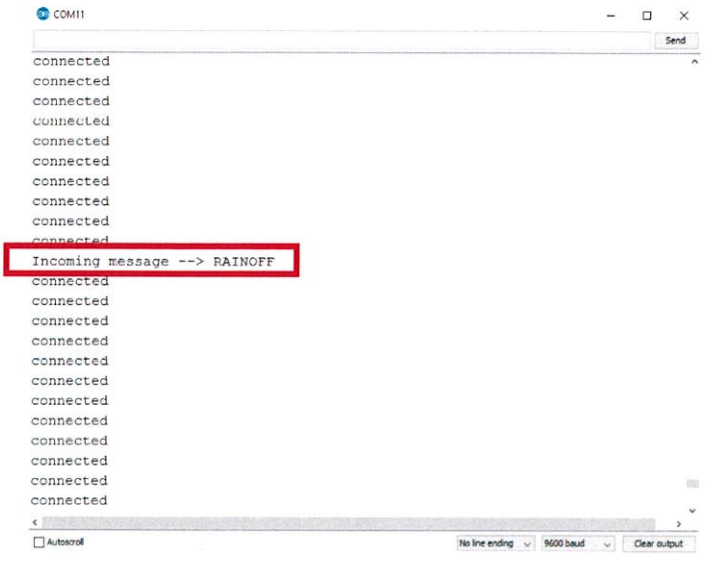


รูปที่ 4.12 NETPIE ขณะกดเปิดสวิตช์ปุ่ม Rainy Mode

จากรูปที่ 4.12 แสดงหน้า NETPIE Freeboard ซึ่งผู้ใช้สามารถควบคุมได้ผ่านอินเตอร์เน็ตเว็บบราวเซอร์ ในภาพเป็นการควบคุมสวิตช์ปุ่ม Rainy Mode ให้มีสถานะ ON

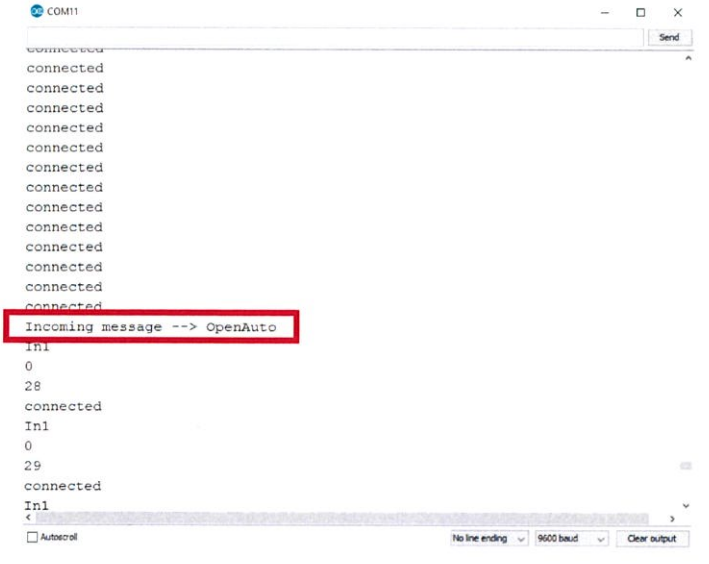


รูปที่ 4.13 Serial Monitor ขณะกดเปิดสวิตช์ปุ่ม Rainy Mode



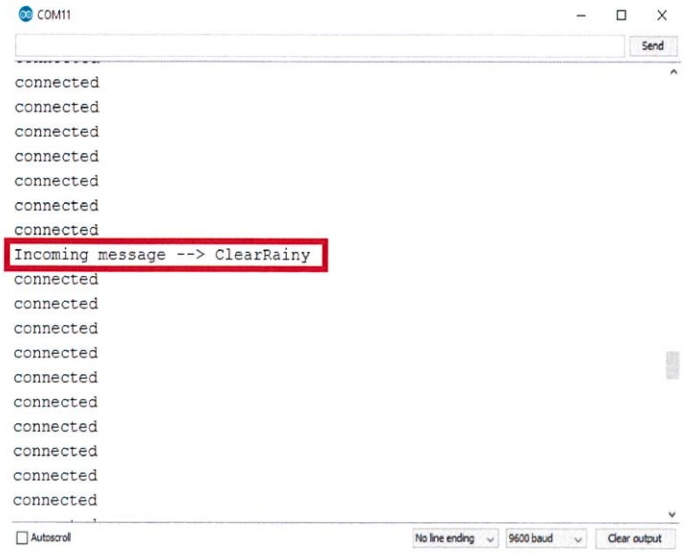
รูปที่ 4.14 Serial Monitor ขณะกดปิดสวิตซ์ปุ่ม Rainy Mode

จากรูปที่ 4.13 เป็นการแสดงข้อความขณะทำการควบคุมสวิตซ์ปุ่ม Rainy Mode ให้มีสถานะ ON ที่ถูกส่งเข้ามาจาก NETPIE Freeboard แสดงผ่าน Serial Monitor บนโปรแกรม Arduino IDE และจากรูปที่ 4.14 เป็นการแสดงข้อความขณะทำการควบคุมสวิตซ์ปุ่ม Rainy Mode ให้มีสถานะ OFF ที่ถูกส่งเข้ามาจาก NETPIE Freeboard แสดงผ่าน Serial Monitor บนโปรแกรม Arduino IDE



รูปที่ 4.15 Serial Monitor ขณะกดเปิดสวิตซ์ปุ่ม Automatic Open

จากรูปที่ 4.15 เป็นการแสดงข้อความขณะทำการควบคุมสวิตซ์ปุ่ม Automatic Open ที่ถูกส่งเข้ามาจาก NETPIE Freeboard แสดงผ่าน Serial Monitor บนโปรแกรม Arduino IDE

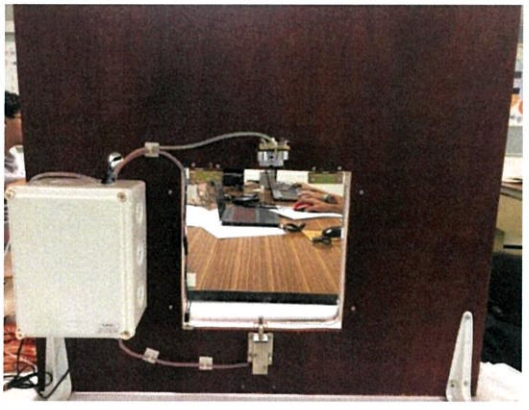


รูปที่ 4.16 Serial Monitor ขณะกดเปิดสวิตซ์ปุ่ม Clear Rainy

จากรูปที่ 4.16 เป็นการแสดงข้อความขณะทำการควบคุมสวิตซ์ปุ่ม Clear Rainy ที่ถูกส่งเข้ามาจาก NETPIE Freeboard แสดงผ่าน Serial Monitor บนโปรแกรม Arduino IDE

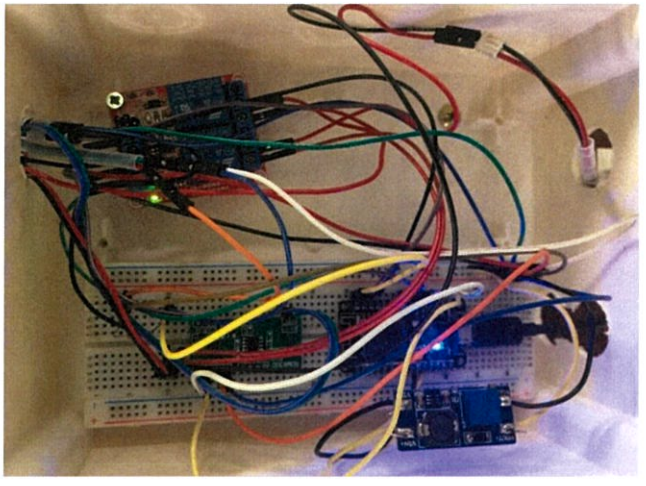
4.5 ผลการทดลองภาพรวมของระบบทั้งหมด

ในส่วนของการทดลอง ในภาพรวมทั้งหมดของระบบประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยงเป็นการทำงานร่วมกันของบอร์ดอาดุยโน่, RF Reader Module, Ultrasonic Module Sensor, Hall Effect Sensor, 2 Relay Module และกลอนไฟฟ้า โดยภาพรวมของระบบถูกแสดงดังรูปที่ 4.17



รูปที่ 4.17 ภาพรวมของประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง

ในส่วนวงจรการทำงาน จะทำการต่ออุปกรณ์ทั้งหมดเข้าด้วยกันดังรูปที่ 4.18

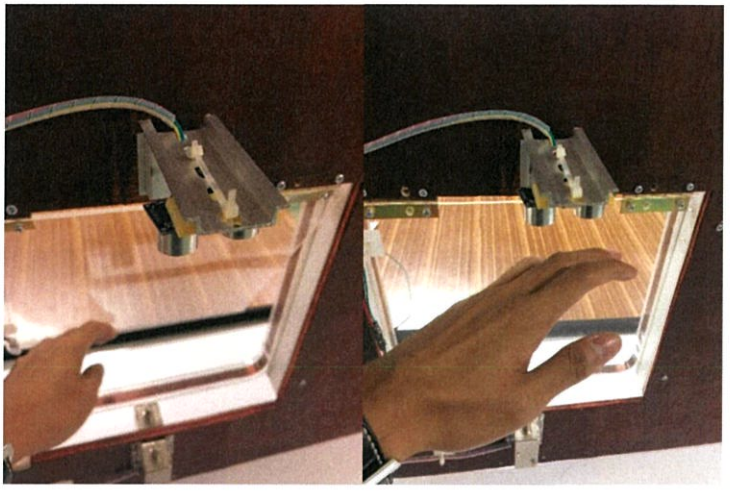


รูปที่ 4.18 วงจรของประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง

การทำงานทางด้านขาเข้าจะเป็นการทำงานของ RFID เมื่อนำ RFID Tag เข้าไปใกล้สายอากาศ (Antenna) RF Reader Module จะส่งสัญญาณเข้า NodeMCU และจะทำการประมวลผล สั่งการให้รีเลย์ปลดล็อคกลอนไฟฟ้าทางด้านใน จากนั้นจะสามารถเปิดเข้าไปด้านในได้ ดังรูปที่ 4.19

