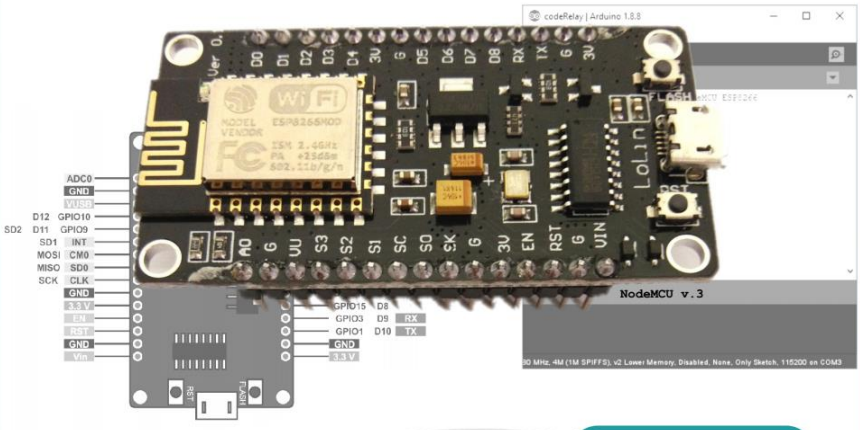
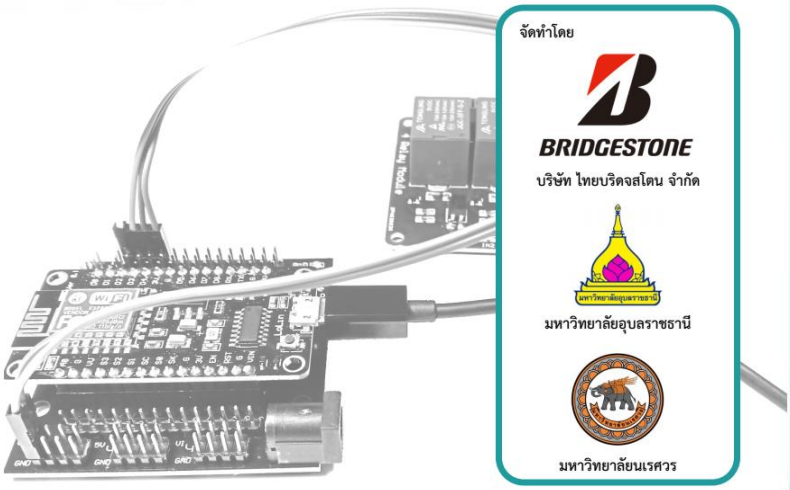
การใช้งานไมโครคอนโทรลเลอร์และเซนเซอร์ในการสร้างนวัตกรรม

Microcontroller and Sensors for Innovative Applications





จัดทำโดย

BRIDGESTONE

บริษัท ไทยบริดจสโตน จำกัด

มหาวิทยาลัยอุบลราชธานี

มหาวิทยาลัยนเรศวร

การใช้งานไมโครคอนโทรลเลอร์และเซนเซอร์ในการสร้างนวัตกรรม

Microcontroller and Sensors for Innovative Applications

ชื่อหนังสือ
การใช้งานไมโครคอนโทรลเลอร์และเซนเซอร์ในการสร้างนวัตกรรม

ผู้แต่ง/ผู้จัดทำ สุวิทย์ กิระวิทยา และคณะ

พิมพ์ครั้งที่ 1 พ.ศ. 2562

จำนวน 169 หน้า

ISBN 978-616-485-757-5

พิมพ์โดย บริษัท ไทยบริดจสโตน จำกัด
เลขที่ 990 อาคารอับดุลราฮิม ชั้น 16
ถ.พระราม 4 แขวงสีลม เขตบางรัก กรุงเทพฯ 10500

คำนำ

หนังสือฉบับนี้จัดทำขึ้นเพื่อใช้ประกอบการอบรมในโครงการสะพานสานฝันเยาวชนนักประดิษฐ์ปีที่ 12 หรือ ที่รู้จักกันในชื่อว่า Bridge 2 Inventor หรือ B2i โดยในการจัดอบรมซึ่งเป็นส่วนหนึ่งของโครงการนี้ ถูกเรียกว่า Bridge 2 Inventor Camp หรือค่ายสะพานสานฝันเยาวชนนักประดิษฐ์ ที่ปกติจะจัดในช่วงเดือนสิงหาคม ซึ่งในปีพ.ศ. 2562 นี้มีการจัดอบรมโดย 2 ศูนย์อบรมที่ 1 คือ ศูนย์มหาวิทยาลัยอุบลราชธานี ซึ่งจัดอบรมที่บ้านวิทยาศาสตร์สิรินธร สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (สวทช.) จ. ปทุมธานี และ ศูนย์อบรมที่ 2 ศูนย์มหาวิทยาลัยนเรศวร ซึ่งจัดอบรมที่ มหาวิทยาลัยนเรศวร จ.พิษณุโลก และมีหัวข้อการอบรมและการประกวดแข่งขัน คือ นวัตกรรมเพื่อความปลอดภัยในการเดินทาง (Innovation for Smart Mobility)

โครงการนี้เป็นความร่วมมืออันยาวนานระหว่างบริษัท ไทยบริดจสโตน จำกัด และมหาวิทยาลัยอุบลราชธานี โดยมีมหาวิทยาลัยนเรศวร มาร่วมดำเนินการด้วยตั้งแต่ปีพ.ศ. 2561 เป็นต้นมา การแข่งขัน B2i Challenge ซึ่งจัดขึ้นในช่วงเดือนพฤศจิกายน เป็นการประกวดนวัตกรรมจากฐานความรู้ที่ได้รับในระหว่างการอบรมและความคิดสร้างสรรค์ของผู้เข้าอบรม โครงการนี้ได้จัดต่อเนื่องมาเป็นปีที่ 12 แล้ว โดยในปีนี้เป็นครั้งแรกที่ทางคณะผู้ดำเนินงานได้จัดทำหนังสือฉบับนี้ขึ้นมา โดยทีมงานผู้จัดทำหนังสือ ได้กำหนดให้หนังสือมีเนื้อหาที่ครอบคลุมเนื้อหาที่ใช้อบรมในโครงการนี้ และพยายามทำให้หนังสือมีเนื้อหาที่เหมาะสมสำหรับการศึกษาด้วยตนเองของผู้ที่สนใจ โดยคณะผู้จัดทำมีวัตถุประสงค์การจัดทำหนังสือนี้ เพื่อเผยแพร่ความรู้ด้านเทคโนโลยีด้านระบบสมองกลฝังตัว (Embedded System) ซึ่งเป็นพื้นฐานในโครงสร้างเทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things, IoT) ผู้เข้าร่วมการอบรมในปีนี้ได้แก่ ครู นักเรียน นิสิต และนักศึกษาที่มีความสนใจในเทคโนโลยีดังกล่าว โดยเมื่อผู้ที่ศึกษาได้ทำการศึกษา ทดลองและทำความเข้าใจการทำงานของอุปกรณ์ต่าง ๆ ดีแล้ว ก็จะสามารถนำความรู้ที่ได้รับไปประยุกต์ใช้งานต่าง ๆ ได้อย่างหลากหลาย โดยเฉพาะอย่างยิ่ง ทางด้านการสร้างนวัตกรรมที่ใช้ระบบอิเล็กทรอนิกส์สมองกลฝังตัวเป็นฐาน

หนังสือเล่มนี้แบ่งออกเป็น 4 ส่วนหลัก ๆ โดยมีลำดับตามเนื้อหาที่อบรมในโครงการ ได้แก่ บทที่ 1 ซึ่งเป็นการบรรยายพื้นฐานเกี่ยวกับแนวคิดการสร้างนวัตกรรมด้วยระบบอิเล็กทรอนิกส์สมองกลฝังตัว บทที่ 2 กล่าวถึงพื้นฐานภาษา C/C++ สำหรับงานด้านระบบสมองกลฝังตัว บทที่ 3-11 (9 บท) เป็นการแสดงการใช้งานบอร์ดไมโครคอนโทรลเลอร์หลัก คือ NodeMCU v.3 ร่วมกับบอร์ดต่าง ๆ ที่นำมาประกอบกันพร้อมด้วยตัวอย่างโค้ดสำหรับการทดลอง และบทที่ 12-13 เป็นการใช้โทรศัพท์มือถือในการเชื่อมต่อและควบคุมการทำงานของระบบผ่าน Blynk App

คณะผู้จัดงานขอขอบคุณผู้ที่ให้การอนุเคราะห์และสนับสนุนในด้านต่าง ๆ ในการจัดทำหนังสือเล่มนี้และการดำเนินการอบรมในปีนี้ โดยคณะผู้จัดงานหวังว่าเอกสารฉบับนี้จะเป็นประโยชน์ต่อผู้เข้าร่วมอบรมในโครงการนี้ รวมถึงผู้ที่สนใจ ซึ่งหากมีข้อผิดพลาดประการใดต้องขออภัยมา ณ ที่นี้ด้วย

คณะผู้จัดทำ

บริษัท ไทยบริดจสโตน จำกัด

มหาวิทยาลัยอุบลราชธานี

มหาวิทยาลัยนเรศวร

สารบาญ

บทที่ 1

ภาพรวมของการสร้างนวัตกรรมด้วยระบบสมองกลฝังตัว

1.1 นวัตกรรมคืออะไร

นวัตกรรม (Innovation) มีความหมายในทำนองว่า ความคิดใหม่ ๆ การคิดสร้างสรรค์ การจินตนาการถึงสิ่งใหม่ ๆ ในรูปแบบของอุปกรณ์หรือวิธีการ นวัตกรรมนี้ถูกมองเสมือนกับการประยุกต์ใช้ของบางสิ่งบางอย่าง ที่เป็นคำตอบของความต้องการใหม่ ๆ หรือความต้องการที่ไม่ชัดเจน การสร้างนวัตกรรมเริ่มจากการเล็งเห็นถึงการได้ประโยชน์จากการสรรสร้างสิ่งใหม่ ๆ ที่ทำให้ระบบเดิมมีประสิทธิภาพดียิ่งขึ้นจากเทคโนโลยีที่มีในปัจจุบัน นวัตกรรมหนึ่ง ๆ ต้องมีความแปลกใหม่ (Originality) และมีประสิทธิภาพ ทำให้สังคมหรือตลาดยอมรับและนำไปใช้ต่อ โดยคำว่า นวัตกรรม มักถูกนำไปเชื่อมโยงกับคำว่า การประดิษฐ์ (Invention) โดยทั้งสองคำนี้มีความหมายที่สัมพันธ์กัน แต่คำว่านวัตกรรมนั้นมักจะหมายถึงสิ่งที่นำไปใช้ได้จริง มีผลกระทบต่อเศรษฐกิจและสังคม มากกว่าการประดิษฐ์สิ่งประดิษฐ์เท่านั้น โดยนวัตกรรมนี้มักจะสรรสร้างจากกระบวนการทางวิศวกรรมซึ่งเป็นกระบวนการที่ใช้แก้ปัญหาทางเทคนิคโดยอาศัยความรู้ทางวิทยาศาสตร์เป็นสำคัญ

นวัตกรรมหนึ่ง ๆ อาจถูกมองได้สองด้าน คือ (1) ระดับความแปลกใหม่ (Novelty) คือแปลกใหม่ในระดับท้องถิ่น แปลกใหม่ในระดับตลาด แปลกใหม่ในระดับอุตสาหกรรม หรือแปลกใหม่ในโลก และ (2) ประเภทของความแปลกใหม่ คือในด้านสิ่งของหรือด้านกระบวนการ โดยปัจจุบันเราสามารถมองแบ่งแยกได้อย่างชัดเจนระหว่างสิ่งที่เป็นนวัตกรรมและความคิดสร้างสรรค์ ในหนังสือเล่มนี้ จะกล่าวถึงความรู้พื้นฐานที่สามารถนำไปใช้ต่อยอดในการสร้างนวัตกรรมได้

1.2 ระบบสมองกลฝังตัว

เทคโนโลยีอิเล็กทรอนิกส์ถือเป็นเทคโนโลยีที่มีการพัฒนาอย่างมากในช่วงหลายทศวรรษที่ผ่านมา โดยระบบสมองกลฝังตัว (Embedded System) คือระบบประมวลผลที่ใช้ไมโครโพรเซสเซอร์ (Microprocessor) หรือไมโครคอนโทรลเลอร์ (Microcontroller) ที่ออกแบบมาเฉพาะสำหรับอุปกรณ์ที่ผู้ประดิษฐ์สรรสร้างขึ้น โดยการใช้ระบบสมองกลฝังตัวนี้จะทำให้อุปกรณ์ที่สร้างขึ้นมีความฉลาดหรือ สมาร์ท (smart) ขึ้นเมื่อเทียบกับอุปกรณ์ที่ไม่มีระบบนี้ ปัจจุบันระบบสมองกลฝังตัวถูกนำมาใช้กันอย่างแพร่หลาย ทั้งในยานพาหนะ เครื่องใช้ไฟฟ้าในบ้านและสำนักงาน อุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ ทั้งนี้เนื่องจากทรัพยากร (ความเร็วและหน่วยความจำ) ของชิปประมวลผลมีประสิทธิภาพเพียงพอสำหรับการใช้งานประเภทนี้

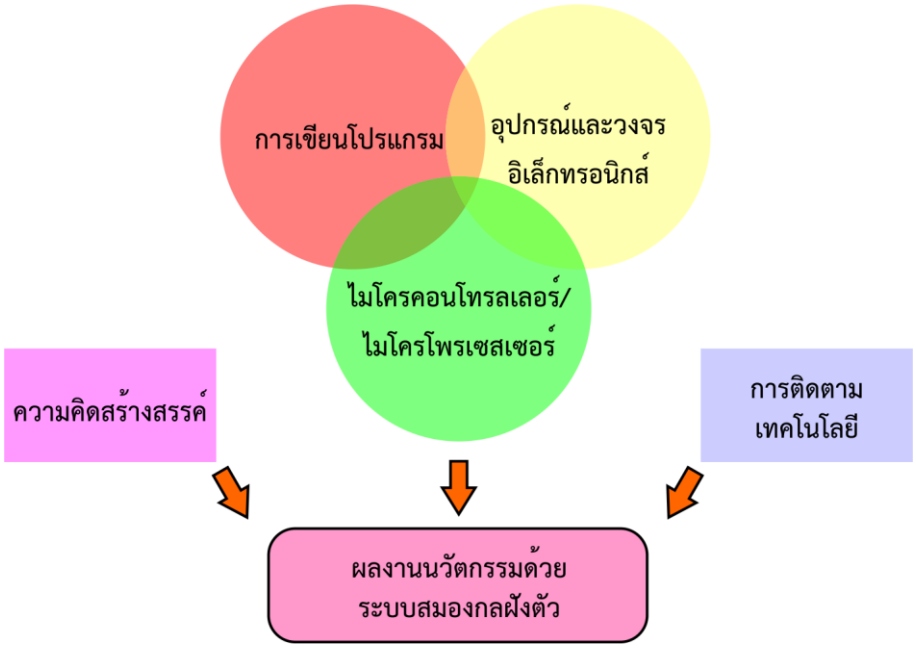
ปัจจุบันการสร้างนวัตกรรมจากสิ่งประดิษฐ์อิเล็กทรอนิกส์จึงเป็นสิ่งที่เกิดขึ้นและพบเห็นได้บ่อย ๆ ตัวอย่างเช่นเทคโนโลยีหุ่นยนต์ (รูปที่ 1.1 (ก)) และเทคโนโลยีอากาศยานไร้คนขับหรือโดรน (รูปที่ 1.1(ข)) ที่มีให้เห็นกันอย่างแพร่หลาย และเริ่มเข้ามาในชีวิตประจำวัน