รูปที่ 4.19 การทำงานของ RFID Reader Module ในด้านขาเข้า

การทำงานด้านขาออกเป็นการทำงานของ Ultrasonic Sensor Module เมื่อเซนเซอร์ตรวจจับวัตถุที่เข้าใกล้ได้ NodeMCU จะทำการประมวลผล และสั่งการให้รีเลย์ปลดล็อคกลอนไฟฟ้าทางด้านนอก แสดงในรูปที่ 4.20



รูปที่ 4.20 การทำงานของ Ultrasonic Sensor Module ในด้านขาออก

ในส่วนของ Hall Effect Sensor เป็นตัวตรวจจับการเปิด-ปิด และตำแหน่งของบานพับประตูโดยที่จะทำการติดตั้งในส่วนของปลายประตูดังรูปที่ 4.21

รูปที่ 4.21 การทำงานของ Hall Effect Sensor

4.6 ผลการทดลองผ่านหน้าแอปพลิเคชั่นแอนดรอยด์

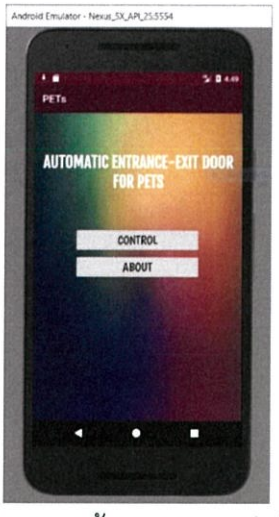
การทดลองแอปพลิเคชั่นบน Android Emulator

1. ทำการเข้าแอปพลิเคชั่น PETs รูปที่ 4.22 ก็จะเข้าสู่หน้าแรกของแอปพลิเคชั่น ดังรูปที่ 4.23

2. คลิกที่ปุ่ม ENTER เพื่อทำการเข้าสู่หน้าควบคุม ดังรูปที่ 4.24

3. ทำการเลื่อนปุ่ม Switch1 จะพบว่าบนหน้าต่างแสดงการเปลี่ยนแปลงดังรูปที่ 4.25 ได้บอกว่า Switch1 ON

รูปที่ 4.22 หน้าแอปพลิเคชั่น



รูปที่ 4.23 หน้าแรกของแอปพลิเคชั่น

รูปที่ 4.24 หน้าควบคุมของแอปพลิเคชั่น

รูปที่ 4.25 เลื่อนปุ่ม Switch1 ON บนแอปพลิเคชั่น

4. ทำการเลื่อนปุ่ม Switch1 จะพบว่าบนหน้าต่างแสดงการเปลี่ยนแปลงดังรูปที่ 4.26 ได้บอกว่า Switch1 OFF

5. ทำการเลื่อนปุ่ม Switch2 จะพบว่าบนหน้าต่างแสดงการเปลี่ยนแปลงดังรูปที่ 4.27 ได้บอกว่า Switch2 ON

6. ทำการเลื่อนปุ่ม Switch2 จะพบว่าบนหน้าต่างแสดงการเปลี่ยนแปลงดังรูปที่ 4.28 ได้บอกว่า Switch2 OFF

7. ทำการเลื่อนปุ่ม Rainy Mode จะพบว่าบนหน้าต่างแสดงการเปลี่ยนแปลงดังรูปที่ 4.29 ได้บอกว่า RainyON

รูปที่ 4.26 เลื่อนปุ่ม Switch1 OFF บนแอปพลิเคชั่น

รูปที่ 4.27 เลื่อนปุ่ม Switch2 ON บนแอปพลิเคชั่น

รูปที่ 4.28 เลื่อนปุ่ม Switch2 OFF บนแอปพลิเคชั่น

รูปที่ 4.29 เลื่อนปุ่ม Rainy Mode ON บนแอปพลิเคชั่น

8. ทำการเลื่อนปุ่ม Rainy Mode จะพบว่าบนหน้าต่างแสดงการเปลี่ยนแปลงดังรูปที่ 4.30 ได้บอกว่า RainOFF

9. หลังจากที่เลื่อนปุ่มให้ RAINOFF จะต้องทำการกดปุ่ม Clear Rainy หลังจากกดปุ่มหน้าต่างแสดงสถานะก็จะขึ้นสถานะของ ClearRainy ดังรูปที่ 4.31

10. เมื่อทำการกดปุ่ม AUTO ON หน้าต่างแสดงสถานะก็จะขึ้นสถานะของ OpenAuto ดังรูปที่ 4.32

รูปที่ 4.30 เลื่อนปุ่ม Rainy Mode OFF บนแอปพลิเคชั่น

รูปที่ 4.31 เลื่อนปุ่ม Clear Rainy บนแอปพลิเคชั่น

รูปที่ 4.32 เลื่อนปุ่ม OpenAuto บนแอปพลิเคชั่น

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์ประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยงนี้ เป็นการทำงานร่วมกันของ NodeMCU, RFID Reader Module, Ultrasonic Module Sensor, Hall Effect Sensor, 2 Relay Module และกลอนไฟฟ้า โดยในการทำงานจะแบ่งเป็น 2 ส่วน คือ ส่วนของการทำงานแบบ Automatic และแบบ Manual โดยในส่วน Automatic จะใช้การรับสัญญาณอินพุตของ Radio Frequency Identification (RFID) และUltrasonic Sensor Module จากนั้นไมโครคอนโทรลเลอร์จะทำการประมวลผลโดย NodeMCU และควบคุมการทำงานโดยรีเลย์ จากนั้นจะตรวจสอบสถานะและตำแหน่งของบานพับประตูโดยใช้ Hall Effect Sensor และส่วน Manual จะเป็นการสั่งงานผ่านแอปพลิเคชั่นแอนดรอยด์

ในการทำงานฝั่ง Automatic จะแบ่งเป็นฝั่งขาเข้า และขาออก โดยฝั่งขาเข้าจะใช้สัญญาณ Radio Frequency Identification (RFID) ในการตรวจสอบสิ่งมีชีวิตที่เข้ามาใกล้เสารับสัญญาณ (Antenna) RFID Reader Module จะเป็นตัวอ่านค่า RFID Tag เพื่อที่จะสั่งงานรีเลย์ให้ทำการปลดล็อคกลอนไฟฟ้า และในด้านขาออกจะใช้ Ultrasonic Module Sensor เป็นตัวตรวจจับสิ่งมีชีวิตที่เข้าใกล้ โดยกำหนดระยะที่ตรวจจับสิ่งมีชีวิตให้มีระยะน้อยกว่า 20 เซนติเมตร หลังจากที่ได้รับสัญญาณเข้ามาแล้ว NodeMCU จะทำการประมวลผล โดยมีรีเลย์ที่ใช้ในการควบคุมการเปิด-ปิดกลอนไฟฟ้า และในส่วนของ Hall Effect Sensor เป็นตัวตรวจจับการเปิด-ปิด และตำแหน่งของบานพับประตู

ต่อมาในการทำงานฝั่ง Manual จะควบคุมโดยแอปพลิเคชั่นแอนดรอยด์ โดยเชื่อมต่อกับ NETPIE ซึ่งการเชื่อมต่อระหว่าง NETPIE กับแอปพลิเคชั่น จะเชื่อมต่อผ่านซอฟต์แวร์ไลบารี่ คือ Microgear (Client Library) โดยการมีปุ่มควบคุมการเปิด-ปิดของรีเลย์ทั้ง 2 ตัว และไฟแสดงสถานะ

5.2 ข้อเสนอแนะ

ในปริญญานิพนธ์ประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง เป็นเพียงแบบจำลองของประตูเท่านั้น และอาจใช้งานไม่ได้ เนื่องด้วยข้อจำกัดของสัตว์เลี้ยง ที่ต้องใช้เวลาในการฝึกฝน และประตูยังสามารถระบุตัวของสัตว์เลี้ยงในการใช้งานประตูได้ โดยการเขียนคำสั่งควบคุมเพิ่มไป รวมไปถึงฟังก์ชั่นต่างๆ ภายในแอปพลิเคชั่นแอนดรอยด์ด้วย

บรรณานุกรม

[1] วิไลลักษณ์. “rfid (radio frequency identification).”
http://rfid-datacom.blogspot.com/

[2] วิทยาลัยสารพัดช่างกำแพงเพชร อาชีวศึกษาจังหวัดกำแพงเพชร วิชางานไฟฟ้าและอิเล็กทรอนิกส์เบื้องต้น. “หน่วยที่ 9 รีเลย์.”
http://kpp.ac.th/elearning/elearning3/book-09.html.

[3] Siraphop Kajaikum. “Hall-Effect Sensors (Magnetic Sensors).”
https://mcu56.learninginventions.org/?page_id=258.

[4] รังสรรค์ ศรีสาคร. “เครื่องวัดสนามแม่เหล็ก.”
https://web.ku.ac.th/schoolnet/snet3/saowalak/hall/hall.htm.

[5] Arvind, Sanjeev. “A Simple Guide to Using a Hall Effect Sensor With Arduino.”
https://diyhacking.com/arduino-hall-effect-sensor-tutorial/.

[6] กองวิจัยและพัฒนา ฝ่ายวิจัยและพัฒนา การไฟฟ้านครหลวง. “ความรู้ความเข้าใจเกี่ยวกับสนามไฟฟ้าและสนามแม่เหล็ก.”
http://mbeddedweekly.blogspot.com/2014/12/ultrasonic-sensor-timers-external.html.

[7] ดร. พนิดา หล่อวงศ์ตระกูล. “บทที่ 11 สนามแม่เหล็ก.”
http://physics.sci.rmutsb.ac.th/images/sheet/CH11.pdf.