(ก) (ข)

รูปที่ 1.1 ตัวอย่างนวัตกรรมในเทคโนโลยีด้าน (ก) หุ่นยนต์ และ (ข) อากาศยานไร้คนขับ (ภาพ (ก) และ (ข) จาก https://www.ecnmag.com/news/2019/01/top-10-robotic-innovations-2018 และ https://airdronecraze.com/drone-trends/ ตามลำดับ)

มากขึ้น โดยเบื้องหลังของการสร้างนวัตกรรมด้วยเทคโนโลยีเหล่านั้นส่วนมากจะมีการนำระบบสมองกลฝังตัวไปใช้ เพื่อทำให้ระบบมีความแปลกใหม่ สามารถทำในสิ่งที่อาจไม่เคยทำได้มาก่อน

การศึกษาในทางวิศวกรรมเพื่อให้มีความรู้ ความสามารถ ในการนำไปพัฒนาระบบสมองกลฝังตัวขึ้นมาได้นั้น จะต้องอาศัยความรู้ในหลายสาขาวิชา โดยอาจแบ่งเนื้อหาวิชาที่จำเป็นต้องใช้เป็น 3 ส่วน คือ วิชาการเขียนโปรแกรมคอมพิวเตอร์ (Computer Programming) วิชาด้านอุปกรณ์และวงจรอิเล็กทรอนิกส์ (Electronic Devices and Circuits) และ วิชาด้านไมโครโพรเซสเซอร์หรือไมโครคอนโทรลเลอร์ (Microprocessor/ Microcontroller) โดยนอกจากความรู้ในวิชาเหล่านี้แล้ว ผู้ที่ต้องการพัฒนาระบบสมองกลฝังตัวขึ้นมาเป็นนวัตกรรมนั้นจะต้องมีทั้งความคิดสร้างสรรค์และการติดตามเทคโนโลยที่สนใจอย่างสม่ำเสมอด้วย



รูปที่ 1.2 องค์ประกอบของการสร้างสรรค์นวัตกรรมด้วยระบบสมองกลฝังตัว ที่ประกอบด้วยความรู้จากวิชาต่าง ๆ ความคิดสร้างสรรค์และการติดตามเทคโนโลยี

1.3 เทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง

เทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things, IoT) เป็นเทคโนโลยีที่อาศัยฐานความรู้เกี่ยวกับระบบสมองกลฝังตัว และเป็นเทคโนโลยีที่กำลังได้รับความสนใจอย่างมากในปัจจุบัน โดยแนวคิดเรื่องการเชื่อมต่อสิ่งต่าง ๆ นี้จะมีมาตั้งแต่ยุคต้นปี 1980 แต่ยังไม่มีการใช้งานอย่างแพร่หลาย เนื่องจากความสามารถในการเชื่อมต่อเป็นเครือข่ายของอุปกรณ์อิเล็กทรอนิกส์ในขณะนั้นยังไม่เพียงพอ ต่างกับในปัจจุบัน โดยปัจจุบัน เราสามารถทำให้อุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ ปฏิสัมพันธ์และ/หรือแลกเปลี่ยนข้อมูลระหว่างกันได้โดยง่าย และเรียกว่าอุปกรณ์เหล่านี้ว่า อุปกรณ์ IoT

องค์ประกอบพื้นฐานของสิ่งที่เรียกว่าอุปกรณ์ IoT อาจแบ่งเป็น 2 ส่วน คือ ฮาร์ดแวร์ (hardware) และ ซอฟต์แวร์ (software) โดยนวัตกรรมทั่วไปนั้นอาจจะมีเพียงส่วนเดียวคือฮาร์ดแวร์หรือซอฟต์แวร์เท่านั้นก็ได้ แต่สำหรับนวัตกรรมที่สร้างด้วยระบบสมองกลฝังตัว ก็คงจะขาดส่วนใดส่วนหนึ่งไม่ได้ โดยสำหรับอุปกรณ์ IoT ทั้งฮาร์ดแวร์และซอฟต์แวร์จะต้องมีส่วนที่เกี่ยวข้องกับระบบเครือข่ายที่ใช้เชื่อมต่อข้อมูลต่าง ๆ อีกด้วย

ปัจจุบันอุปกรณ์ IoT แบ่งได้เป็นหลายประเภท โดยมีประเภทที่ได้รับความสนใจมาก (เรียงตามลำดับ โดยอ้างอิงจาก iot-analytics.com) คือ

บ้านอัจฉริยะ (Smart Home) หรือบ้านอัตโนมัติ (Home Automation) ซึ่งเป็นกลุ่มอุปกรณ์ที่เชื่อมต่อของใช้ในบ้าน เช่น เครื่องวัดอุณหภูมิ เครื่องตรวจสอบควัน ไฟส่องสว่าง เครื่องใช้ไฟฟ้า เครื่องเล่นเกมส์ เครื่องเสียง กลอนประตู/หน้าต่าง

อุปกรณ์สวมใส่ (Wearables) โดยอุปกรณ์จำพวกนาฬิกาอัจฉริยะ (Smart Watch) ถือเป็นกลุ่มอุปกรณ์ที่กำลังได้รับความนิยมอย่างมากในปัจจุบัน

เมืองอัจฉริยะ (Smart City) ถือเป็นกลุ่มที่มีการใช้อุปกรณ์ IoT เป็นวงกว้าง มีทั้ง ระบบไฟจราจรอัจฉริยะ ระบบบริหารการแจกจ่ายน้ำ ระบบความปลอดภัย ระบบสังเกตการณ์สิ่งแวดล้อมต่าง ๆ โดยการใช้เทคโนโลยี IoT ในกลุ่มนี้ มีจุดประสงค์หลักเพื่อพัฒนาการใช้ชีวิต

ในเมืองให้สะดวกมากยิ่งขึ้น และ/หรือ แก้ปัญหาต่าง ๆ ที่พบเห็นได้ทั่วไปในเมือง (เช่นปัญหาจราจร อาชญากรรมและมลพิษ)

สมาร์ทกริด (Smart Grid) เป็นการนำเทคโนโลยี IoT มาใช้ส่งข้อมูลต่าง ๆ เพื่อที่จะให้ระบบการจ่ายไฟฟ้ามีประสิทธิภาพ มีเสถียรภาพ และประหยัดมากยิ่งขึ้น

อินเทอร์เน็ตอุตสาหกรรม (Industrial Internet) คือระบบการผลิตอัจฉริยะหรือโรงงานอันฉริยะ ที่มีการนำเอาอุปกรณ์ IoT มาติดตั้ง เพื่อที่จะทำให้ระบบการผลิตมีประสิทธิภาพมากยิ่งขึ้น

ยานยนต์ที่เชื่อมต่อ (Connected Car) สำหรับยานยนต์แห่งอนาคต คงยังไม่มีใครตอบได้ชัดเจนว่าจะออกมาในทิศทางใด แต่ไม่ว่าจะเป็นยานยนต์ไร้คนขับ, ยานยนต์ไฟฟ้า หรือยานยนต์กึ่งอัตโนมัติ แนวคิดการเชื่อมต่อยานยนต์ต่าง ๆ การใช้ระบบแผนที่นำทางและระบบไฟจราจร น่าจะเป็นแนวคิดหนึ่งที่จะถูกนำมาใช้กับยานยนต์ทุกประเภท ซึ่งอุปกรณ์เหล่านั้นล้วนต้องอาศัยเทคโนโลยี IoT

ระบบสุขภาพที่เชื่อมต่อ (Connected Health) หรือในชื่อระบบสุขภาพดิจิทัล (Digital Health) ระบบสุขภาพทางไกล (Telehealth/Telemedicine) เป็นแนวคิดที่จะนำระบบการดูแลสุขภาพมาเชื่อมต่อกันและนำอุปกรณ์อัจฉริยะทางด้านการแพทย์มาใช้ ด้วยเทคโนโลยีที่มีในปัจจุบัน การตรวจสุขภาพทางไกลแบบเวลาจริงสามารถทำได้จริงและระบบช่วยการตัดสินใจในการวินิจฉัยโรคได้พัฒนาขึ้นไปมาก

นอกจากเทคโนโลยีที่กล่าวมาข้างต้น แล้วยังมีเทคโนโลยี การค้าปลีกอัจฉริยะ (Smart Retail) โซ่อุปทานอัจฉริยะ (Smart Supply Chain) และ ฟาร์มอัจฉริยะ (Smart Farming) ซึ่งกำลังได้รับความนิยมสนใจเพิ่มขึ้นในปัจจุบัน

บทที่ 2

พื้นฐานภาษา C/C++

ในบทนี้ เราจะขอกล่าวถึงความรู้พื้นฐานเกี่ยวกับภาษา C/C++ ที่จำเป็นในการเขียนโปรแกรมเพื่อให้สามารถใช้งานอุปกรณ์อิเล็กทรอนิกส์และเซนเซอร์ต่าง ๆ ผ่านระบบไมโครคอนโทรลเลอร์

2.1 ตัวแปรในภาษา C/C++ และคำสั่งพื้นฐาน

ในภาษา C/C++ เราจะต้องประกาศชื่อและชนิดตัวแปรก่อนจะนำไปใช้เก็บข้อมูลหรือประมวลผล โดยตัวแปรในภาษา C/C++ สามารถแบ่งได้เป็น 4 ชนิดหลัก ๆ ได้แก่ ตัวอักษร (character) ตัวเลขจำนวนเต็ม (integer) ตัวเลขทศนิยม (floating point number) และค่าความจริง (true/false) โดยอาจแบ่งย่อยลงไปอีก ขึ้นกับการกำหนดในแต่ละระบบปฏิบัติการ (16, 32 หรือ 64 บิต) ตารางที่ 2.1 แสดงชนิดของตัวแปรในภาษา C/C++ โดยตัวแปรพื้นฐานที่ใช้งานได้แก่ ตัวอักษร, ตัวเลขจำนวนเต็มและตัวเลขทศนิยม โดยคำสำคัญที่ใช้ประกาศตัวแปรทั้ง 3 คือ char, int และ float ตามลำดับ ผู้ศึกษาควรจำคำเหล่านี้เพื่อใช้ในการเขียนโปรแกรม

ตัวอย่างการตั้งชื่อตัวแปร	ตัวอย่างการตั้งชื่อตัวแปรที่ผิด
int Relay1; char buff; float volt_LDR; long distance;	int 1relay; ผิด ชื่อตัวแปรขึ้นต้นด้วยตัวเลขไม่ได้ char bool; ผิดเพราะว่า bool เป็นคำสงวน float volt LDR; ผิดเพราะชื่อตัวแปรมีเว้นวรรคไม่ได้

ตารางที่ 2.1 ชนิดตัวแปรในภาษา C/C++

ชนิดตัวแปร	คำสั่งสำหรับการประกาศ	ขนาด (บิต)	ช่วงของข้อมูล
ตัวอักษร*	char	8	-128 ถึง 127
ตัวอักษร	unsigned char	8	0 ถึง 255
จำนวนเต็ม	int	16	-32768 ถึง 32767
จำนวนเต็ม	unsigned int	16	0 ถึง 65535
จำนวนเต็ม	long	32	-2^{31} ถึง $2^{31}-1$
เลขทศนิยม	float	32	-3.4E+38 ถึง 3.4E+38**
เลขทศนิยม	double	64	-1.8E+308 ถึง 1.8E+308
ค่าความจริง	bool	8	True (= 1) หรือ False (= 0)

* เก็บด้วยรหัสแอสกี (ASCII Code)

**เราใช้ E แทนการเขียนด้วยสิบยกกำลัง (เช่น 3E+2 = 3×10^2)

สำหรับในภาษา C/C++ นั้นชื่อตัวแปรที่มีตัวพิมพ์เล็ก-ใหญ่ต่างกัน ถือเป็นตัวแปรคนละตัว (case sensitive) เช่นตัวแปรชื่อ relay เป็นคนละตัวแปรกับ Relay หรือ ReLay หรือ RELAY

โดยทุกคำสั่งในในภาษา C/C++ จะต้องลงท้ายด้วยเครื่องหมายเซมิโคลอน ; (หรือเครื่องหมายอัฒภาค) เพื่อบอกให้ภาษา C/C++ ทราบว่าเป็นการสิ้นสุดคำสั่งนั้น

เราสามารถใส่ คำหรือข้อความที่เราไม่ต้องการให้โปรแกรมประมวล แต่เขียนเพื่อบันทึกช่วยในการอ่านโค้ดนั้นได้ โดยทำได้ 2 วิธี คือ

1. ใส่เครื่องหมาย /* */ ครอบข้อความที่เราเขียน เช่น

```
/*  ข้อความนี้ไม่ถูกประมวลผลด้วยโปรแกรม  */
```

2. ใส่เครื่องหมาย // ไว้ด้านหน้าของข้อความ ทำให้โปรแกรมไม่ประมวลข้อความที่ตามมาทั้งบรรทัด เช่น

```
int x = 2;                    // ประกาศตัวแปร x ให้มีค่าเท่ากับ 2
```

สำหรับสตริง (String) ซึ่งเป็นข้อมูลที่สร้างได้จากอาเรย์ของตัวอักษร จะสามารถสร้างได้จากการประกาศอาเรย์ของตัวอักษร โดยในการใช้งานเบื้องต้นนั้น เราจะกำหนด ตัวอักษรที่อยู่ใน เครื่องหมายอัญประกาศเดี่ยว ‘ ’ เป็นชนิดตัวอักษรตัวเดียว (character) และตัวอักษรในเครื่องหมายอัญประกาศ “ ” เป็นข้อมูลชนิดสตริง เช่น “ABC”, “Hello”

สำหรับตัวอักษรพิเศษต่าง ๆ เช่น การกั้นหน้า (tab, \t) ที่มีการกำหนดในรหัสแอสกีมาตรฐานและไม่สามารถเขียนแทนด้วยตัวอักษรภาษาอังกฤษได้ เราก็จะใช้ เครื่องหมาย \ นำหน้า เพื่อใช้แจ้งให้โปรแกรมภาษา C/C++ ทราบ ‘\n’ คือ ตัวอักษรขึ้นบรรทัดใหม่ (newline character) และเมื่อโปรแกรมต้องแสดงผลตัวอักษรนี้ ก็จะแสดงโดยการเลื่อนเคอ-เซอร์ไปขึ้นบรรทัดใหม่

ในการเขียนภาษา C/C++ คำสั่งพื้นฐานจะมีลักษณะเป็นฟังก์ชัน โดยฟังก์ชันที่ถูกเรียกใช้บ่อย ๆ คือฟังก์ชันหรือคำสั่ง printf() ซึ่งฟังก์ชันนี้จะใช้เพื่อให้โปรแกรมพิมพ์แสดงข้อความออกมาทางจอภาพ (ฟังก์ชันนี้ถูกกำหนดไว้ในไลบรารีมาตรฐานของภาษา C คือใน stdio.h) รูปแบบการใช้งานคือ

printf(สตริง);