[8] Andrew McHutchon. “RLC Resonant Circuits.”
http://mlg.eng.cam.ac.uk/mchutchon/ResonantCircuits.pdf.

[9] Vowstar. “NodeMCU DEVKIT V1.0.”
https://github.com/nodemcu/nodemcu-devkit-v1.0.

[10] BraveNewCode “MQTT”
https://mqtt.org/

[11] ElectronicWings “NodeMCU MQTT Client with ESPlorer IDE”
http://www.electronicwings.com/nodemcu/nodemcu-mqtt-client-with-esplorer-ide

ภาคผนวก ก

โค้ดคำสั่งควบคุมการเปิด-ปิด ประตูเข้า-ออกอัตโนมัติสำหรับสัตว์เลี้ยง

```
#include <EEPROM.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WiFiClient.h>
#include <WiFiClientSecure.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
#include <SoftwareSerial.h>
#include <Ultrasonic.h>

#include <AuthClient.h>
#include <debug.h>
#include <MicroGear.h>
#include <MQTTClient.h>
#include <PubSubClient.h>
#include <SHA1.h>

#define TRIGGER 5   //D1
#define ECHO    4   //D2

#define D7 13       //D7  RX
#define D8 15       //D8  TX

int relay1 =  14;  //D5
int relay2 =  12;  //D6
int hallPin = 16;  //D0

const char* ssid     = "ECC510-2.4Ghz";
const char* password = "telecom510";

#define APPID   "AutoDoorForPet"
#define KEY     "NVeS7ixyQKrSIWX"
#define SECRET  "3KvF6nvfnFhvecPNVmBaVgjKb"
#define ALIAS   "RelayControl"
```

```
SoftwareSerial RFID(D7,D8);              // RX | TX
Ultrasonic ultrasonic(D1,D2);            // Trig PIN,Echo PIN

WiFiClient client;

char k;
int a,s;
int r=0;
int hallState=0;

int timer=0;
int i=1;
MicroGear microgear(client);

/* If a new message arrives, do this */
void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen)
{
    Serial.print("Incoming message --> ");
    msg[msglen] = '\0';

char strState[msglen];
      for (int i = 0; i < msglen; i++)
      {
        strState[i] = (char)msg[i];
        Serial.print((char)msg[i]);
      }
      Serial.println();

      String stateStr = String(strState).substring(0, msglen);

   //========== ช่วงประมวลผลคำสั่ง ==========


          if (stateStr == "SW1ON")
```

```
{
 digitalWrite(relay1,LOW);
 digitalWrite(relay2,HIGH);
 r = 99;
 microgear.chat("sw1","ON");
}
if (stateStr == "SW1OFF")
{
 digitalWrite(relay1,LOW);
 digitalWrite(relay2,LOW);
 r = 99;
 microgear.chat("sw1","OFF");
}
if (stateStr == "SW2ON")
{
 digitalWrite(relay1,HIGH);
 digitalWrite(relay2,LOW);
 r=99;
 microgear.chat("sw2","ON");
}
if (stateStr == "SW2OFF")
{
 digitalWrite(relay1,LOW);
 digitalWrite(relay2,LOW);
 r = 99;
 microgear.chat("sw2","OFF");
}
if (stateStr == "RAINON")
{
 digitalWrite(relay1,LOW);
 digitalWrite(relay2,LOW);
 r = 99;
 microgear.chat("rain","ON");
}
```

```
            if (stateStr == "RAINOFF")
            {
             digitalWrite(relay1,HIGH);
             digitalWrite(relay2,HIGH);
             r = 99;
             microgear.chat("rain","OFF");
            }
            if (stateStr == "OpenAuto")
            {
             digitalWrite(relay1,HIGH);
             digitalWrite(relay2,HIGH);
             r = 0
             ;microgear.chat("Auto","OPEN");
            }
            if (stateStr == "ClearRainy")
            {
             digitalWrite(relay1,LOW);
             digitalWrite(relay2,LOW);
             r = 99;
             microgear.chat("Clear","CLOSE");
            }
}

void onFoundgear(char *attribute, uint8_t* msg, unsigned int msglen) {
    Serial.print("Found new member --> ");
    for (int i=0; i<msglen; i++)
    Serial.print((char)msg[i]);
    Serial.println();
}

void onLostgear(char *attribute, uint8_t* msg, unsigned int msglen) {
    Serial.print("Lost member --> ");
    for (int i=0; i<msglen; i++)
    Serial.print((char)msg[i]);
```

```
    Serial.println();
}

/* When a microgear is connected, do this */
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen) {
    Serial.println("Connected to NETPIE...");
    /* Set the alias of this microgear ALIAS */
    microgear.setAlias(ALIAS);
}

void setup() {
    //ESP.wdtDisable(); ESP.wdtEnable(WDTO_8S);
    microgear.on(MESSAGE,onMsghandler);
    microgear.on(PRESENT,onFoundgear);
    microgear.on(ABSENT,onLostgear);
    microgear.on(CONNECTED,onConnected);

    RFID.begin(9600);
    Serial.begin(9600);
    Serial.println("Starting...");
    if (WiFi.begin(ssid, password)) {
        while (WiFi.status() != WL_CONNECTED) {
            delay(500);
            Serial.print(".");
        }
    }

    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
```

```
microgear.init(KEY,SECRET,ALIAS);
    microgear.connect(APPID);
    pinMode(relay1, OUTPUT);
    pinMode(relay2, OUTPUT);
    digitalWrite(relay1, LOW);
    digitalWrite(relay2, LOW);
    pinMode(hallPin, INPUT);
    pinMode(D7, INPUT);
    pinMode(D8, OUTPUT);
}

void loop()
{
        ESP.wdtFeed();
        if(microgear.connected())
        {
          Serial.println("connected");
          microgear.loop();
        }
        else
        {
        Serial.println("connection lost, reconnect...");
        if(timer >= 5000)
        {
          microgear.connect(APPID);
          timer = 0;
        }
        else timer += 100;
        }
        delay(100);

        s = ultrasonic.Ranging(CM);
        hallState = digitalRead(hallPin);
        i = RFID.read();
```

```
                if (RFID.available() == 0 && hallState == LOW && r == 0)     // No
RFID and no sensor
                {
                  digitalWrite(relay1, LOW);
                  digitalWrite(relay2, LOW);
                  Serial.println("In1");
                  Serial.println(a);
                  Serial.println(s);
                  delay(500);
                }

                if (RFID.available() > 0 && hallState == LOW && r == 0 && s > 20)
                {
                  digitalWrite(relay1, LOW);
                  digitalWrite(relay2, HIGH);
                  Serial.println("In2");
                  Serial.println(a);
                  Serial.println(s);
                  delay(7500);

                  hallState = digitalRead(hallPin);
                  if (hallState == LOW && r == 0)
                  {
                    digitalWrite(relay1, LOW);
                    digitalWrite(relay2, LOW);
                    Serial.println("In3");
                    Serial.println(a);
                    serialFlush();
                    delay(5000);
                  }
                  else if (hallState == HIGH)
                  {
                    r = 1;
```

```
      digitalWrite(relay2, HIGH);
      Serial.println("In4");
      Serial.println(a);
    }
  }

  hallState = digitalRead(hallPin);
  if (hallState == LOW && r == 1)
  {
    digitalWrite(relay1, LOW);
    digitalWrite(relay2, LOW);
    Serial.println("In5");
    Serial.println(a);
    serialFlush();
    delay (5000);
    r = 0;
  }

  if (RFID.available() == 0 && hallState == 0 && r == 0 &&
(s==1||s==2||s==3||s==4||s==5||s==6||s==7||s==8||s==9||s==10||s==11||s==12||s==13||s=
=14||s==15||s==16||s==17||s==18||s==19))
  {
    digitalWrite(relay1, HIGH);
    digitalWrite(relay2, LOW);
    r = 2;
    Serial.println("Out2");
    Serial.println(a);
    Serial.println(s);
    delay(7500);

    hallState = digitalRead(hallPin);
    if (hallState == LOW && r == 2)
    {
      digitalWrite(relay1, LOW);
```

```
      digitalWrite(relay2, LOW);
      Serial.println("Out3");
      r = 0;
      Serial.println(a);
      delay(5000);
    }
    else if (hallState == HIGH && r == 2)
    {
      r = 3;
      digitalWrite(relay1, HIGH);
      Serial.println("out4");
      Serial.println(a);
    }
  }

  hallState = digitalRead(hallPin);
  if (hallState == LOW && r == 3)
  {
    digitalWrite(relay1, LOW);
    digitalWrite(relay2, LOW);
    delay (5000);
    Serial.println("out5");
    Serial.println(a);
    r = 0;
  }

}

void serialFlush() {                    // Function for clear buffer
  while (RFID.available() > 0) {
    char k = RFID.read();
    r = 0;
  }
}
```

ภาคผนวก ข

โค้ดการเขียนแอปพลิเคชั่นแอนดรอยด์

Application

mainfests

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.jameswich.testnetpie">

    <application
        android:allowBackup="true"
        android:icon="@drawable/iconnn"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".Main2Activity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity" />

        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />

        <activity android:name=".Main3Activity"></activity>
    </application>

</manifest>
```

Java

```
package com.example.jameswich.testnetpie;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import java.util.ArrayList;
import java.util.IllegalFormatCodePointException;

import io.netpie.microgear.Microgear;
import io.netpie.microgear.MicrogearEventListener;

public class MainActivity extends AppCompatActivity {

    private Microgear microgear = new Microgear(this);
    private String appid = "AutoDoorForPet"; //APP_ID
    private String key = "NVeS7ixyQKrSIWX"; //KEY
    private String secret = "3KvF6nvfnFhvecPNVmBaVgjKb"; //SECRET
    private String alias = "RelayControl";
    //private String alias = "A2";

    private ArrayList<String> test= new ArrayList<>();
```

```
    boolean state_SW1 = false;
    boolean state_SW2 = false;
    boolean state_SW3 = false;