เช่นคำสั่ง printf(“Hello World.”); จะแสดงข้อความ Hello World. บนจอ โดยไม่แสดงเครื่องหมายคำพูดที่เป็นตัวกำหนดชนิดข้อมูล ซึ่งสตริงนี้อาจกำหนดให้มีข้อมูลที่เป็นตัวแปรและแทนที่ด้วยค่าตัวแปรในขณะแสดงผลได้ เช่นคำสั่ง printf(“value of x = %d”, x); จะแสดงข้อความ value x = ค่าตัวเลขจำนวนเต็มที่เก็บอยู่ในตัวแปรชื่อ x โดย %d หมายถึงการแสดงค่าตัวแปรที่ตามมานั้นเป็นค่าในเลขฐานสิบ (decimal value)

การดำเนินการในการเขียนโปรแกรมพื้นฐานคือการกำหนดค่า (assignment operator) โดยใช้เครื่องหมายเท่ากับ (=) นั่นคือ

ตัวแปร = ผลของการดำเนินการ

โดยเครื่องหมายเท่ากับนี้จะต่างกับเครื่องหมายเท่ากับในทางคณิตศาสตร์ เพราะ ในการเขียนโปรแกรม เครื่องหมายนี้จะหมายถึงการนำค่าที่ได้จากผลการดำเนิน (หรือผลลัพธ์จากการเรียกฟังก์ชัน) การลงไปเก็บในตัวแปรที่อยู่ด้านซ้านมือเท่านั้น

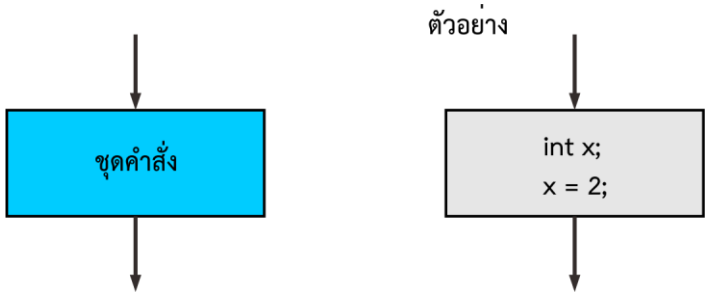
สำหรับการดำเนินการพื้นฐาน แบ่งได้เป็น 4 กลุ่ม ซึ่งสรุปไว้ในตารางที่ 2.2 สำหรับตัวดำเนินการที่มิใช่ตัวดำเนินการพื้นฐาน เช่น การหาค่ารากที่สอง การหาค่าจากการคำนวณเลขยกกำลัง x^y ภาษา C/C++ ก็สามารถดำเนินการได้โดยเรียกใช้คำสั่งหรือฟังก์ชันเฉพาะ เช่น sqrt(), pow() โดยอาจจะต้องเรียกใช้ไลบรารีเพิ่มเติมในตอนต้นของโปรแกรม โดยใช้คำสั่ง #include เช่น #include <math.h> เป็นการเพิ่มไลบรารีที่มีชุดคำสั่งในการคำนวณทางคณิตศาสตร์เพิ่มเติมเข้าไปในโปรแกรม โดยคำสั่งที่เพิ่ม จะมีแสดงอยู่ในไฟล์ math.h

ตารางที่ 2.2 ตัวดำเนินการในภาษา C/C++

กลุ่มตัวดำเนินการ	ตัวดำเนินการ	คำอธิบาย
ตัวดำเนินการคำนวณ	+ (บวก), – (ลบ) * (คูณ), และ / (หาร)	เครื่องหมาย * และ / จะถูกกระทำก่อนเครื่องหมาย + และ -
ตัวดำเนินการเชิงความสัมพันธ์ (relational operator)	> (มากกว่า), < (น้อยกว่า) >= (มากกว่าหรือเท่ากับ), <= (น้อยกว่าหรือเท่ากับ), == (เท่ากับ) และ != (ไม่เท่ากับ)	ผลลัพธ์จากตัวดำเนินการนี้จะเป็นค่าความจริง (จริงหรือเท็จ) เท่านั้น เช่น 2 => 1 ให้ค่าความจริงเป็นจริง และ 1 != 7/7 ให้ค่าความจริงเป็นเท็จ
ตัวดำเนินการทางตรรก (logical operator)	&& (และ) || (หรือ) และ ! (นิเสธ)	ใช้ในการกำหนดเงื่อนไขการทำงานของโปรแกรมโดยดำเนินการกับผลลัพธ์ที่ได้จากตัวดำเนินการความสัมพันธ์
ตัวดำเนินการประกอบ (compound operator)	++ (เพิ่มค่าอีกหนึ่งแล้วแทนที่) -- (ลดค่าลงหนึ่งแล้วแทนที่) += (เพิ่มค่าแล้วแทนที่) -= (ลดค่าแล้วแทนที่)	เช่น i++ หมายถึง i = i +1; j += 2 หมายถึง j = j + 2; โดยตัวดำเนินการนี้ทำให้สามารถเขียนโปรแกรมได้กระชับขึ้น

2.2 คำสั่งควบคุม

ในการเขียนโปรแกรมภาษา C/C++ เพื่อควบคุมการทำงานของคอมพิวเตอร์นั้น เราจะต้องกำหนดการดำเนินการคือการเขียนชุดคำสั่งให้โปรแกรมทำงานเป็นลำดับ ในกรณีอย่างง่ายที่สุดคือการทำงานตามลำดับดังแสดงเป็นแผนผังการทำงานได้ดังรูปที่ 2.1



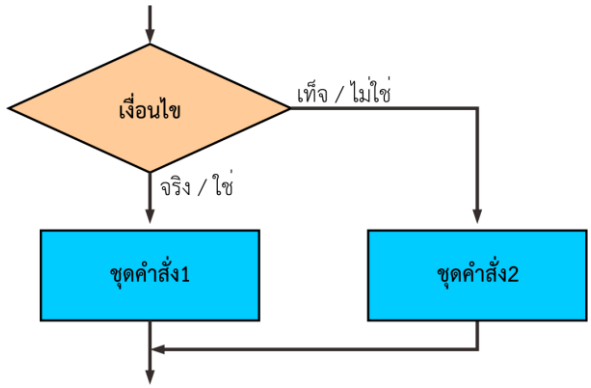
รูปที่ 2.1 แผนผังการทำงานตามลำดับและ
ตัวอย่างคำสั่งประกาศตัวแปร x และกำหนดค่า x = 2

ชุดคำสั่งควบคุมที่จะเป็นตัวกำหนดลำดับการทำงานแบ่งได้เป็น 2 ประเภท คือ แบบเลือกทางใดทางหนึ่งโดยใช้คำสั่ง if-else และ แบบวนลูปโดยการตรวจสอบเงื่อนไข คือคำสั่ง for โดยคำสั่งควบคุมทั้งสองประเภทจะใช้ผลลัพธ์จากการดำเนินการเชิงความสัมพันธ์ เป็นตัวตัดสินใจ

- ชุดคำสั่ง if-else มีรูปแบบคือ

```
if (เงื่อนไข) {
        ชุดคำสั่ง1
} else {
        ชุดคำสั่ง2
}
```

ซึ่งเงื่อนไขจะให้ผลลัพธ์ออกมาเป็นจริง (True) หรือเท็จ (False) ได้เท่านั้น โดยเมื่อเงื่อนไขเป็นจริง โปรแกรมก็จะทำชุดคำสั่ง1 และหากเป็นเท็จโปรแกรมก็จะทำชุดคำสั่ง2 โดยชุดคำสั่ง if-else นี้ สามารถเขียนแสดงได้ในรูปแผนผังการทำงาน ดังรูปที่ 2.2

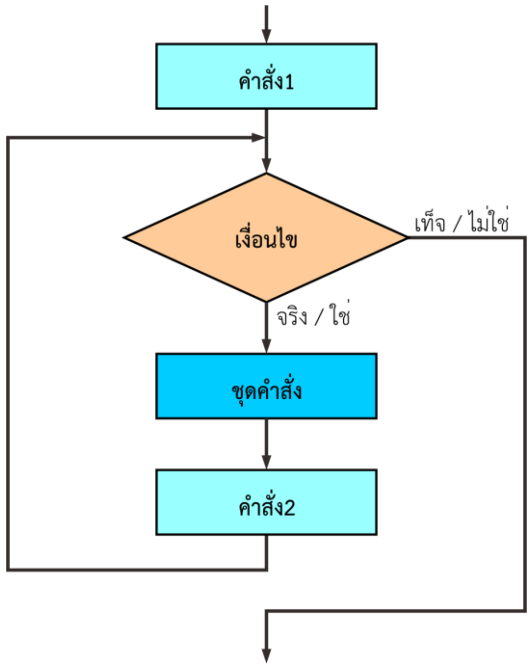


รูปที่ 2.2 แผนผังการทำงานของชุดคำสั่ง if-else

- ชุดคำสั่ง for เป็นชุดคำสั่งที่ใช้ในการวนลูป มีรูปแบบคือ

```
for (คำสั่ง1; เงื่อนไข; คำสั่ง2) {
        ชุดคำสั่ง
}
```

ในการทำงานของคำสั่ง for นี้ ก็จะเริ่มจากการทำคำสั่ง1 แล้วตรวจสอบเงื่อนไข จากนั้นจึงทำชุดคำสั่งภายในลูป 1 ครั้ง จึงทำคำสั่ง2 ก่อนตรวจสอบเงื่อนไข เพื่อดำเนินการในรอบถัดไป โดยหากเงื่อนไขที่ตรวจสอบให้ผลลัพธ์เป็นเท็จ โปรแกรมก็จะออกจากคำสั่ง for นี้แล้วดำเนินการในลำดับถัดไป ชุดคำสั่ง for นี้ สามารถเขียนแสดงได้ในรูปแผนผังการทำงาน ดังรูปที่ 2.3



รูปที่ 2.3 แผนผังการทำงานของชุดคำสั่ง for

ตัวอย่างการใช้งานชุดคำสั่ง for ที่พบเห็นได้บ่อยคือ

for (int i=0; i < 10; i++) { ชุดคำสั่ง }

ซึ่งเป็นการ กำหนดตัวแปรชื่อ i เป็นตัวแปรที่ใช้ในการวนลูป โดยเริ่มต้นที่ i = 0 แล้วจากนั้นให้ตรวจสอบว่า i มีค่าน้อยกว่า 10 หรือไม่ หากใช่ ก็จะทำงานในชุดคำสั่งในลูป for นี้ โดยเมื่อทำงานเสร็จ โปรแกรมก็จะเพิ่มค่า i อีก 1 ด้วยคำสั่ง i++ ซึ่งหมายความว่าให้ i+1 กลายเป็น i (i = i + 1) ดังนั้น ชุดคำสั่งในลูปนี้ก็จะถูกนำมาประมวล 10 ครั้ง

- ชุดคำสั่ง while เป็นชุดคำสั่งที่ใช้ในการวนลูปโดยตรวจสอบเงื่อนไขก่อน (คล้าย for) มีรูปแบบคือ

```
while (เงื่อนไข) {
        ชุดคำสั่ง
}
```

ในการทำงานของคำสั่ง while นี้ ก็จะตรวจสอบเงื่อนไข โดยหากผลการตรวจสอบเป็นจริง โปรแกรมก็จะทำชุดคำสั่งภายในลูป 1 ครั้ง แล้วจึงกลับมาตรวจสอบเงื่อนไขอีกที แผนผังการทำงานของชุดคำสั่ง while จะคล้ายกับของชุดคำสั่ง for แต่จะไม่มีคำสั่ง1 และคำสั่ง2 ในรูปแผนผัง (รูปที่ 2.3)

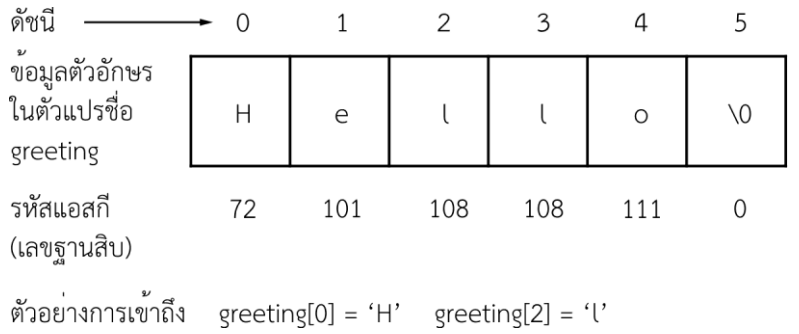
นอกจากคำสั่งควบคุมพื้นฐาน if-else, for และ while ที่กล่าวถึงในหัวข้อนี้ โปรแกรมภาษา C/C++ ยังมีคำสั่งควบคุมอีกหลายตัว ได้แก่ ชุดคำสั่ง switch-case และชุดคำสั่ง do-while นอกจากนี้ยังมีคำสั่งที่ใช้เปลี่ยนเส้นทางการดำเนินการคือคำสั่ง continue, break, goto และ return ที่มิได้กล่าวถึงในรายละเอียดในที่นี้

2.3 อาเรย์และฟังก์ชัน

อาเรย์ (array) เป็นตัวแปรชุดหรือตัวแปรลำดับ ที่เป็นข้อมูลชนิดเดียวกัน รูปแบบอาเรย์ที่มีใช้งานกันมากคืออาเรย์ของข้อมูล 1 มิติ (โดยในทางคณิตศาสตร์จะเรียกว่าเวกเตอร์) และอาเรย์ของข้อมูล 2 มิติ (ที่เรียกว่าเมทริกซ์) ในภาษา C/C++ เราสามารถประกาศตัวแปรอาเรย์ได้โดยใช้เครื่องหมายวงเล็บ [*N*] ตามหลังชื่อของอาเรย์โดย *N* คือขนาดของอาเรย์ ตัวอย่างเช่น ตัวแปรสตริงซึ่งเป็นอาเรย์ 1 มิติของตัวอักษร สามารถประกาศและกำหนดค่าได้ด้วยคำสั่ง

```
char greeting[6] = “Hello”;
```

ตัวอักษร 6 ตัวในข้อความข้างต้น จะถูกกำหนดลงในอาเรย์ของตัวแปรตัวอักษรที่ชื่อว่า greeting โดยมีลำดับการเก็บข้อมูลภายในหน่วยความจำแสดงได้ดังรูปที่ 2.4 โดยสัญลักษณ์ \0 ซึ่งเป็นตัวอักษรสุดท้ายที่ซ่อนอยู่ เป็นตัวอักษณะพิเศษที่ ใช้ในการบอกจุดสิ้นสุดของอาเรย์ข้อความ โดยอาเรย์ในภาษา C/C++ จะมีเลขดัชนี (index) เป็นจำนวนเต็มบวกที่เริ่มจาก 0 เสมอ และเราอาจไม่ใส่ขนาดของอาเรย์ก็ได้หากเรากำหนดข้อมูลที่เก็บในอาเรย์ทันที นั่นคือคำสั่ง char greeting[] = “Hello”; จะให้ผลเหมือนกับผลของตัวอย่างข้างต้น