    Handler handler = new Handler() {
        @Override

        public void handleMessage(Message msg) {
            Bundle bundle = msg.getData();
            String string = bundle.getString("myKey");

            test.add(string);
            //myTextView.setVisibility(View.INVISIBLE);
            ArrayAdapter<String> adapter = new
ArrayAdapter<String>(MainActivity.this,android.R.layout.simple_list_item_1,test);

            //myTextView.append(string+"\n");
        }

    };


    @SuppressLint("WrongViewCast")
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        MicrogearCallBack callback = new MicrogearCallBack();
        microgear.connect(appid,key,secret,alias);
        microgear.setCallback(callback);
        microgear.subscribe("RelayControl");
```

```
        findViewById(R.id.CR).setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View v){
                //microgear.publish("Topictest","SW1OFF");
                microgear.chat("RelayControl","ClearRainy");

Toast.makeText(getBaseContext(),"ClearRainy",Toast.LENGTH_SHORT).show();

            }
        });

        findViewById(R.id.AutoON).setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View v){
                //microgear.publish("Topictest","SW2ON");
                microgear.chat("RelayControl","OpenAuto");
                Toast.makeText(getBaseContext(), "OpenAuto",
Toast.LENGTH_SHORT).show();
            }
        });

        findViewById(R.id.swTest).setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {

                if(!state_SW1){
                    state_SW1 = true;
                    microgear.chat("RelayControl","SW1ON");
                    Toast.makeText(getBaseContext(),"Exit is
ON",Toast.LENGTH_SHORT).show();
                }
```

```
            else{
                state_SW1 = false;
                microgear.chat("RelayControl","SW1OFF");
                Toast.makeText(getBaseContext(),"Exit is
OFF",Toast.LENGTH_SHORT).show();
            }

        }

    });


    findViewById(R.id.switch2).setOnClickListener(new View.OnClickListener(){

        @Override
        public void onClick(View v) {


            if(!state_SW2){
                state_SW2 = true;
                microgear.chat("RelayControl","SW2ON");
                Toast.makeText(getBaseContext(),"Entrance is
ON",Toast.LENGTH_SHORT).show();
            }
            else{
                state_SW2 = false;
                microgear.chat("RelayControl","SW2OFF");
                Toast.makeText(getBaseContext(),"Entrance is
OFF",Toast.LENGTH_SHORT).show();
            }

        }
    });
```

```
        findViewById(R.id.switch3).setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View v) {
                if(!state_SW3){
                    state_SW3 = true;
                    microgear.chat("RelayControl","RAINON");
                    Toast.makeText(getBaseContext(),"RainMode is
ON",Toast.LENGTH_SHORT).show();
                }
                else{
                    state_SW3 = false;
                    microgear.chat("RelayControl","RAINOFF");
                    Toast.makeText(getBaseContext(),"RainMode is
OFF",Toast.LENGTH_SHORT).show();
                }

            }
        });

    }

        /*(new Thread(new Runnable()
        {
            int count = 1;
            @Override
            public void run()
            {
                while (!Thread.interrupted())
                    try
                    {
                        runOnUiThread(new Runnable() // start actions in UI thread
                        {
                            @Override
```

```
                    public void run(){
                        //SW2ON
                        //microgear.publish("Topictest", String.valueOf(count)+".  Test
message");
                        microgear.publish("Topictest", "SW2ON");
                        count++;

                    }
                });
                Thread.sleep(2000);
            }
            catch (InterruptedException e)
            {
                // ooops
            }
        }
    }).start();*/


    protected void onDestroy() {
        super.onDestroy();
        microgear.disconnect();
    }

    protected void onResume() {
        super.onResume();
        microgear.bindServiceResume();
    }

    class MicrogearCallBack implements MicrogearEventListener {
        @Override
        public void onConnect() {
            Message msg = handler.obtainMessage();
            Bundle bundle = new Bundle();
```

```
        bundle.putString("myKey", "Now I'm connected with netpie");
        msg.setData(bundle);
        handler.sendMessage(msg);
        Log.i("Connected","Now I'm connected with netpie");
    }

    @Override
    public void onMessage(String topic, String message) {
        Message msg = handler.obtainMessage();
        Bundle bundle = new Bundle();
        bundle.putString("myKey", topic+" : "+message);
        msg.setData(bundle);
        handler.sendMessage(msg);
        Log.i("Message",topic+" : "+message);
    }

    @Override
    public void onPresent(String token) {
        Message msg = handler.obtainMessage();
        Bundle bundle = new Bundle();
        bundle.putString("myKey", "New friend Connect :"+token);
        msg.setData(bundle);
        handler.sendMessage(msg);
        Log.i("present","New friend Connect :"+token);
    }

    @Override
    public void onAbsent(String token) {
        Message msg = handler.obtainMessage();
        Bundle bundle = new Bundle();
        bundle.putString("myKey", "Friend lost :"+token);
        msg.setData(bundle);
        handler.sendMessage(msg);
        Log.i("absent","Friend lost :"+token);
```

```
        }

        @Override
        public void onDisconnect() {
            Message msg = handler.obtainMessage();
            Bundle bundle = new Bundle();
            bundle.putString("myKey", "Disconnected");
            msg.setData(bundle);
            handler.sendMessage(msg);
            Log.i("disconnect","Disconnected");
        }

        @Override
        public void onError(String error) {
            Message msg = handler.obtainMessage();
            Bundle bundle = new Bundle();
            bundle.putString("myKey", "Exception : "+error);
            msg.setData(bundle);
            handler.sendMessage(msg);
            Log.i("exception","Exception : "+error);
        }

        @Override
        public void onInfo(String info) {
            Message msg = handler.obtainMessage();
            Bundle bundle = new Bundle();
            bundle.putString("myKey", "Exception : "+info);
            msg.setData(bundle);
            handler.sendMessage(msg);
            Log.i("info","Info : "+info);
        }
    }

}
```

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Main2Activity extends AppCompatActivity {
Button Buttlink;
Button Butt1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        Buttlink=(Button) findViewById(R.id.button);
        Buttlink.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent a = new Intent(Main2Activity.this,MainActivity.class);
                startActivity(a);


            }
        });

        Butt1=(Button) findViewById(R.id.button2);

        Butt1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent b = new Intent(Main2Activity.this,Main3Activity.class);
                startActivity(b);
```

```
            }
        });
    }
}

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class Main3activity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
    }
}
```

XML

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back1"
    tools:context="com.example.jameswich.testnetpie.MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="36dp"
        android:textColor="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.498" />


    <Button
        android:id="@+id/CR"
        android:layout_width="350dp"
        android:layout_height="45dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:fontFamily="@font/boogaloo"
        android:text="Clear Rainy"
        android:textSize="20dp"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintEnd_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.308" />

<Button
    android:id="@+id/AutoON"
    android:layout_width="350dp"
    android:layout_height="45dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:fontFamily="@font/boogaloo"
    android:text="AUTO ON"
    android:textSize="20dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.408" />


<Switch
    android:id="@+id/swTest"
    android:layout_width="300dp"
    android:layout_height="40dp"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="8dp"
    android:fontFamily="@font/boogaloo"
    android:text="Entrance"
    android:textColor="#FFFFFF"
```

```
    android:textSize="35sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.017" />

<Switch
    android:id="@+id/switch2"
    android:layout_width="300dp"
    android:layout_height="40dp"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="8dp"
    android:fontFamily="@font/boogaloo"
    android:text="Exit"
    android:textColor="#FFFFFF"
    android:textSize="35sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.105" />

<Switch
    android:id="@+id/switch3"
    android:layout_width="300dp"
    android:layout_height="40dp"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="104dp"
    android:fontFamily="@font/boogaloo"
    android:text="Rainy Mode"
    android:textColor="#FFFFFF"
```

```
        android:textSize="35sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="@+id/listItem"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.216" />

</android.support.constraint.ConstraintLayout>


<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back1"
    tools:context="com.example.jameswich.testnetpie.Main2Activity">


    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:fontFamily="@font/fjalla_one"
        android:text="AUTOMATIC ENTRANCE-EXIT DOOR FOR PETS"
        android:textAlignment="center"
        android:textColor="#FFFFFF"
        android:textSize="30sp"
```

```
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.136" />

<Button
    android:id="@+id/button"
    android:layout_width="250dp"
    android:layout_height="50dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:fontFamily="@font/boogaloo"
    android:text="CONTROL"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.402" />

<Button
    android:id="@+id/button2"
    android:layout_width="250dp"
    android:layout_height="50dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:fontFamily="@font/boogaloo"
    android:text="ABOUT"
```

```
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>


<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back1"
    tools:context="com.example.jameswich.testnetpie.Main3Activity">

    <TextView
        android:layout_width="100dp"
        android:layout_height="46dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="ABOUT"
        android:textColor="#FFFFFF"
        android:textSize="30dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.029"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
```

```
    app:layout_constraintVertical_bias="0.035" />

<TextView
    android:layout_width="350dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="@string/t1"
    android:textAlignment="viewStart"
    android:textColor="#FFFFFF"
    android:textSize="16dp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.234" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="Author"
    android:textColor="#FFFFFF"
    android:textSize="25dp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.029"
```

```
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.645" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="@string/n1"
    android:textColor="#FFFFFF"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.033"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.697" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="@string/n2"
    android:textColor="#FFFFFF"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.033"
```

```
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.737" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="@string/n3"
    android:textColor="#FFFFFF"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.036"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.777" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="@string/n4"
    android:textColor="#FFFFFF"
    android:textSize="30dp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintHorizontal_bias="0.03"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.856" />

    <TextView
        android:layout_width="216dp"
        android:layout_height="18dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="@string/n5"
        android:textColor="#FFFFFF"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.052"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.901" />


</android.support.constraint.ConstraintLayout>


<?xml version="1.0" encoding="utf-8"?>
<resources>

    <color name="colorPrimary">#360323</color>
    <color name="colorPrimaryDark">#4e0422</color>
    <color name="colorAccent">#3dfa89</color>