รูปที่ 2.4 ลักษณะการเก็บข้อมูลแบบอาเรย์ 1 มิติ

บ่อยครั้งในโปรแกรมภาษา C/C++ ที่เราเขียนขึ้นที่เรามิได้ประกาศตัวแปรสตริง แต่นำมาใช้ เช่นการแสดงข้อความผ่านฟังก์ชัน printf ทั้งนี้เพราะว่าเรามิได้ต้องการเปลี่ยนแก้ไขหรือประมวลข้อความเหล่านั้น โดยสำหรับการดำเนินการเกี่ยวกับอาเรย์สตริงนั้น ก็สามารถทำได้ โดยภาษา C/C++ มีไลบรารี string.h ที่รวบรวมคำสั่งมากมายสำหรับการจัดการสตริง เช่น การคัดลอก (strcpy) การรวมสตริง (strcat) การหาความยาวสตริง (strlen) เป็นต้น

อาเรย์ของข้อมูลประเภทต่าง ๆ สามารถประกาศได้ในลักษณะเดียวกันกับอาเรย์ของตัวอักษร ตัวอย่างเช่น

```
int a[3];
```

เป็นการประกาศตัวแปรอาเรย์ 1 มิติ คือ a[0], a[1], และ a[2] โดยการเข้าถึงค่าของสมาชิกแต่ละตัวอาจทำโดยการใช้ชุดคำสั่ง for เช่น

```
int a[3];
for (int i = 0; i < 3; i++) {
    a[i] = 5*i;
}
```

โดยชุดคำสั่งข้างต้นนี้ จะกำหนดค่าให้กับตัวแปร a[0], a[1], และ a[2] ตามลำดับ

สำหรับอาเรย์สองมิติ ก็สามารถกำหนดได้โดยใช้วงเล็บ 2 ชุด ตัวอย่างเช่น

```
float M[3][2];
```

เป็นการประกาศตัวแปรอาเรย์ 2 มิติ ชื่อ M โดยมีสมาชิก 6 ตัว คือ M[0][0], M[0][1], M[1][0], M[1][1], M[2][0], และ M[2][1] ซึ่งการเข้าถึงหรือการเรียกใช้ค่าในอาเรย์นี้ ก็สามารถทำได้โดยใช้ชุดคำสั่ง for ซ้อนกัน เช่น

```
float M[3][2];
for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++) {
                M[i][j] = 3.0*i + j;
                printf("i = %d, j = %d, M[%d][%d] = %f", i, j, i, j, M[i][j]);
        }
}
```

สำหรับการเข้าถึงอาเรย์อีกวิธีหนึ่ง คือการใช้พอยเตอร์ (pointer) ที่เป็นเสมือนตัวแปรที่เก็บข้อมูลของที่อยู่ของข้อมูลอีกทีหนึ่ง การใช้งานตัวแปรพอยเตอร์จะมีประโยชน์มากในการสร้างโครงสร้างข้อมูล (data structure) บางประเภท เช่น ลิงค์ลิสต์ (linked list) และการจัดการกับข้อมูล เช่น การเรียงลำดับ (sorting) ผู้สนใจในส่วนนี้ สามารถศึกษาเพิ่มเติมได้จากหนังสือเกี่ยวกับการเขียนโปรแกรมภาษา C/C++ โดยตรง

ฟังก์ชัน (function) หรือคำสั่งในภาษา C/C++ เป็นสิ่งที่ผู้เขียนโปรแกรมนำมาใช้ โดยฟังก์ชันอาจแบ่งเป็นสองประเภท คือฟังก์ชันภายในตัวภาษา (built-in function) และฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง (external function) ในหัวข้อนี้เราจะกล่าวถึงการสร้างฟังก์ชันขึ้นเอง สำหรับฟังก์ชันพื้นฐานภายในภาษา C/C+ และฟังก์ชันจากไลบรารีมาตรฐาน จะมีจำนวนมาก และผู้ใช้สามารถศึกษาได้จากเอกสารออนไลน์ได้โดยง่าย โดยการค้นหาคำที่เกี่ยวข้อง

รูปแบบของการประกาศฟังก์ชันในภาษา C/C++ คือ

```
ชนิดข้อมูลที่ส่งคืน ชื่อฟังก์ชัน (ชนิดและชื่อข้อมูลที่รับเข้า)
{
        ชุดคำสั่ง
        return( ชื่อตัวแปรที่ส่งค่าคืน );
}
```

โดยคำสั่ง return อาจใช้หรือไม่ใช้ก็ได้ ขึ้นกับว่า ฟังก์ชันที่เรียกมีการคืนค่าหรือไม่ หากไม่มีการคืนค่า จะใช้ คำว่า **void** บ่งบอกชนิดข้อมูลที่ส่งคืนว่า ไม่มีการส่งคืน (void แปลว่า ว่างเปล่า) ตัวอย่างเช่น

ตัวอย่างที่ 1

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

คือฟังก์ชันชื่อ setup ที่ไม่มีการรับค่าใด ๆ เข้าและไม่มีการคืนค่ากลับออกไป

ตัวอย่างที่ 2

```
float cal_area(float x, float y)   {
        float area;
        area = x * y;
        return(area);
}
```

คือฟังก์ชันชื่อ cal_area ที่มีการรับค่าตัวแปร x และ y เข้าและมีการคืนค่าของตัวแปรชื่อ area กลับออกไป โดยตัวแปรที่ประกาศในฟังก์ชันนี้จะมีค่าเฉพาะภายในฟังก์ชันนี้เท่านั้น (เรียกว่า local variable) เมื่อโปรแกรมทำการคำนวณเสร็จ ก็จะล้างหน่วยความจำของตัวแปรภายในออกไป โดยหากเราต้องการให้ตัวแปรนั้นมีลักษณะที่ใช้ได้ทั่วไปตลอดทั้งโปรแกรม (เรียกว่า global variable) เราจะต้องทำการประกาศตัวแปรไว้ภายนอกฟังก์ชัน ซึ่งภาษา C/C++ จะทราบว่าตัวแปรที่ประกาศไว้ภายนอก จะเป็นตัวแปรที่สามารถใช้ได้ในทุกฟังก์ชันที่อยู่ภายในโปรแกรม

2.4 รูปแบบการเขียนโปรแกรมเชิงวัตถุเบื้องต้น

ในเชิงประวัติศาสตร์ ภาษา C++ เป็นภาษาที่พัฒนามาจากภาษา C โดยมีการเพิ่มความสามารถในการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming, OOP) เข้าไปด้วย ซึ่งทำให้ภาษา C++ เป็นภาษาที่มีประสิทธิภาพสูง ผู้ใช้สามารถเขียนโปรแกรมได้หลากหลายรูปแบบ

การเขียนโปรแกรมเชิงวัตถุเป็นแนวคิดในการเขียนโปรแกรมที่กำหนดให้ “วัตถุ” (instance) มีทั้งข้อมูลและกระบวนการอยู่ภายใน โดยข้อมูลนั้นจะอยู่ในฟิลด์ (field) ที่ประกาศไว้ในตอนที่สร้างตัววัตถุ และ กระบวนการที่อยู่ภายในวัตถุจะถูกเรียกว่า เมธอด (method) ซึ่งเมธอดนี้คือฟังก์ชันประเภทหนึ่งที่กำหนดภายในโครงสร้างของ “วัตถุ” และใช้ข้อมูลที่เก็บในตัววัตถุนี้ในการประมวลผลด้วย โดยในการเขียนโปรแกรมขั้นต้น เราไม่จำเป็นต้องสร้างวัตถุขึ้นเอง แต่เราจะต้องเข้าใจรูปแบบของการเขียนโปรแกรมเชิงวัตถุนี้เพื่อให้สามารถนำ “วัตถุ” ที่ผู้อื่นกำหนดมาใช้ได้อย่างถูกต้อง

การใช้งานไลบรารีที่เขียนโปรแกรมเชิงวัตถุนี้ เริ่มต้นจากการประกาศชื่อตัวแปร “วัตถุ” ตามต้นแบบที่กำหนดอยู่ภายในไลบรารีที่เรียกใช้ ตัวอย่าง (นำมาจากบทที่ 6) เช่น ชุดคำสั่ง

```
#include <SH1106.h>
SH1106 display(0x3c, D1, D2);
```

เป็นการเพิ่มไลบรารี SH1106.h เข้ามาในโปรแกรม จากนั้นจึงประกาศตัวแปรวัตถุหรือ instance ที่ให้ชื่อว่า display ซึ่งจะนำโครงสร้างของวัตถุนี้มาจากคลาส (class) ชื่อ SH1106 มาใช้ โดยการสร้างวัตถุที่ชื่อว่า display นี้ถูกกำหนดให้เชื่อมต่อกับหน่วยความจำตำแหน่ง 0x3c และต่อกับขา D1 และ D2 โดย D1 และ D2 เป็นชื่อขาของไมโครคอนโทรลเลอร์ ที่มีการกำหนดตำแหน่งและวิธีการเข้าถึงในหน่วยความจำของเครื่องอยู่ก่อนแล้ว

ผู้ใช้สามารถเข้าถึงเมธอดหรือฟังก์ชันที่กำหนดในไลบรารีได้โดยการเปิดไฟล์ไลบรารีดูได้โดยตรง และการเรียกใช้งานเมธอดเหล่านั้นก็สามารถทำได้โดยการใช้จุด . ซึ่งคำสั่งจะมีรูปแบบคือ

วัตถุ.เมธอด(พารามิเตอร์รับเข้า) หรือ instance.method(parameters);
ตัวอย่างเช่น

display.setFont(ArialMT_Plain_16);

เป็นการเรียกใช้เมธอดชื่อ setFont ซึ่งประกาศในวัตถุนี้ โดยมีพารามิเตอร์ คือ ArialMT_Plain_16 ซึ่งเป็นกำหนดชนิดของตัวอักษรหรือฟอนต์ที่แสดงด้วยวัตถุชื่อ display โดยหากว่าเรามีวัตถุหลาย ๆ ตัว การกำหนดนี้ก็จะไม่ส่งผลใด ๆ ต่อวัตถุอื่น เพราะว่า ตัวแปรที่ถูกเปลี่ยนแปลงค่า ซ่อนอยู่ภายในวัตถุชื่อ display อีกทีหนึ่ง

แนวคิดการเขียนโปรแกรมเชิงวัตถุนี้อาจจะดูซับซ้อนสำหรับผู้ฝึกหัดเขียนโปรแกรม แต่หากศึกษาให้ลึกลงไปจะพบว่า การเขียนโปรแกรมขนาดใหญ่ เช่นโปรแกรมระบบปฏิบัติการ จะมีความซับซ้อนมากหากใช้วิธีการเขียนโปรแกรมแบบดั้งเดิม เนื่องจากจะพบความยุ่งยากในการจำกัดเขตของการกำหนดค่าของตัวแปรต่าง ๆ และการส่งค่าของข้อมูลระหว่างฟังก์ชันจะมีความซับซ้อนมาก การเขียนโปรแกรมเชิงวัตถุนี้ เป็นแนวคิดที่ทำให้การพัฒนาโปรแกรมทำงานได้ง่ายขึ้น โดยผู้ที่นำไลบรารีที่มีการเขียนโปรแกรมลักษณะนี้มาใช้จำเป็นจะต้องศึกษาเพียงฟังก์ชัน (หรือเมธอด) ที่อยู่ภายในเพื่อให้สามารถใช้งานได้ตามต้องการ สำหรับการนำแนวคิดของการเขียนโปรแกรมเชิงวัตถุมาใช้กับการพัฒนาฮาร์ดแวร์ด้านไมโครคอนโทรลเลอร์จะทำให้เราสามารถมองเสมือนว่า ฮาร์ดแวร์ต่าง ๆ ที่นำมาต่อ เป็นเสมือนกับวัตถุหนึ่ง ๆ ที่เราสามารถเขียนโปรแกรมสั่งการได้ตามความต้องการโดยตรง โดยการเรียกเมธอดที่มีอยู่มาใช้งาน

บทที่ 3

แนะนำสถาปัตยกรรมของ NodeMCU v.3

ในบทนี้จะเป็นการแนะนำบอร์ด NodeMCU v.3 ซึ่งเป็นไมโครคอนโทรลเลอร์หลักที่เลือกมานำเสนอในหนังสือเล่มนี้ โดยการประยุกต์ใช้อุปกรณ์ต่าง ๆ ที่นำเสนอในบทต่อ ๆ ไปก็จะใช้บอร์ดนี้เป็นฐานในการเชื่อมต่อ

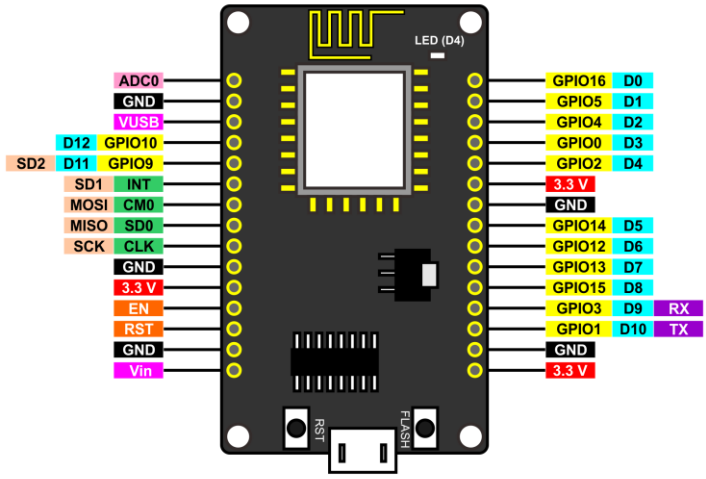
3.1 คุณสมบัติของ NodeMCU v.3

NodeMCU คือ แพลตฟอรมหนึ่งที่ใชชวยในการสรางโปรเจคอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things, IoT) ที่ประกอบไปดวยตัวบอร์ด และ ซอฟต์แวร์ในตัว (Firmware) ที่เปนลักษณะโอเพ่นซอร์ส (open-source) ในตอนแรก ผู้ใช้จะต้องเขียนโปรแกรมดวยภาษา Lua เพื่อให้ใช้งาน NodeMCU ได แต่ต่อมาได้พัฒนาให้สามารถใช้ภาษา C ใน Arduino IDE เพื่อควบคุมได้ ทำใหใชงานไดงายขึ้น โดยบอร์ดนี้มาพรอมกับโมดูล WiFi (ESP8266) ซึ่งเปนหัวใจสำคัญในการใชเชื่อมตอกับอินเทอรเน็ต ตัวโมดูล WiFi (ESP8266) นั้นมีอยูดวยกันหลายรุน ตั้งแตเวอรชันแรกคือ ESP-01 จนในมีถึงเวอร์ชัน ESP-12 โดยโมดูลนี้ที่เป็นองค์ประกอบใน NodeMCU รุ่นแรกนั้นคือ ESP-12 และในเวอร์ชันต่อมา (NodeMCU v.2) นั้นจะใชโมดูล ESP-12E แทน ซึ่งในการใชงานโดยรวมนั้นจะไมแตกตางกันมากนัก แต่บอร์ดเวอร์ชัน 3 จะมีขนาดใหญ่ขึ้น NodeMCU นั้นมีลักษณะคลายกับ Arduino ตรงที่ มีพอรตขาเข้าหรืออินพุตพอร์ต (input port) และ พอร์ตขาออกหรือเอาต์พุตพอร์ต (output port) อยู่ในตัวเอง ทำให้เราสามารถเขียนโปรแกรมควบคุมอุปกรณ์ต่าง ๆ ไดโดยตรง ไมตองทำผานชิปอื่น ๆ NodeMCU v.3 นี้สามารถทำอะไรไดหลายอยางมาก โดยเฉพาะงานที่เกี่ยวของกับ IoT

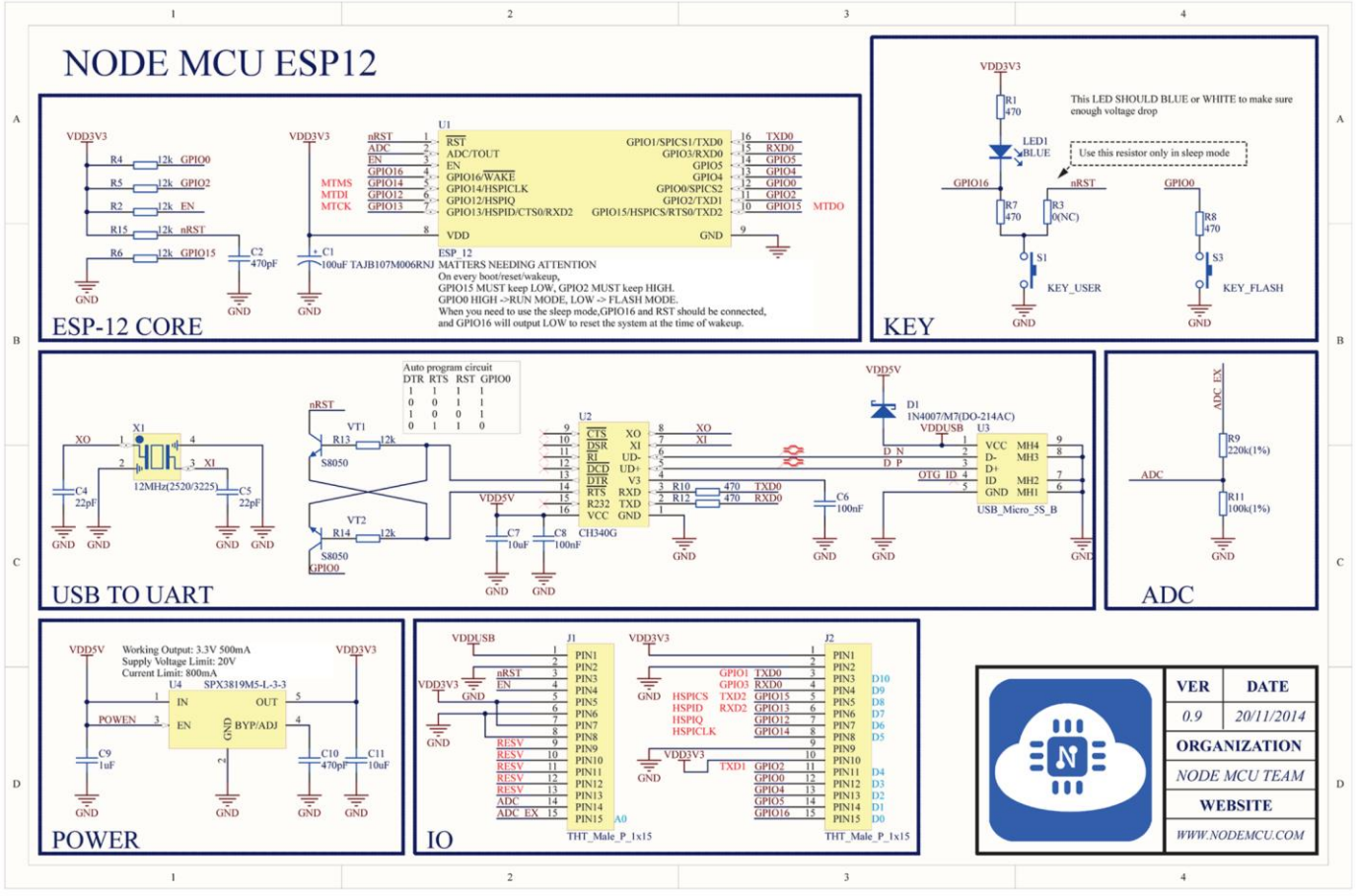
ไมวาจะเป็น การทำเว็บเซิร์ฟเวอร์ขนาดเล็ก การควบคุมการเปดปดไฟผานเครือข่าย และอื่น ๆ อีกมากมาย ตัวบอร์ด NodeMCU v.3 แสดงดังรูปที่ 3.1

รูปที่ 3.2 แสดงตำแหน่งพินและขาของ NodeMCU v.3 โดยมีขาที่ใช้งานบ่อย ๆ คือ ขา GPIO (General Purpose Input/Output) หรือขา D0 – D12 ขา ADC (Analog to Digital) หรือขา A0 และ ขาไฟเลี้ยง (VUSB และ 3.3 V) และกราวด์ (Ground, GND)

รูปที่ 3.1 ภาพบอร์ด NodeMCU v.3



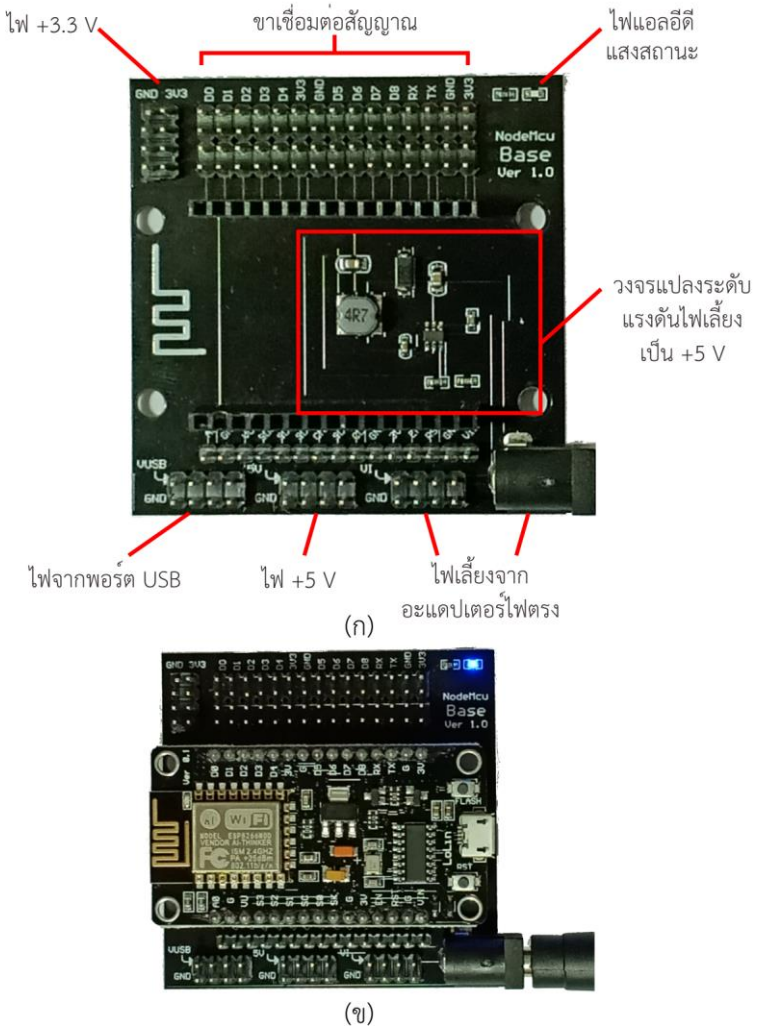
รูปที่ 3.2 ตำแหนงพินและขาของ NodeMCU v.3



รูปที่ 3.3 วงจรของบอรด NodeMCU v.3 (ภาพจาก https://github.com/nodemcu/nodemcu-devkit)

3.2 NodeMCU Base v.1

เพื่อความสะดวกในการใช้งาน NodeMCU ซึ่งเป็นบอร์ดขนาดเล็ก ผู้พัฒนาได้สร้าง NodeMCU Base ขึ้นเพื่อเป็นฐานสำหรับการเชื่อมต่อต่าง ๆ ดังแสดงในรูปที่ 3.4(ก) และมีลักษณะการใช้งาน ดังแสดงในรูปที่ 3.4(ข)



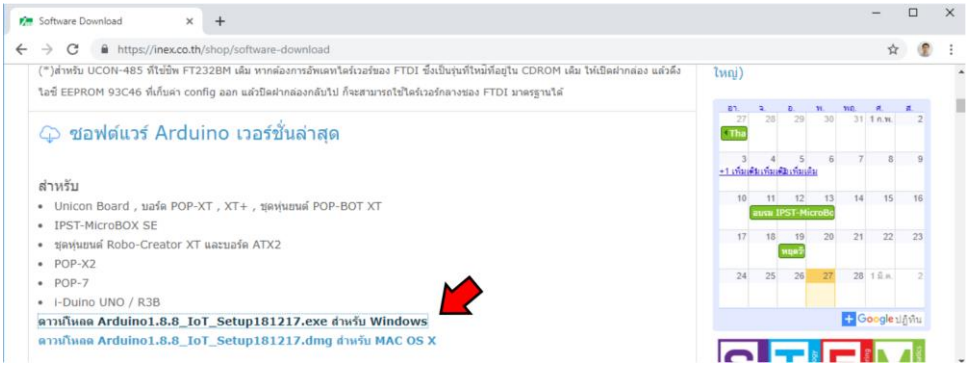
รูปที่ 3.4 (ก) บอรด NodeMCU Base Version 1.0
และ (ข) บอร์ด NodeMCU และ NodeMCU Base

3.3 การติดตั้งโปรแกรม Arduino IDE สำหรับ NodeMCU

การติดตั้งโปรแกรม Arduino IDE เป็นการติดตั้งโปรแกรมเพื่อใช้ในการเขียนโปรแกรมให้กับบอรด ESP8266 หรือบอรดอื่น ๆ ผานทาง Arduino IDE ซึ่งจะใช้หลักการของภาษา C/C++ ที่กล่าวถึงโดยย่อในบทที่ 2 การเชื่อมต่อกับบอร์ดนี้ทำได้ด้วยวิธีการเดียวกับการเชื่อมต่อบอร์ด Arduino อื่น ๆ เพื่อให้การเขียนโปรแกรมมีความงายมากยิ่งขึ้น สำหรับผู้ที่คุ้นเคยกับบอร์ด Arduino อยู่ก่อนแล้ว ขั้นตอนการติดตั้งมีรายละเอียดดังต่อไปนี้

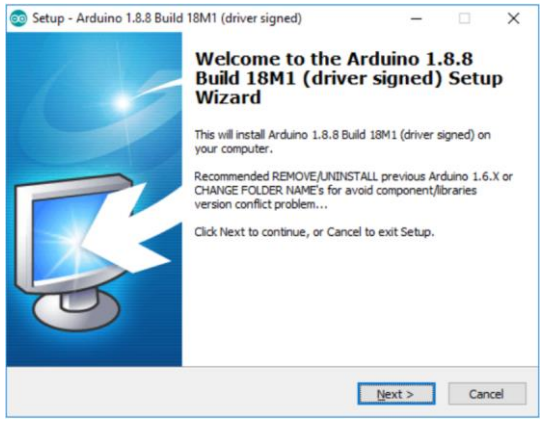
ดาวน์โหลดโปรแกรม

โปรแกรมที่เราใช้ในการทดสอบคือ Arduino IDE 1.8.8 IoT โดยโปรแกรมจะรองรับระบบปฏิบัติการ Windows, Mac OS X, Linux ทั้งแบบ 32 บิต และ 64 บิต ซึ่งเราสามารถดาวน์โหลดตัวติดตั้งไดจาก https://www.arduino.cc/en/main/software หรือ https://inex.co.th/shop/software-download รูปที่ 3.5 แสดงหน้าต่างดาวน์โหลด Arduino IDE ของเว็บบริษัท Inex สำหรับได้ติดตั้ง Arduino IDE (จาก arduino.cc) จะต้องทำการติดตั้ง Board Manager URL เพิ่มเติม และดาวน์โหลดส่วนขยายสำหรับ ESP8266 โดยมีรายละเอียดแสดงในภาคผนวก ก

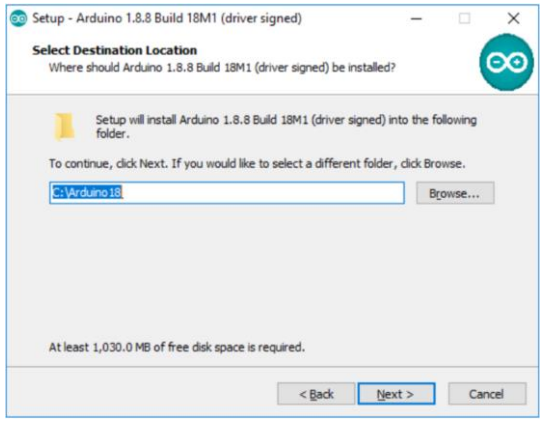


รูปที่ 3.5 หน้าต่างดาวนโหลด Arduino IDE ที่มีส่วนขยาย IoT จาก https://inex.co.th/shop/software-download

สำหรับการติดตั้ง Arduino IDE บน Windows จะมีแบบให้เลือกทั้ง แตกไฟล์ใช้ได้เลย (ZIP file for non admin install) หรือ ติดตั้งเหมือนโปรแกรมอื่นทั่ว ๆ ไป คือ รันโปรแกรมติดตั้ง (Installer) หากดาวน์โหลดเรียบร้อยแล้วให้เปิด Arduino 1.8.8_IoT_Setup ขึ้นมาดังรูปที่ 3.6 โดยเลือกติดตั้งโปรแกรมลงที่ **C:\Arduino18** ดังรูปที่ 3.7



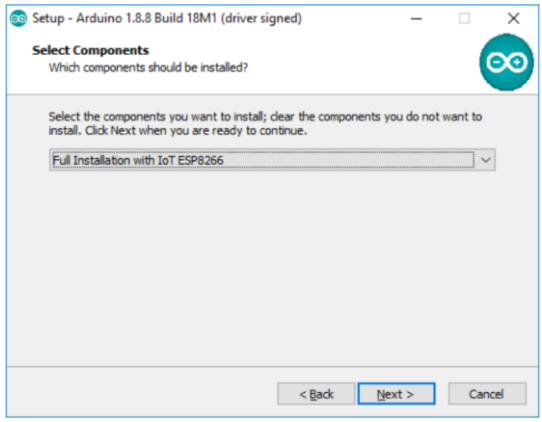
รูปที่ 3.6 หน้าต่างเริ่มต้นการติดตั้งโปรแกรม Arduino IDE