</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="com_google_android_gms_fonts_certs">
        <item>@array/com_google_android_gms_fonts_certs_dev</item>
        <item>@array/com_google_android_gms_fonts_certs_prod</item>
    </array>
    <string-array name="com_google_android_gms_fonts_certs_dev">
        <item>
```

MIIEqDCCA5CgAwIBAgIJANWFuGx90071MA0GCSqGSIb3DQEBBAUAMIGUMQswCQYDVQ
QGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEWMBQGA1UEBxMNTW91bnRhaW4g
VmlldzEQMA4GA1UEChMHQW5kcm9pZDEQMA4GA1UECxMHQW5kcm9pZDEQMA4GA
1UEAxMHQW5kcm9pZDEiMCAGCSqGSIb3DQEJARYTYW5kcm9pZEBhbmRyb2lkLmNvb
TAeFw0wODA0MTUyMzM2NTZaFw0zNTA5MDEyMzM2NTZaMIGUMQswCQYDVQQGEw
JVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEWMBQGA1UEBxMNTW91bnRhaW4gVmlld
zEQMA4GA1UEChMHQW5kcm9pZDEQMA4GA1UECxMHQW5kcm9pZDEQMA4GA1UEAx
MHQW5kcm9pZDEiMCAGCSqGSIb3DQEJARYTYW5kcm9pZEBhbmRyb2lkLmNvbTCCAS
AwDQYJKoZIhvcNAQEBBQADggENADCCAQgCggEBANbOLggKv+IxTdGNs8/TGFy0PTP6D
HThvbbR24kT9ixcOd9W+EaBPWW+wPPKQmsHxajtWjmQwWfna8mZuSeJS48LIgAZlKk
pFeVyxW0qMBujb8X8ETrWy550NaFtI6t9+u7hZeTfHwqNvacKhp1RbE6dBRGWynwMVX
8XW8N1+UjFaq6GCJukT4qmpN2afb8sCjUigq0GuMwYXrFVee74bQgLHWGJwPmvmLHC
69EH6kWr22ijx4OKXlSIx2xT1AsSHee70w5iDBiK4aph27yH3TxkXy9V89TDdexAcKk/cVHY
NnDBapcavl7y0RiQ4biu8ymM8Ga/nmzhRKya6G0cGw8CAQOjgfwwgfkwHQYDVR0OBBY
EFI0cxb6VTEM8YYY6FbBMvAPyT+CyMIHJBgNVHSMEgcEwgb6AFI0cxb6VTEM8YYY6FbB
MvAPyT+CyoYGapIGXMIGUMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm
5pYTEWMBQGA1UEBxMNTW91bnRhaW4gVmlldzEQMA4GA1UEChMHQW5kcm9pZDEQ
MA4GA1UECxMHQW5kcm9pZDEQMA4GA1UEAxMHQW5kcm9pZDEiMCAGCSqGSIb3DQ
EJARYTYW5kcm9pZEBhbmRyb2lkLmNvbYIJANWFuGx90071MAwGA1UdEwQFMAMBAf
8wDQYJKoZIhvcNAQEEBQADggEBABnTDPEF+3iSP0wNfdIjIz1AlnrPzgAIHVvXxunW7SBrD
hEglQZBbKJEk5kT0mtKoOD1JMrSu1xuTKEBahWRbqHsXclaXjoBADb0kkjVEJu/Lh5hgYZ
nOjvlba8Ld7HCKePCVePoTJBdI4fvugnL8TsgK05aIskyY0hKI9L8KfqfGTl1lzOv2KoWD0KW
wtAWPoGChZxmQ+nBli+gwYMzM1vAkP+aayLe0a1EQimlOalO762r0GXO0ks+UeXde2Z
4e+8S/pf7pITEI/tP+MxJTALw9QUWEv9lKTk+jkbqxbsh8nfBUapfKqYn0eidpwq2AzVp3ju

```
Yl7//fKnaPhJD9gs=
        </item>
    </string-array>
    <string-array name="com_google_android_gms_fonts_certs_prod">
        <item>

MIIEQzCCAyugAwIBAgIJAMLgh0ZkSjCNMA0GCSqGSIb3DQEBBAUAMHQxCzAJBgNVBAYT
AlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYDVQQHEw1Nb3VudGFpbiBWaW
V3MRQwEgYDVQQKEwtHb29nbGUgSW5jLjEQMA4GA1UECxMHQW5kcm9pZDEQMA4GA
1UEAxMHQW5kcm9pZDAeFw0wODA4MjEyMzEzMzRaFw0zNjAxMDcyMzEzMzRaMHQx
CzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYDVQQHEw1Nb3V
udGFpbiBWaWV3MRQwEgYDVQQKEwtHb29nbGUgSW5jLjEQMA4GA1UECxMHQW5kcm
9pZDEQMA4GA1UEAxMHQW5kcm9pZDCCASAwDQYJKoZIhvcNAQEBBQADggENADCCA
QgCggEBAKtWLgDYO6IIrgqWbxJOKdoR8qtW0I9Y4sypEwPpt1TTcvZApxsdyxMJZ2JORla
nd2qSGT2y5b+3JKkedxiLDmpHpDsz2WCbdxgxRczfey5YZnTJ4VZbH0xqWVW/8lGmPav
5xVwnliJS6HXk+BVKZF+JcWjAsb/GEuq/eFdpuzSqeYTcfi6idkyugwfYwXFU1+5fZKUaRKY
CwkkFQVfcAs1fXA5V+++FGfvjJ/CxURaSxaBvGdGDhfXE28LWuT9ozCl5xw4Yq5OGazvV2
4mZVSoOO0yZ31j7kYvtwYK6NeADwbSxDdJEqO4k//0zOHKrUiGYXtqw/A0LFFtqoZKFjn
kCAQOjgdkwgdYwHQYDVR0OBBYEFMd9jMIhF1Ylmn/Tgt9r45jk14alMIGmBgNVHSMEgZ4
wgZuAFMd9jMIhF1Ylmn/Tgt9r45jk14aloXikdjB0MQswCQYDVQQGEwJVUzETMBEGA1UE
CBMKQ2FsaWZvcm5pYTEWMBQGA1UEBxMNTW91bnRhaW4gVmlldzEUMBIGA1UEChM
LR29vZ2xlIEluYy4xEDAOBgNVBAsTB0FuZHJvaWQxEDAOBgNVBAMTB0FuZHJvaWSCCQ
DC4IdGZEowjTAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBAUAA4IBAQBt0lLO74Uw
LDYKqs6Tm8/yzKkEu116FmH4rkaymUIE0P9KaMftGlMexFlaYjzmB2OxZyl6euNXEsQH8g
jwyxCUKRJNexBiGcCEyj6z+a1fuHHvkiaai+KL8W1EyNmgjmyy8AW7P+LLlkR+ho5zEHatR
bM/YAnqGcFh5iZBqpknHf1SKMXFh4dd239FJ1jWYfbMDMy3NS5CTMQ2XFI1MvcyUTdZ
PErjQfTbQe3aDQsQcafEQPD+nqActifKZ0Np0IS9L9kR/wbNvyz6ENwPiTrjV2KRkEjH78ZM
cUQXg0L3BYHJ3lc69Vs5Ddf9uUGGMYldX3WfMBEmh/9iFBDAaTCK
        </item>
    </string-array>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="ic_launcher_background">#26A69A</color>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="preloaded_fonts" translatable="false">
        <item>@font/abril_fatface</item>
        <item>@font/boogaloo</item>
        <item>@font/bungee</item>
        <item>@font/bungee_inline</item>
        <item>@font/fjalla_one</item>
        <item>@font/freckle_face</item>
        <item>@font/keania_one</item>
        <item>@font/monoton</item>
        <item>@font/smythe</item>
        <item>@font/viga</item>
        <item>@font/yanone_kaffeesatz_bold</item>
    </array>
</resources>
```

```
<resources>

    <string name="app_name">PETs</string>
    <color name="yellow">#FFC300</color>
    <color name="black">#020202</color>
    <string name="t1">This thesis aims to design and create the automatic entrance-exit door for pets. A device is collaborated between microcontroller, RFID reader, Ultrasonic sensor module and application. This thesis is divided into two parts: the first part is the opening and closing solenoid using RFID reader for the entrance and using Ultrasonic sensor module for the exit. For checking the opening and closing of
```

```
the hinges, use hall effect sensor. The second part is the opening and closing
solenoid control via the internet, control by android's application.</string>
    <string name="n1">Nattapong Chaiyarat</string>
    <string name="n2">Patima Netlomwong</string>
    <string name="n3">Phattharaphorn Phimpa</string>
    <string name="n4">Advisor</string>
<string name="n5">Asst.Prof.Dr. Siraphop Tooprakai</string>

</resources>


<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

Microgear

Java

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="io.netpie.microgear">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>

    <application
        android:allowBackup="true"
        android:label="@string/app_name"
        android:supportsRtl="true">
        <service android:name=".MicrogearService" />
    </application>

</manifest>

<resources>
    <string name="app_name">Microgear</string>
</resources>
```

Gradle Scripts

build.gradle(Project: Microgear-test-microgear)

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.0.1'


        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

build.gradle (Module: app)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.jameswich.testnetpie"
        minSdkVersion 18
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-
core:3.0.1'
    compile 'io.netpie:microgear:1.1.3'
}
```

build.gradle (Module: microgear)

```
apply plugin: 'com.android.library'

ext {
    PUBLISH_GROUP_ID = 'io.netpie'
    PUBLISH_ARTIFACT_ID = 'microgear'
    PUBLISH_VERSION = '1.1.3'
}

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.0"

    defaultConfig {
        minSdkVersion 15
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
```

```
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.0.1'
    testCompile 'junit:junit:4.12'
    compile files('libs/BASE64Decoder.jar')
    compile files('libs/org.eclipse.paho.client.mqttv3-1.0.2.jar')
}

apply from: 'https://raw.githubusercontent.com/blundell/release-android-
library/master/android-release-aar.gradle'
```

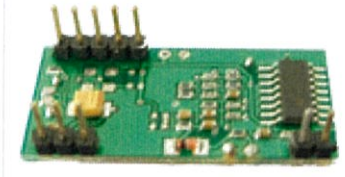
gradle-wrapper.properties(Gradle Version)