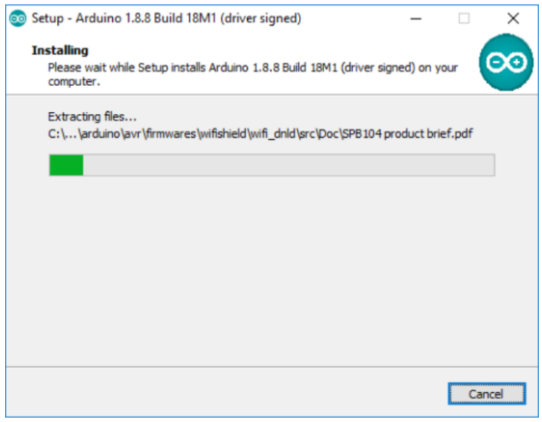
รูปที่ 3.7 หน้าต่างเลือกโฟลเดอร์ที่ติดตั้งโปรแกรม Arduino IDE

เลือกติดตั้งบอร์ด ESP8266 ลงใน Arduino IDE

หลังจากเลือกติดตั้ง Arduino IDE แล้วก็ให้กดเลือกติดตั้งแบบ Full Installation with IoT ESP2866 ดังรูปที่ 3.8 จากนั้นก็เลือกติดตั้งไอคอนลงบน Start Menu และ เลือกติดตั้งโปรแกรม จากนั้นโปรแกรมจะถูกติดตั้งดังรูปที่ 3.9



รูปที่ 3.8 หนาตางแสดงการเลือกการติดตั้ง ESP2866 add-on

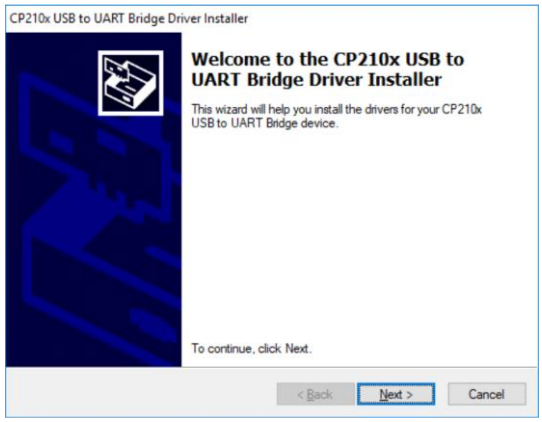


รูปที่ 3.9 หนาตางแสดงการติดตั้งโปรแกรม Arduino IDE

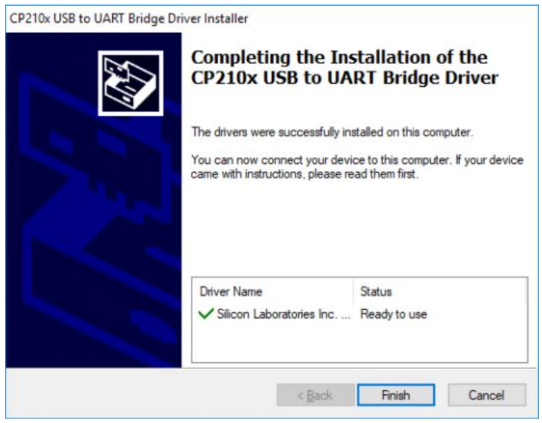
รอจนกวาจะติดตั้งโปรแกรม Arduino IDE เสร็จสิ้น จากนั้น จะมีหน้าต่างปรากฏขึ้นให้ผู้ติดตั้งทำการติดตั้งไดรเวอร CP210x USB to UART โดยกดปุม Next ดังรูปที่ 3.10 และอ่านและกดเลือกยอมรับ (accept) ข้อตกลงเรื่องไลเซนต์ จากนั้นไดรเวอร์จะติดตั้งลงในคอมพิวเตอร์ และแสดงผลลัพธ์ดังแสดงในรูปที่ 3.11 จากนั้นเป็นการสิ้นสุดการติดตั้ง

สำหรับหน้าต่างเลือกติดตั้งไดรเวอร์อื่น ๆ ที่อาจปรากฏขึ้นหลังติดตั้งโปรแกรม Arduino IDE นั้น ขอให้ผู้ติดตั้งเลือกติดตั้งไดรเวอร์เหล่านั้นด้วย

หลังจากติดตั้งเสร็จ โปรแกรมจะให้เลือกลักษณะตั้งต้นของหน้าตาโปรแกรม ขอให้เลือก Arduino ดังรูปที่ 3.12



รูปที่ 3.10 หนาตางแสดงการติดตั้งไดรเวอร์ CP210x USB to UART



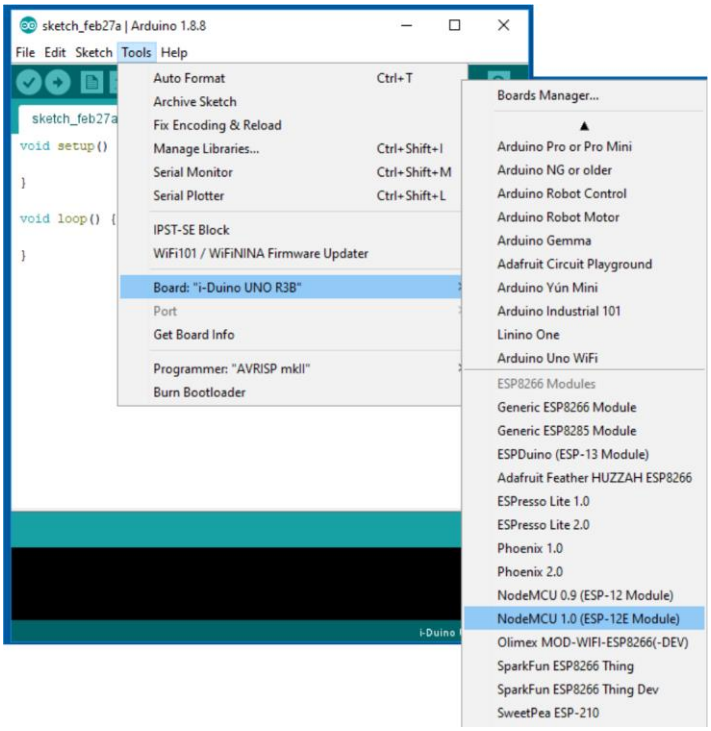
รูปที่ 3.11 หนาตางแสดงการผลการติดตั้งไดรเวอร์ CP210x USB to UART

รูปที่ 3.12 หน้าต่างแสดงตัวเลือกลักษณะตั้งต้นของบอร์ดสำหรับโปรแกรม Arduino IDE

3.4 การเริ่มต้นทดสอบโปรแกรม Arduino IDE

เลือกบอร์ดในโปรแกรม Arduino IDE

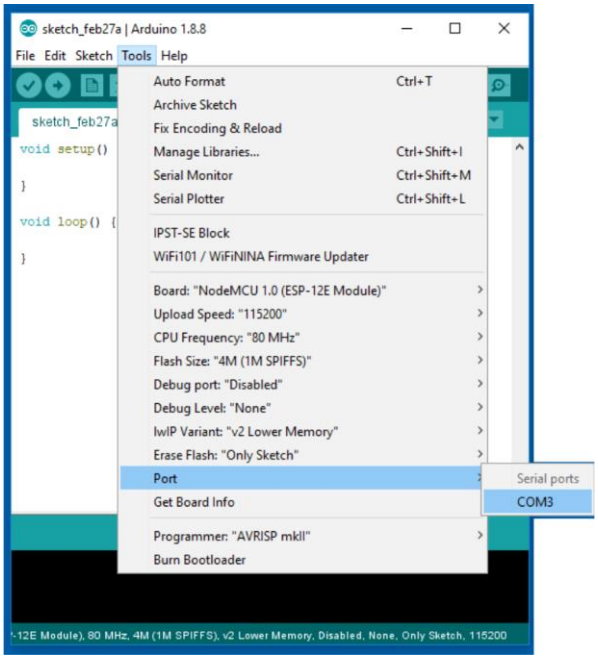
เมื่อติดตั้งโปรแกรม Arduino IDE ตามขั้นตอนในหัวข้อที่แล้วเสร็จสมบูรณแลว ขอให้เปดโปรแกรม Arduino 1.8.8 และทำการเลือกบอรด โดยที่เลือกที่เมนู Tools แลวเลือก Board: **NodeMCU 1.0 (ESP-12E Module)** ดังรูปที่ 3.13 เพื่อใชเขียนโปรแกรมควบคุม



รูปที่ 3.13 การเลือก Board: NodeMCU 1.0 (ESP-12E Module) ในโปรแกรม Arduino IDE

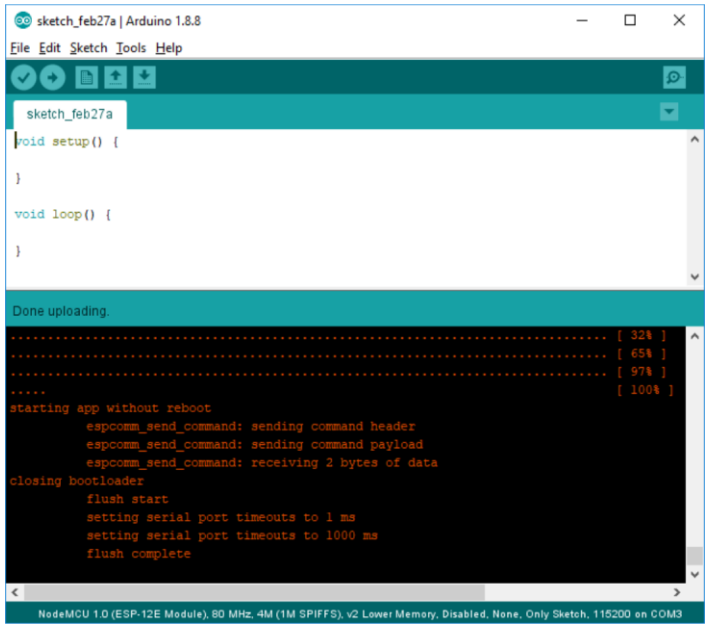
การเชื่อมต่อ NodeMCU v.3 กับคอมพิวเตอร์

ขอให้ตอสาย USB เขากับ NodeMCU แลวตอสาย USB อีกด้านเขาที่เครื่องคอมพิวเตอร หลังจากนั้นใหกำหนดชองทางเชื่อมตอโปรแกรม Arduino IDE กับ NodeMCU โดยเลือกที่ดูที่เมนู Tool >> Ports: สำหรับตัวอย่างนี้ NodeMCU ที่เชื่อมต่อ จะต่ออยู่ที่ COM3 ดังนั้นจึงเลือก Port เป็น COM3 ดังรูปที่ 3.14



รูปที่ 3.14 การเลือก Port ที่ติดต่อกันระหว่าง NodeMCU และโปรแกรม Arduino IDE

ขอให้ทำการทดลองอัปโหลดโปรแกรม โดยคลิกที่ปุม (Upload) หรือไปที่เมนู Sketch >> Upload โดยขอให้สังเกตในขณะที่อัปโหลด LED ที่มุมข้าง ๆ ชิป ESP8266 (D4) จะกระพริบ และหน้าต่างข้อความด้านล่างของโปรแกรมจะแสดงเปอรเซ็นตการอัปโหลด รอจนการอัปโหลดจะเสร็จสิ้น หากไม่พบข้อความบอกความผิดพลาด (error message) ใด ๆ ก็แสดงว่าบอร์ด NodeMCU v.3 ติดต่อกับโปรแกรม Arduino IDE ได้สำเร็จดังรูปที่ 3.15



รูปที่ 3.15 รูปแสดงการอัปโหลดโปรแกรมลงสู่บอร์ด NodeMCU v.3

โครงสร้างโปรแกรม Arduino

ในการเขียนโปรแกรมสำหรับควบคุมบอรด NodeMCU v.3 ในการอบรมนี้เราจะตองเขียนโปรแกรมโดยใชภาษาของ Arduino (Arduino programming language) ซึ่งตัวภาษาของ Arduino เองก็นำเอาโอเพนซอรสโปรเจ็กต์ที่ชื่อ wiring มาพัฒนาตอ ภาษาของ Arduino แบงไดเปน 2 สวนหลักคือ

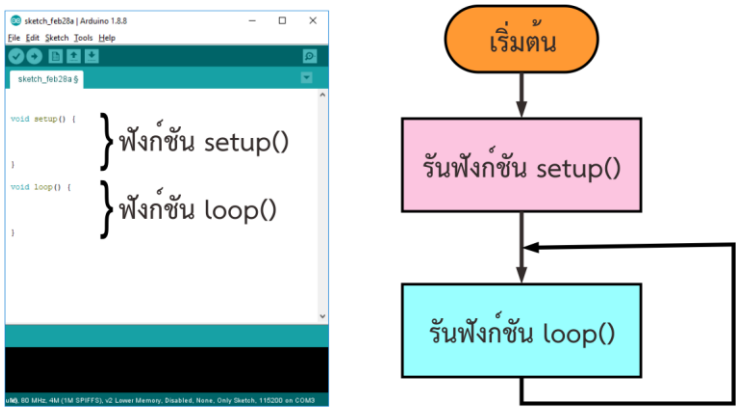
1. โครงสรางภาษา (structure) ตัวแปรและคาคงที่
2. ฟงกชัน (function)

ภาษา Arduino จะอางอิงวิธีการเขียนโปรแกรมตามภาษา C/C++ จึงอาจกลาวไดวาการเขียนโปรแกรมสำหรับบอร์ด Arduino (ซึ่งก็รวมถึงบอรด NodeMCU ด้วย) ก็คือการเขียนโปรแกรมภาษา C/C++ โดยเรียกใชฟงกชันและไลบรารีต่าง ๆ ที่ทางผู้พัฒนา ไดเตรียมไวใหแลวซึ่งสะดวกและ ทำใหผูที่ไมมีความรูดานไมโครคอนโทรลเลอรอยางลึกซึ้งสามารถเขียน

โปรแกรมสั่งงานได โดยโปรแกรมภาษา Arduino มีสองสวนหลัก ๆ คือ ฟังก์ชัน setup() และฟังก์ชัน loop() ดังรูปที่ 3.16

เมื่อโปรแกรมที่เขียนถูกคอมไพล์ ก็จะมีการกำหนดตัวแปรต่าง ๆ ที่อยู่ด้านบนของโค้ดก่อนจะเข้าสู่ฟังก์ชัน setup() โดยการทำงานของโปรแกรมนั้นจะเริ่มที่ฟงกชัน setup() ซึ่งจะมีการรันคำสั่งในฟงกชันนี้เพียงครั้งเดียว ฟังก์ชันนี้อาจใชในการกำหนดคาเริ่มต้นต่าง ๆ ของการทำงาน สวนฟงกชัน loop() เปนสวนทำงานโปรแกรมจะทำคำสั่งในฟงกชันนี้อย่างตอเนื่องกันตลอดเวลา โดยปกติคำสั่งในฟังก์ชัน setup() จะใชกำหนดการตั้งค่าการทำงานของขาตาง ๆ ของบอร์ด รวมถึงกำหนดการสื่อสารแบบอนุกรมกับคอมพิวเตอร์