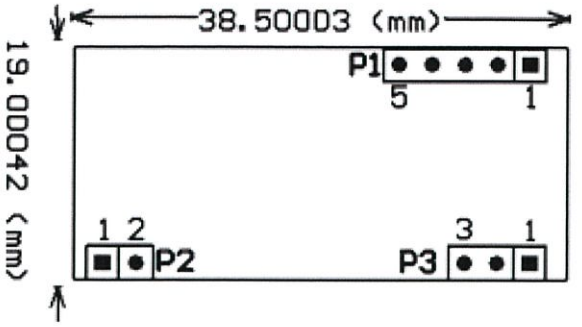
```
#Sat Mar 03 22:51:31 ICT 2018
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-4.1-all.zip
```

ภาคผนวก ค

DATASHEET ของอุปกรณ์ต่างๆ

RDM630 Specification





1. Pin Definition (WEIGAND):

P1:

PIN1	DATA0
PIN2	DATA1
PIN3	GND
PIN4	
PIN5	+5V(DC)

P2:

PIN1	ANT1
PIN2	ANT2

P3:

PIN1	LED
PIN2	+5V(DC)
PIN3	GND

2. Pin definition (TTL interface RS232 data format):

P1:

PIN1	TX
PIN2	RX
PIN3	GND
PIN4	
PIN5	+5V(DC)

P2:

PIN1	ANT1
PIN2	ANT2

P3:

PIN1	LED
PIN2	+5V(DC)
PIN3	GND

Specification and Parameter:

Frequency	125KHz
Baud Rate	9600 (TTL Electricity Level RS232 format)
interface	Weigang26 Or TTL Electricity Level RS232 format
Power supply	DC 5V (±5%)
Current	< 50Ma
Operating range	>50mm (Depend on Card/Tag shape, manufacturer)
Expand I/O port	N/A
Indication light	N/A
Working temperature	-10℃~ +70℃
Storage temperature	-20℃~ +80℃
Max. humidity	Relative humidity 0 ~ 95%
Size	38.5mm×19mm×9mm

Wiegand Format 26bit output data format

Read only ID as 4 byte, form as below:-

:

D37	D36	D35	D34	D33	D32	D31	D30
D27	D26	D25	D24	D23	D22	D21	D20
D17	D16	D15	D14	D13	D12	D11	D10
D07	D06	D05	D04	D03	D02	D01	D00

Wiegand Format 26 bits format form by 26 bits data which including 24 bit user data and 2 bit parity bit. As far as module concerned, former 12bit of 24 bit data is even & the behind 12bits is odd.

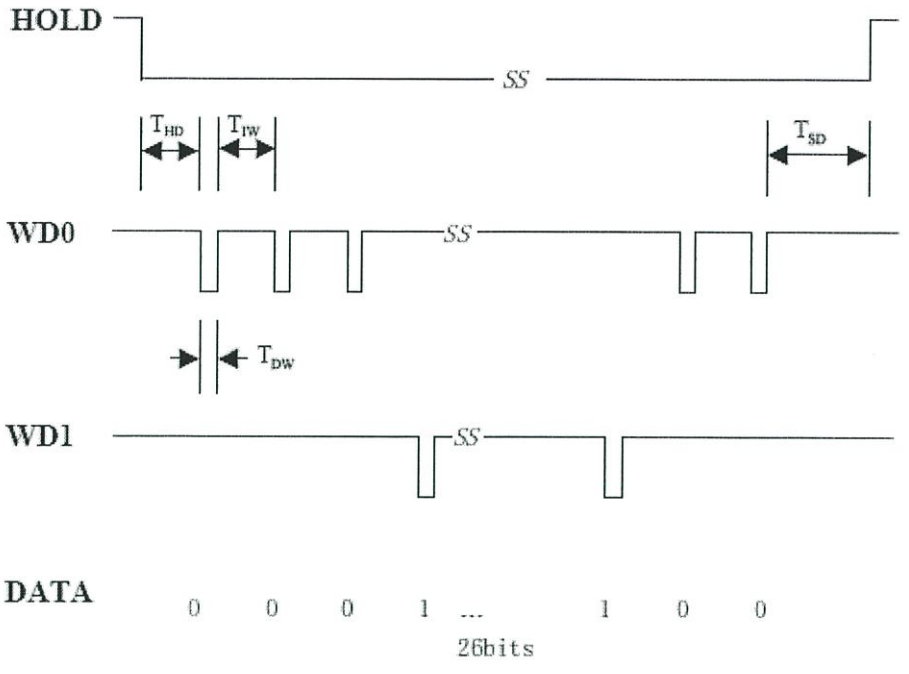
24 bit data correspond to 32 bit is read only 24bit which behind ID, I.E. D27-D20, D17-D10, D07-D00,

Output data format as chart below:

位	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	PE	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	PO
		E	E	E	E	E	E	E	E	E	E	E	E	O	O	O	O	O	O	O	O	O	O	O	O	
		D27	D26	D25	D24	D23	D22	D21	D20	D17	D16	D15	D14	D13	D12	D11	D10	D07	D06	D05	D04	D03	D02	D01	D00	

Thereinto :

- PE is even bit, PO is odd bit;
- E is the data bit which was involved in even, O is the data bit which was involved in odd;
- DXX is the data bit which correspond to Mifare@ Standard & Light card read only ID;
- Wiegand Format 26bit output time sequence:



Symbol	Specification	Representative value
T_{HD}	Send data to active time extension	2ms
T_{SD}	Send data to finish time extension	2ms
T_{DW}	Data impulse width	80μs
T_{IW}	Data impulse interval width	1ms

Reading Type :

Standard design -- card can't be read again within the range of read antenna after read once. User must move it out of reading antenna range first, swipe card again & can be read.

Special design -- card can be read continuously within the range of read antenna.

TTL Interface RS232 Data output format

1. 9600bps,N,8,1
2. CHECKSUM: card 10byte DATA entire do XOR operation

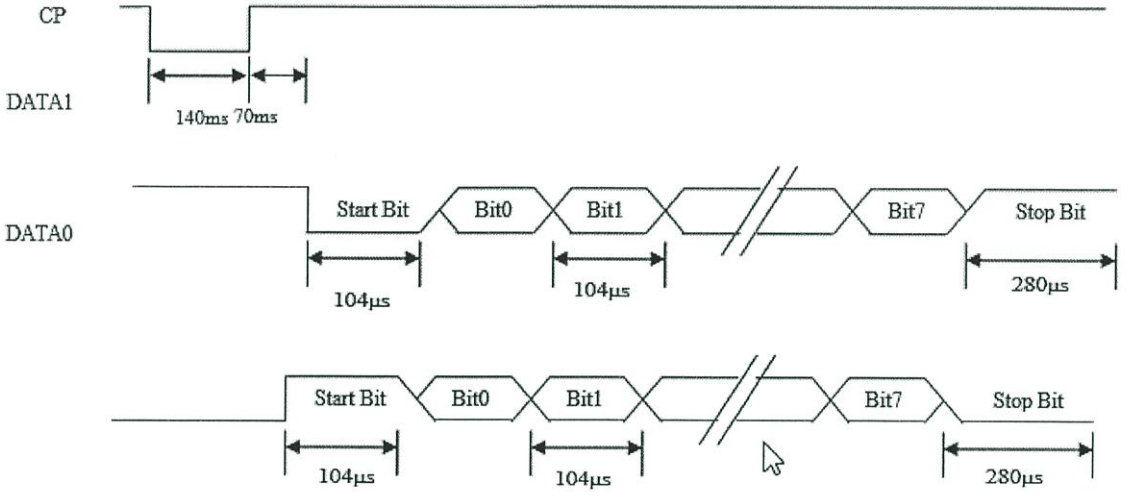
02	10ASCII Data Characters	Checksum	03

Example: card number: 62E3086CED

Output data:36H、32H、45H、33H、30H、38H、36H、43H、45H、44H

CHECKSUM: (62H)XOR (E3H)XOR (08H)XOR (6CH)XOR (EDH)=08H

2. Time sequence chart



AEROSEMI

MT3608

High Efficiency 1.2MHz 2A Step Up Converter

FEATURES

- Integrated 80mΩ PowerMOSFET
- 2V to 24V InputVoltage
- 1.2MHz Fixed SwitchingFrequency
- Internal 4A Switch Current Limit
- Adjustable Output Voltage
- Internal Compensation
- Up to 28V OutputVoltage
- Automatic Pulse Frequency Modulation Mode at LightLoads
- up to 97% Efficiency
- Available in a 6-Pin SOT23-6 Package

APPLICATIONS

- Battery-Powered Equipment
- Set-Top Boxed
- LCD Bais Supply
- DSL and Cable Modems and Routers
- Networking cards powered from PCI or PCI express slots

GENERAL DESCRIPTION

The MT3608 is a constant frequency, 6-pin SOT23 current mode step-up converter intended for small, low power applications. The MT3608 switchesat 1.2MHz and allows the use of tiny, low cost capacitors and inductors 2mm or less in height. Internal soft-start results in small inrush currentand extends battery life.

The MT3608 features automatic shifting to pulse frequency modulation mode at light loads. The MT3608 includes under-voltage lockout, current limiting, and thermal overload protection to prevent damage in the event of an output overload. The MT3608 is available in a small 6-pin SOT-23 package.

TYPICAL APPLICATION

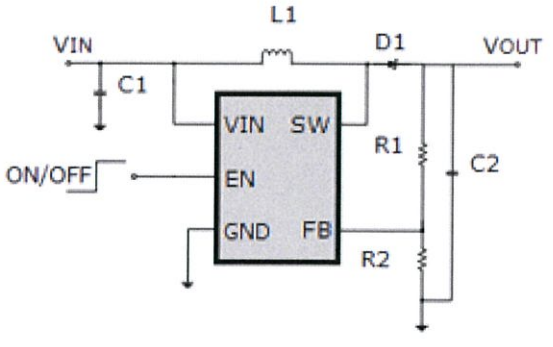


Figure 1. Basic Application Circuit

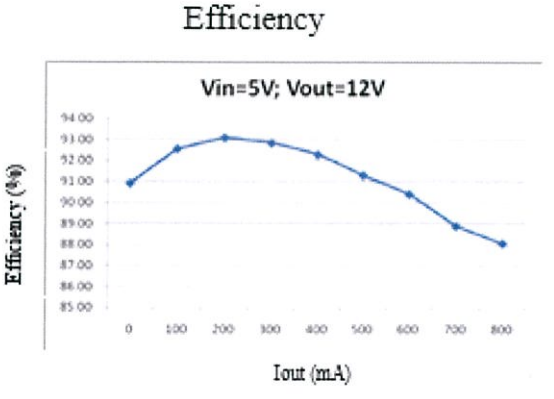


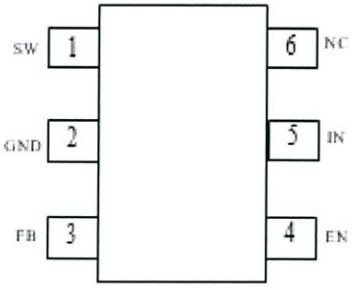
Figure 2. Efficiency Curve

ABSOLUTE MAXIMUM RATINGS

IN, EN voltages ..... -0.3V to 26V
Operating Temperature..... -40°C to +85°C FB Voltages.....-0.3V to 6V
Junction Temperature .....160°C

SW Voltage .....-0.3V to 30V
Storage Temperature Range -65°C to 150°C Peak SW Sink and Source Current .....4A Lead Temperature (Soldering, 10s)...+300°C

PACKAGE/ORDER INFORMATION



SOT23-6 (MT3608)

PIN DESCRIPTION

PIN	NAME	FUNCTION
1	SW	Power Switch Output. SW is the drain of the internal MOSFET switch. Connect the power inductor and output rectifier to SW. SW can swing between GND and 28V.
2	GND	Ground Pin
3	FB	Feedback Input. The FB voltage is 0.6V. Connect a resistor divider to FB.
4	EN	Regulator On/Off Control Input. A high input at EN turns on the converter, and a low input turns it off. When not used, connect EN to the input supply for automatic startup.
5	IN	Input Supply Pin. Must be locally bypassed.
6	NC	NC

ELECTRICAL CHARACTERISTICS

(V_{IN}=V_{EN}=5V, T_A = 25°C, unless otherwise noted.)

Parameter	Conditions	MIN	TYP	MAX	unit
Operating Input Voltage		2		24	V
Under Voltage Lockout				1.98	V
Under Voltage Lockout Hysteresis			100		mV
Current (Shutdown)	V_{EN}= 0V		0.1	1	μA
Quiescent Current (PFM)	V_{FB}=0.7V, No switch		100	200	μA
Quiescent Current (PWM)	V_{FB}=0.5V, switch		1.6	2.2	mA
Switching Frequency			1.2		MHz
Maximum Duty Cycle	V_{FB} = 0V	90			%
EN Input High Voltage		1.5			V
EN Input Low Voltage				0.4	V
FB Voltage		0.588	0.6	0.612	V
FB Input Bias Current	V_{FB} = 0.6V	-50	-10		nA
SW On Resistance (1)			80	150	mΩ
SW Current Limit (1)	V_{IN}= 5V, Duty cycle=50%		4		A
SW Leakage	V_{SW} = 20V			1	μA
Thermal Shutdown			155		°C

Note:

1) Guaranteed by design, not tested.

OPERATION

The MT3608 uses a fixed frequency, peak current mode boost regulator architecture to regulate voltage at the feedback pin. The operation ofthe MT3608 can be understood by referring to the block diagram of Figure 3. At the start of each oscillator cycle the MOSFET is turned on through the control circuitry. To prevent sub-harmonic oscillations at duty cycles greater than 50 percent, a stabilizing ramp is added to the output of the current sense amplifier and the result is fed into the negative input of the PWM comparator. When this voltage equals

The output voltage of the error amplifier the power MOSFET is turned off. The voltage at the output of the error amplifier is an amplified version of the difference between the 0.6V bandgap reference voltage and the feedback voltage. In this way the peak current level keeps the output in regulation. If the feedback voltage starts to drop, the output of the error amplifier increases. These results in more current to flow through the power MOSFET, thus increasing the power delivered to the output. The MT3608 has internal soft start to limit the amount of input current at startup and to also limit the amount of overshoot on the output.

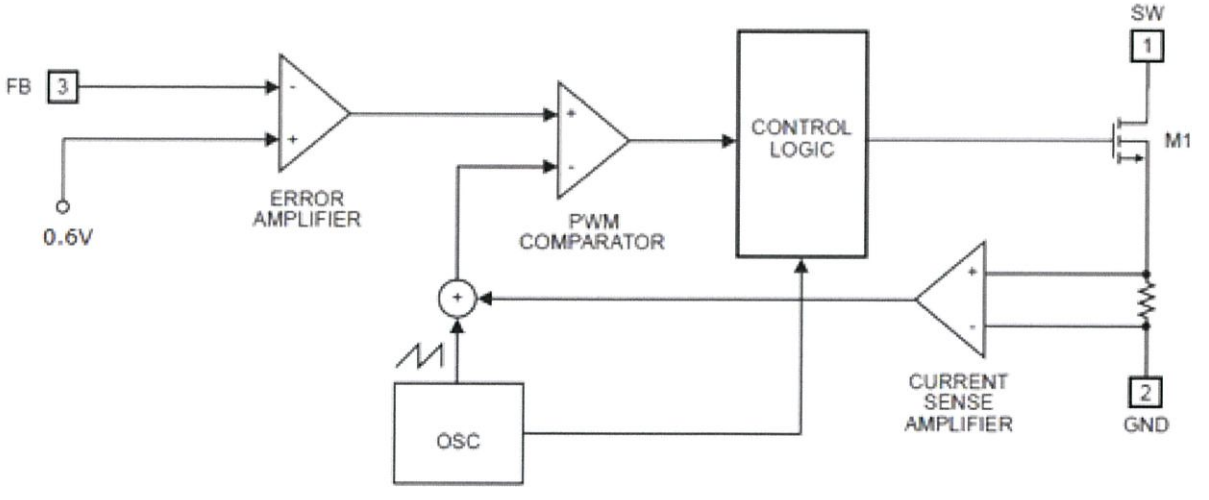
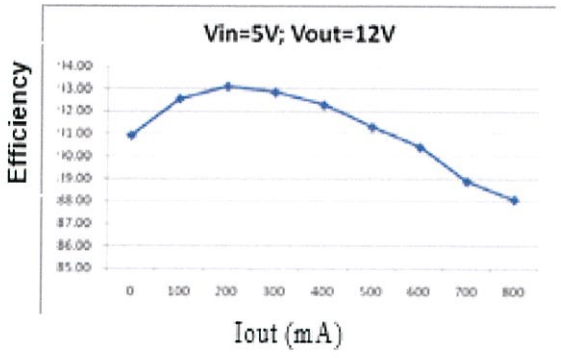


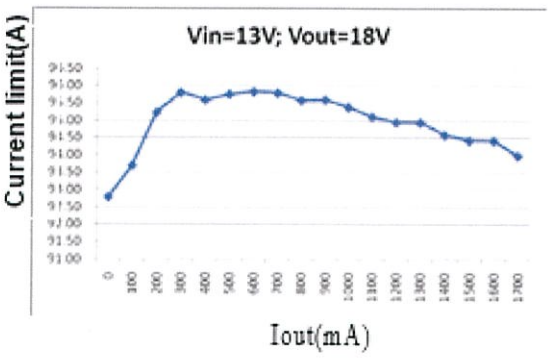
Figure 3. Functional Block Diagram

TYPICAL OPERATING CHARACTERISTICS

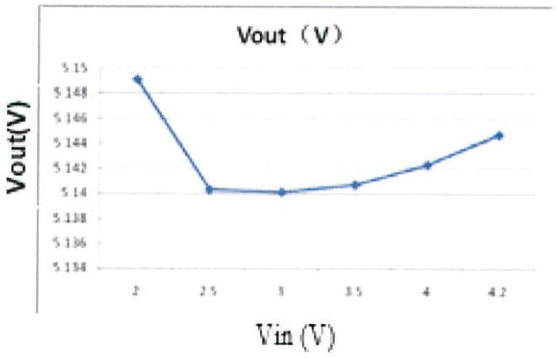
Efficiency Curve



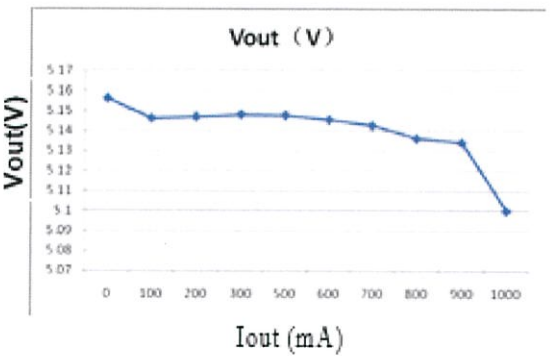
Efficiency Curve



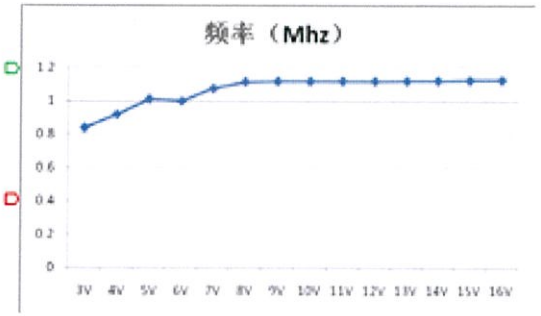
line Regulation



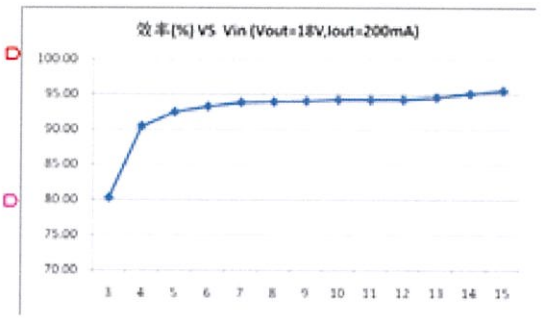
Load regualation



Freq VS Vin



Efficiency VS Vin



APPLICATION INFORMATION

Setting the Output Voltage

The internal reference VREF is 0.6V (Typical).The output voltage is divided by a resistor divider,R1 and R2 to the FB pin. The output voltage is given by

$$V_{OUT} = V_{REF} \times (1 + \frac{R_1}{R_2})$$

Inductor Selection

The recommended values of inductor are 4.7 to 22μH. Small size and better efficiency are the major concerns for portable device, such as MT3608 used for mobile phone. The inductor should have low core loss at 1.2MHz and low DCR for better efficiency. To avoid inductor saturation current rating should be considered.