ฟงกชัน loop() เปนโคดหลักของโปรแกรมที่ทำงานซ้ำ ๆ เชน อานคาอินพุต ประมวลผล กำหนดค่าเอาตพุต ฯลฯ โดยสวนกำหนดคาเริ่มตนเชนตัวแปรจะตองเขียนที่สวนหัวของโปรแกรมกอนถึงตัวคำสั่ง เช่นเดียวกับการเขียนโปรแกรมในภาษา C/C++ โดยเราจะตองคำนึงถึงตัวพิมพเล็ก-ใหญของชื่อตัวแปรและชื่อของฟงกชันด้วย คือจะต้องเขียนชื่อตัวแปรเหล่านี้ใหถูกตองตรงกันเสมอ สำหรับการอบรมนี้ เราจะเรียนรู้วิธีการและเทคนิคการเขียนโปรแกรม ภาษา C/C++ ใน Arduino IDE ผ่านการทดลองในแต่ละการทดลอง



(ก) (ข)

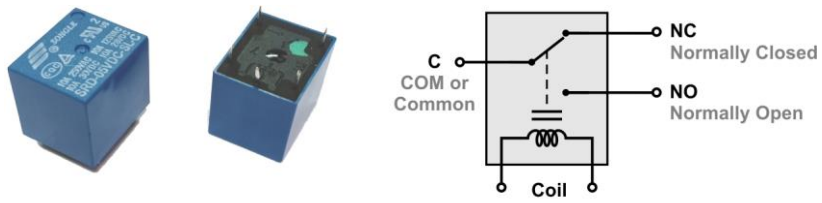
รูปที่ 3.16 (ก) ลักษณะโค้ดเริ่มต้นและ (ข) แผนผังการทำงานพื้นฐานของ Arduino

บทที่ 4

การใช้งานรีเลย์

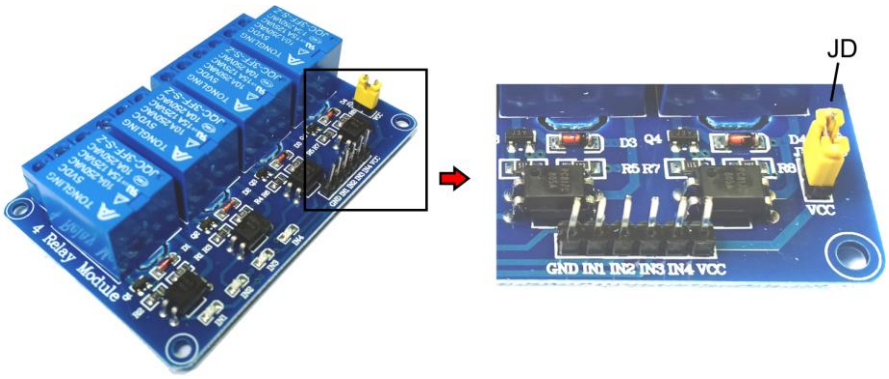
4.1 รีเลย์

รีเลยเปนอุปกรณอิเล็กทรอนิกสที่ใชในการตัดตอวงจรคลายกับสวิตช รีเลย์ทั่วไปจะใช้ปรากฏการณ์ทางไฟฟ้าแม่เหล็กในการทำงาน และเรียกวาเป็นแบบมีหนาสัมผัส รีเลย์หนึ่ง ๆ จะประกอบดวยชุดหนาสัมผัส (Contact) ที่ตอกับ แท่งอารเมเจอร (Armature) และคอยล (Coil) ที่ถูกพันดวยขดลวด เมื่อมีการจายแรงดันไฟฟาใหกับคอยลจะทำใหเกิดสนามแมเหล็ก แทงอารเมเจอรที่ตอกับหนาสัมผัสจะถูกดูดด้วยแรงแม่เหล็ก ทำใหหนาสัมผัสเปลี่ยนสถานะการเชื่อมตอเปนสถานะตรงกันขาม กลาวคือ ปกติเปด (NO, Normally Open) จะกลายเปนปด และปกติปด (NC, Normally Closed) จะกลายเปนเปด และเมื่อตัดไฟเลี้ยงที่จายใหคอยล จะทำใหรีเลยกลับสูสถานะเดิม คือหนาสัมผัสตางๆ จะกลับสูสภาวะแรกกอนการจายไฟดวยแรงจากสปริงที่ติดตั้งอยู่ภายใน รูปรางทั่วไปของรีเลย์ขั้วเดียวสองทาง (Single Pole Double Throw, SPDT) และลักษณะภายในของรีเลย์แสดงดังรูปที่ 4.1 (ก) และ (ข) และ บอรดรีเลยที่ใชในการทดลองแสดงดังรูปที่ 4.2 โดยบอร์ดที่เลือกใช้นี้มีวงจรควบคุมรีเลยอยู่ 4 ชุด



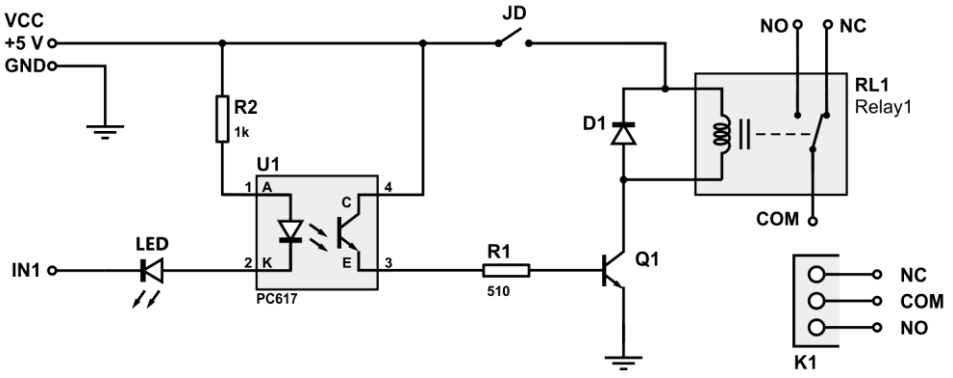
(ก) (ข)

รูปที่ 4.1 (ก) รูปร่างและ (ข) ลักษณะภายในของรีเลย์



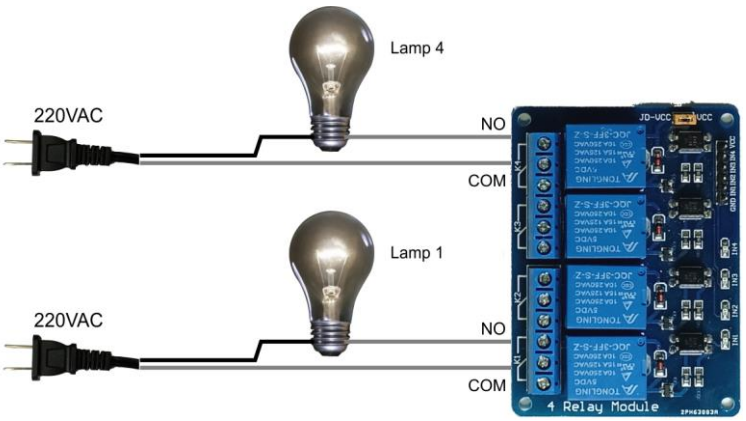
รูปที่ 4.2 บอร์ดรีเลย์ชนิด 4 ช่อง แบบ isolation control, 250V/10A และ Active Low

การทำงานของวงจรควบคุมรีเลยขึ้นอยู่กับลักษณะการต่อวงจร โดยสำหรับบอร์ดที่แสดงในรูปที่ 4.2 นี้ รีเลยจะทำงานเมื่อได้รับสัญญาณลอจิกต่ำ (Active Low) ที่ขา IN แต่ละขา ซึ่งเราจะทำการเชื่อมตออยู่กับขาของ NodeMCU v.3 โดยเมื่อป้อนสัญญาณระดับต่ำ (LOW) ที่ขา IN1 (ดูรูปที่ 4.3) แล้ว กระแสไฟฟาจะไหลผาน LED ทำให้ LED สวาง สวนภายในชิฟ U1 นั้น เมื่อมีกระแสไหลผ่านก็จะมีการส่งสัญญาณแสงทำใหทรานซิสเตอร์ใน U1 นำกระแสและทำให้ทรานซิสเตอร์ Q1 มีสถานะนำกระแสด้วย ซึ่งก็จะทำให้รีเลย์ทำงาน (คือมีการเปลี่ยนสถานะ) สำหรับไดโอด D1 ที่ต่อขนานกับคอยล์ของรีเลย์นั้น เพื่อทำให้กระแสไฟฟ้าสามารถไหลย้อนกลับทางได้ในช่วงเวลาสั้น ๆ ขณะที่รีเลย์เปลี่ยนสถานะ



รูปที่ 4.3 วงจรบอร์ดรีเลย์ (เฉพาะส่วนของรีเลย์ตัวที่ 1)

รูปที่ 4.4 แสดงตัวอย่างการนำรีเลย์ไปใช้ควบคุมอุปกรณ์ไฟฟ้าทั่วไป เช่น หลอดไฟ โดยเราสามารถต่ออุปกรณ์แรงดัน 220 V กระแสสูงสุด 10 แอมแปร์กับบอร์ดรีเลย์นี้ได้ โดยในการทดลองเบื้องต้น เราจะไม่มีการต่อไฟฟ้าแรงดัน 220 V จริง ดังนั้นเราอาจจะเอาจัมเปอร์ JD บนบอร์ดรีเลย์นี้ออก (ดูรูปที่ 4.2 และ รูปที่ 4.3) เพื่อมิให้รีเลย์ทำงานจริงแต่จะสังเกตการทำงานผ่านหลอด LED ที่แสดงสถานะของรีเลย์บนบอร์ดรีเลย์แทน



รูปที่ 4.4 ตัวอย่างการนำรีเลย์ไปควบคุมกับอุปกรณ์ไฟฟ้าทั่วไป

4.2 การเขียนโปรแกรมส่งออกค่าดิจิทัล

คำสั่งเบื้องต้นในการใช้งานรีเลย์ ในภาษา Arduino แสดงแยกเป็นหัวข้อย่อย ดังนี้

1. ฟงกชันเอาตพุตแบบดิจิทัล (Digital Output)

1) **pinMode(pin, mode)** เปนคำสั่งใชกำหนดขา (หรือพอรต) ใด ๆ ที่ทำหนาที่เปนพอรตดิจิทัล ปกติจะใช้ในฟังก์ชัน setup() เพื่อกำหนดการทำงานเริ่มต้นของขาต่าง ๆ ที่เกี่ยวข้อง

- pin หมายถึงชื่อหรือเลขขาของบอร์ด NodeMCU เปน D0, D1, ... หรือเป็นเลขจำนวนเต็ม โดยชื่อขาเหล่านี้ ถูกกำหนดให้สอดคล้องกับพิน GPIO อยู่ก่อนแล้ว (ดูรูปที่ 3.2)
- mode หมายถึงโหมดการทำงาน กำหนดให้มีค่าเปน INPUT หรือ OUTPUT เท่านั้น

ตัวอย่างเชน คำสั่ง pinMode(D1, OUTPUT) คือการกำหนดใหขา D1 ของ NodeMCU ทำงานโดยส่งสัญญาณออกไปเท่านั้น

2) **digitalWrite(pin, value)** เปนคำสั่งให้ขาที่กำหนดมีสถานะเปนลอจิกสูง (HIGH หรือ “1”) หรือเปนลอจิกต่ำ (LOW หรือ “0”)

- pin หมายถึงชื่อหรือเลขขาของบอร์ด NodeMCU
- value หมายถึงคา กำหนดให้มีค่าเปน HIGH หรือ LOW เท่านั้น

ตัวอย่างเชน คำสั่ง digitalWrite(D0, HIGH) คือการกำหนดใหขา D0 ของ NodeMCU มีคาเปนลอจิกสูง (HIGH) ซึ่งจะมีระดับแรงดัน 3.3 V

2. ฟงกชันหนวงเวลา (Time Delay)

1.) **delay(value)** เปนคำสั่งใหหนวงเวลา

- value หมายถึงคาการหนวงเวลามีหนวย 1/1000 วินาที หรือมิลลิวินาที (ms) โดยจะรับค่าที่มีชนิดเป็นเลขจำนวนเต็ม (integer, int) เท่านั้น

ตัวอย่างเชน delay(1000); คือการกำหนดใหโปรแกรมหนวงเวลา 1 วินาที (1 s = 1000 ms = 1000000 μs)

2.) **delayMicroseconds(time)** เปนคำสั่งใหหนวงเวลา

- time หมายถึงคาการหนวงเวลามีหนวย 1/1000000 วินาที (μs หรือ us โดยเรามักใช้ ‘u’ แทน สัญลักษณ์ไมโคร ‘μ’) ฟังก์ชันนี้จะรับค่าเป็นเลขจำนวนเต็ม

ตัวอย่างเชน delayMicroseconds(1000); กำหนดใหการหนวงเวลามีคาเทากับ 1 มิลลิวินาที

ข้อมูลเพิ่มเติม: สำหรับตัวแปร int (คือจำนวนเต็มหรือ integer) ในภาษา C จะมีค่าระหว่าง -32768 ถึง 32767 (-2^{15} ถึง $2^{15}-1$) ในขณะที่ ตัวแปร unsigned int (คือจำนวนเต็มแบบไม่คิดเครื่องหมาย) จะมีค่าระหว่าง 0 ถึง 65535 (0 ถึง $2^{16}-1$)

4.3 การทดลองควบคุมรีเลย์

วัตถุประสงค์

1. สามารถตอบอร์ด NodeMCU v.3 กับบอร์ดรีเลย์ได
2. สามารถเขียนโปรแกรมให้ NodeMCU ควบคุมการทำงานของรีเลย์ได
3. เขียนโปรแกรมควบคุมรีเลย์โดยมีการหนวงเวลาประกอบด้วยได

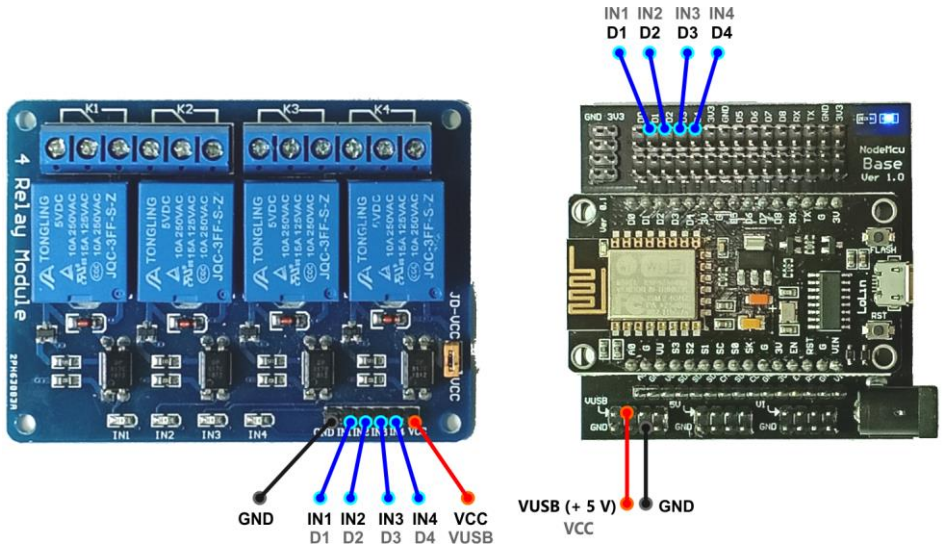
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดรีเลย์ชนิด 4 ช่อง	1 บอร์ด
5.	สาย USB	1 เส้น
6.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	6 เส้น