Capacitor Selection

Input and output ceramic capacitors of 22μF are recommended for MT3608 applications. For better voltage filtering, ceramic capacitors with low ESR are recommended. X5R and X7R types are suitable because of their wider voltage and temperature ranges.

Diode Selection

Schottky diode is a good choice for MT3608 because of its low forward voltage drop and fast reverses recovery. Using Schottky diode can get better efficiency. The high speed rectification is also a good characteristic of Schottky diode for high switching frequency. Current rating of the diode must meet the root mean square of the peak current and output average current multiplication as following :

$$I_D(RMS) \approx \sqrt{I_{OUT} \times I_{PEAK}}$$

Layout Consideration

For best performance of the MT3608, the following guidelines must be strictly followed.

- Input and Output capacitors should be placed close to the IC and connected to ground plane to reduce noise coupling
- The GND should be connected to a strong ground plane for heat sinking and noise protection.
- Keep the main current traces as possible as short and wide.
- SW node of DC-DC converter is with high frequency voltage swing. It should be kept at a small area.
- Place the feedback components as close as possible to the IC and keep away from the noisy devices.

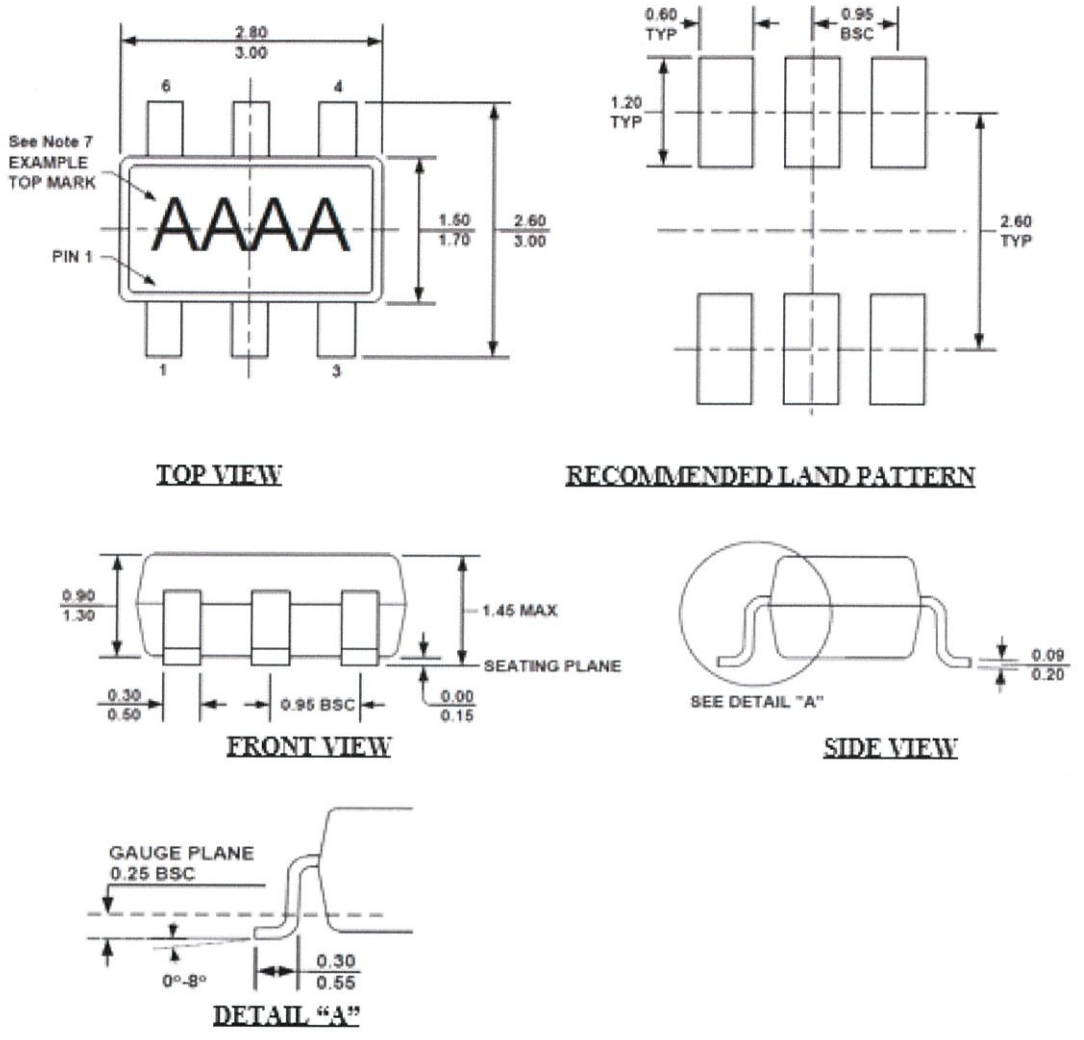


Figure 4. TSOT23-6/SOT23-6 Physical Dimensions

NOTE:

1) ALL DIMENSIONS ARE IN MILLIMETERS.
2) PACKAGE LENGTH DOES NOT INCLUDE MOLD FLASH, PROTRUSION OR GATE BURR.
3) PACKAGE WIDTH DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION.
4) LEAD COPLANARITY (BOTTOM OF LEADS AFTER FORMING) SHALL BE 0.10 MILLIMETERS MAX.
5) DRAWING CONFORMS TO JEDEC MO-193, VARIATION AB.
6) DRAWING IS NOT TO SCALE.
7) PIN 1 IS LOWER LEFT PIN WHEN READING TOP MARK FROM LEFT TO RIGHT, (SEE EXAMPLE TOP MARK)

Xi' an Aerosemi Technology Co., Ltd
Tel: 029-88868021 0755-82879616 021-51905952
Fax: 029-88445284 0755-82877171 021-51905952
Http ://www.aerosemi.com
E-mail: sales@aerosemi.com

Tech Support: services@elecfreaks.com

Ultrasonic Ranging Module HC - SR04

Product features:

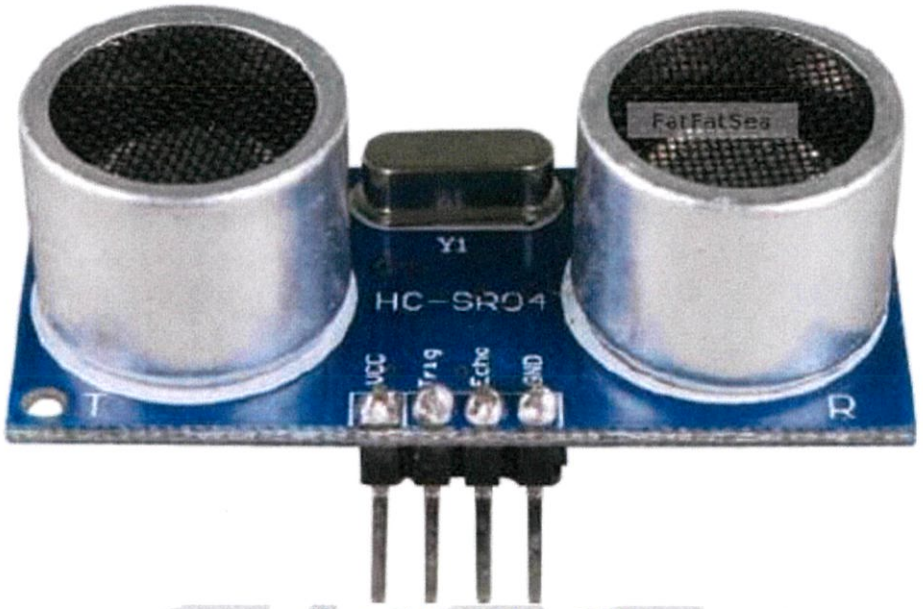
Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

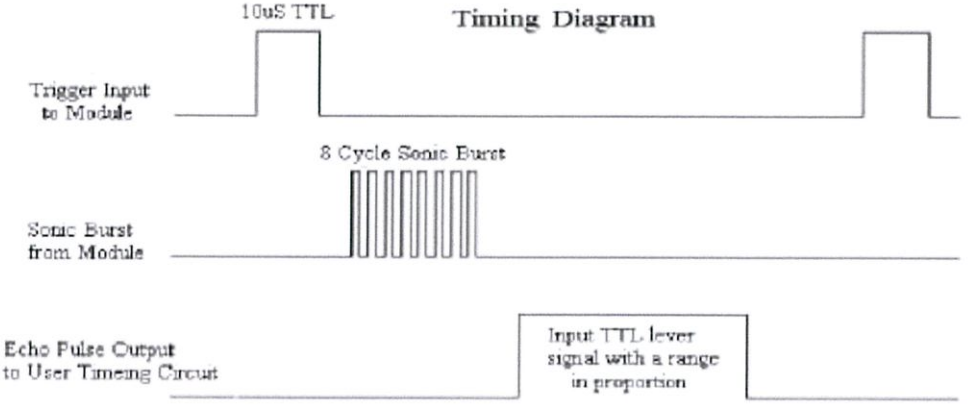
Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm



Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.Elecfreaks.com