วิธีการทดลอง

1. ตอวงจรตามรูปที่ 4.5 โดยต่อทางดานอินพุท (input) ของรีเลย์หมายเลข 1 (IN1) ที่ขา D1, รีเลย์หมายเลข 2, 3, และ 4 (IN2, IN3, และ IN4) ที่ขา D2, D3 และ D4 สวนขา GND (ขากราวด์) ให้ต่อร่วมกันและ ต่อขา VCC กับ VUSB ซึ่งคือไฟบวก +5 V ที่บอร์ด NodeMCU Base Ver 1.0 ที่ได้รับมาจากคอมพิวเตอร์ จากนั้นดึงจัมเปอร์ JD บนบอร์ดรีเลย์ออกก่อนเพื่อมิให้ไฟเข้าไปเลี้ยงคอยล์และสังเกตการทำงานของรีเลย์ผ่าน LED บนบอร์ดก่อน

ในการเขียนโปรแกรมควบคุมใหรีเลย์ที่ทำงานแบบ Active Low เราตองป้อนสัญญาณลอจิกต่ำ (LOW) ที่ตำแหนงขา ของ NodeMCU ที่ต่อเพื่อควบคุมรีเลย์ ดังนั้น เราจึงใช้คำสั่ง #define ในการกำหนดชื่อ ให้ LOW คือ ON และ HIGH คือ OFF เพื่อให้คำสั่งที่เขียนสอดคล้องกับการทำงานของรีเลย์

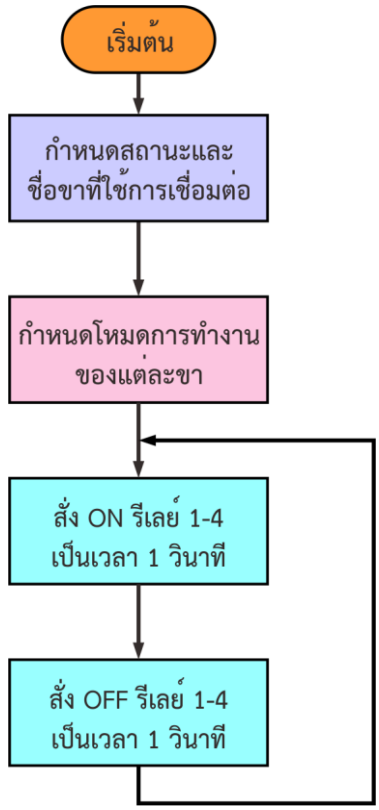


รูปที่ 4.5 การต่อ NodeMCU v.3 เพื่อควบคุมการเปิด-ปิดรีเลย์

2. เขียนโค้ดควบคุมรีเลย์ในโปรแกรม Arduino IDE เพื่อควบคุมการเปด-ปดการทำงานของรีเลย 4 ตัวพร้อม ๆ กัน โดยให้เปิด-ปิดเป็นเวลา 1 วินาทีสลับกันไปเรื่อย ๆ ตลอดเวลา แผนผังการทำงานของโปรแกรมในการทดลองนี้แสดงได้ดังรูปที่ 4.6

```
// Control 4-channel Relay by NodeMCU ESP8266

#define ON LOW
#define OFF HIGH
int Relay1 = D1;
int Relay2 = D2;
int Relay3 = D3;
int Relay4 = D4;

void setup() {
    pinMode(Relay1, OUTPUT);
    pinMode(Relay2, OUTPUT);
    pinMode(Relay3, OUTPUT);
    pinMode(Relay4, OUTPUT);
}

```

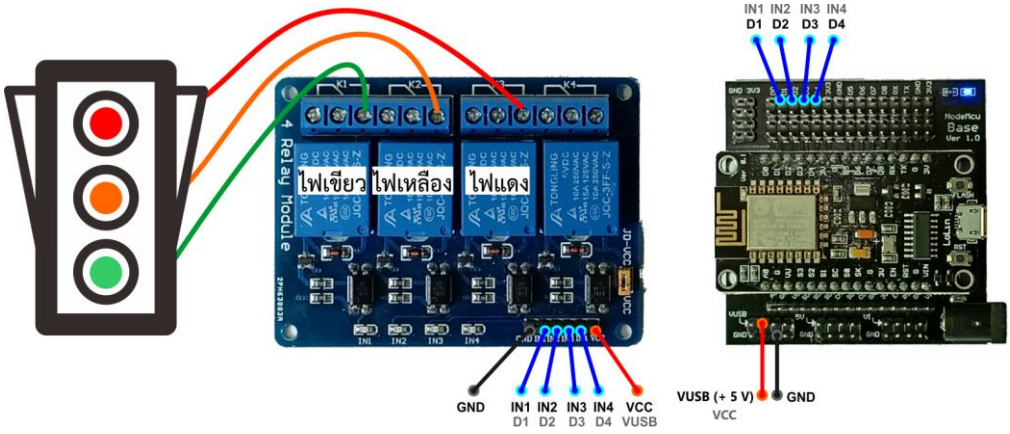
```
17 void loop() {
18     digitalWrite(Relay1, ON);
19     digitalWrite(Relay2, ON);
20     digitalWrite(Relay3, ON);
21     digitalWrite(Relay4, ON);
22     delay(1000);
23     digitalWrite(Relay1, OFF);
24     digitalWrite(Relay2, OFF);
25     digitalWrite(Relay3, OFF);
26     digitalWrite(Relay4, OFF);
27     delay(1000);
28 }
29
```



รูปที่ 4.6 แผนผังการทำงานของโปรแกรมที่แสดงในการทดลองนี้

แบบฝึกหัดท้ายการทดลอง

หากเราต้องการสร้างสัญญาณไฟจราจร (ไฟเขียว-เหลือง-แดง) โดยใช้ NodeMCU และบอร์ดรีเลย์ควบคุมการทำงานของไฟสัญญาณ (ไฟสัญญาณแดง-เหลือง-เขียว) (ดูรูปที่ 4.7) โดยกำหนดให้ระบบควบคุมรีเลย์ทำงานดังแสดงในตารางที่ 4.1 จงเขียนโค้ดโปรแกรมสำหรับการสร้างสัญญาณไฟจราจรนี้



รูปที่ 4.7 รูปประกอบแบบฝึกหัดท้ายการทดลอง

ตารางที่ 4.1 สถานะของรีเลย์ที่ช่วงเวลาต่าง ๆ เพื่อใช้ควบคุมไฟสัญญาณจราจร

เวลา (วินาที)	รีเลย์ 1 สีเขียว	รีเลย์ 2 สีเหลือง	รีเลย์ 3 สีแดง
10	ON	OFF	OFF
3	OFF	ON	OFF
8	OFF	OFF	ON

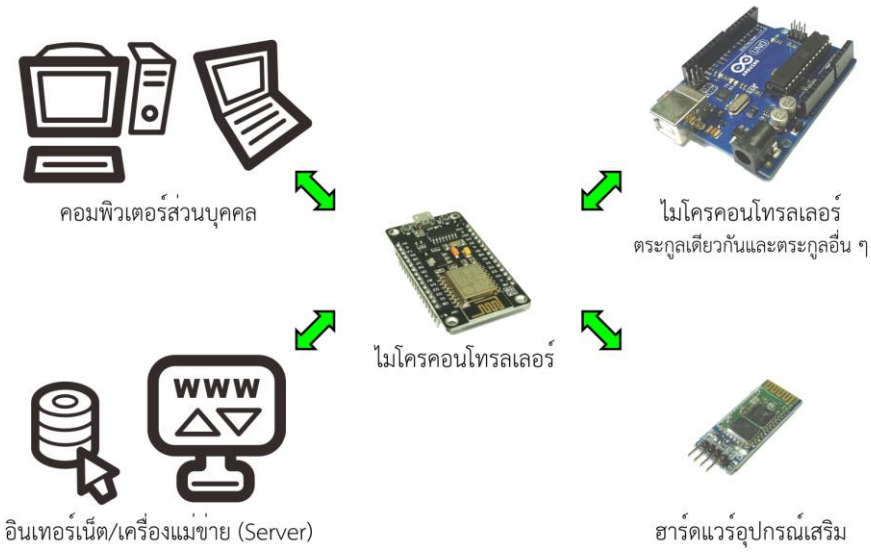
บทที่ 5

การสื่อสารผ่านพอร์ตอนุกรม

5.1 มาตรฐานการสื่อสาร

ในปัจจุบันมีการนำไมโครคอนโทรลเลอร์ไปเชื่อมต่อกับอุปกรณ์ต่าง ๆ มากมาย (ดูรูปที่ 5.1) โดยการเชื่อมต่อนั้นก็มีอยู่ด้วยกันหลายรูปแบบ โดยการเชื่อมต่อแบบหนึ่งที่ได้รับความนิยมก็คือการเชื่อมต่อแบบอนุกรม (Serial Communication) การทำความเข้าใจเกี่ยวกับการสื่อสารข้อมูลแบบอนุกรม เราจำเป็นต้องทราบศัพท์ที่เกี่ยวข้องดังนี้

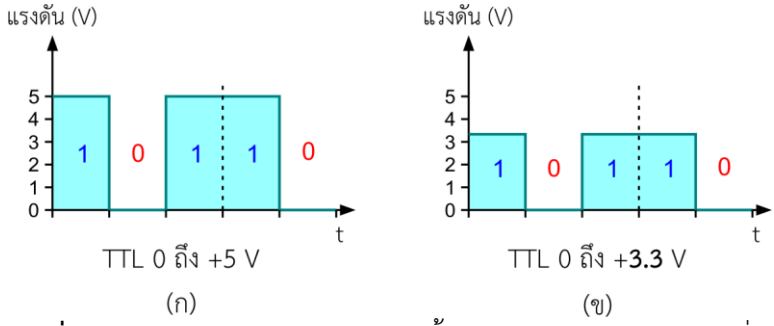
- TTL (Transistor-Transistor Logic)
- UART (Universal Asynchronous Receiver Transmitter)
- USB (Universal Serial Bus)



รูปที่ 5.1 ตัวอย่างการสื่อสารแบบอนุกรมระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์อื่น ๆ

ระดับสัญญาณ TTL (Transistor-Transistor Logic)

TTL เป็นระดับแรงดันที่ถูกกำหนดขึ้นในยุคแรก ๆ ของการเชื่อมต่อสื่อสารเพื่อใช้ในการกำหนดการเชื่อมต่อระหว่างขาของทรานซิสเตอร์ที่อยู่ภายในไอซี (วงจรรวม, Integrated Circuit, IC) ระดับสัญญาณ TTL จะใช้ระดับแรงดัน อยู่ที่ 0 ถึง +5 V (รูปที่ 5.2) แต่ในปัจจุบันมีอุปกรณ์หลายอย่างและไอซีจำนวนมากที่ทำงานในช่วงระดับแรงดัน 0 ถึง 3.3 V ซึ่งเรียกแรงดันระดับนี้ว่าระดับแรงดันต่ำ (Low Voltage TTL, LVTTL) ซึ่งผู้ใช้ควรตรวจสอบจากแผ่นข้อมูล (Datasheet) ของอุปกรณ์ที่จะนำมาใช้เสียก่อนว่าใช้กับระดับแรงดันใด เพราะหากใช้ผิดประเภทจะทำให้อุปกรณ์เสียหาย ดังนั้นสำหรับ NodeMCU ที่ใช้ระดับแรงดัน 3.3 V ในการส่งข้อมูล เราจะต้องตรวจสอบอุปกรณ์ที่นำมาต่อพ่วงว่าใช้กับระดับแรงดัน 3.3 V ได้หรือไม่



รูปที่ 5.2 ระดับแรงดัน TTL (ก) แบบดั้งเดิม และ (ข) แบบแรงดันต่ำ

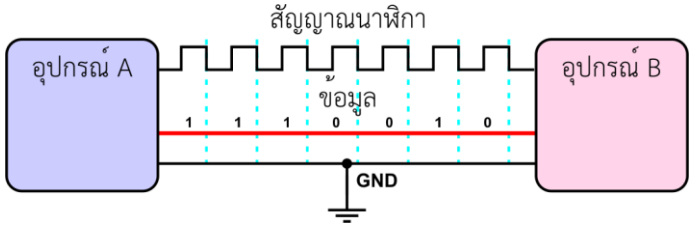
UART (Universal Asynchronous Receiver-Transmitter)

UART หมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส (asynchronous) ซึ่งมักใช้กับการสื่อสารแบบอนุกรม โดยการสื่อสารแบบอนุกรมอาจจะแบ่งได้เป็น 2 แบบ คือ

1) ซิงโครนัส (Synchronous)

เป็นการส่งข้อมูลเป็นบล็อก ครั้งละหลายไบต์ โดยจะมีสัญญาณนาฬิกา (CLK, CLocK) ที่ช่วยในการทำงานของตัวส่งและตัวรับสอดคล้องกัน โดยสัญญาณที่ส่งอาจจะถูก

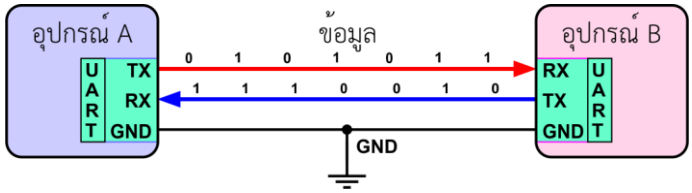
เข้ารหัสร่วมกันอยู่ในชุดข้อมูลนั้นหรือแยกอิสระออกเป็นสายต่างหากก็ได้ รูปที่ 5.3 แสดงลักษณะแนวคิดพื้นฐานในการส่งข้อมูลแบบซิงโครนัส



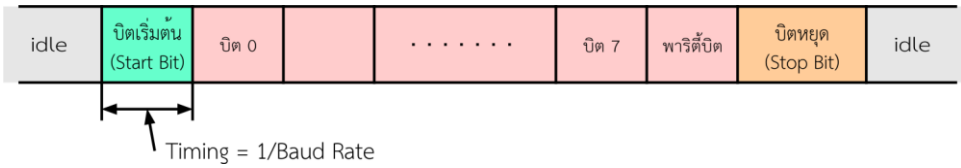
รูปที่ 5.3 การสื่อสารแบบซิงโครนัส

2) อะซิงโครนัส (Asynchronous)

การส่งข้อมูลแบบอะซิงโครนัสเป็นการสื่อสารแบบที่ใช้มากในการสื่อสารในไมโครคอนโทรลเลอร์ รูปแบบการสื่อสารจะเป็นการรับส่งข้อมูลครั้งละ 1 ไบต์ ดังรูปที่ 5.4 โดยมีรูปแบบของข้อมูลดังแสดงในรูปที่ 5.5



รูปที่ 5.4 การสื่อสารแบบอะซิงโครนัส



รูปที่ 5.5 รูปแบบข้อมูลที่ใช้ในการสื่อสารอนุกรมแบบอะซิงโครนัส

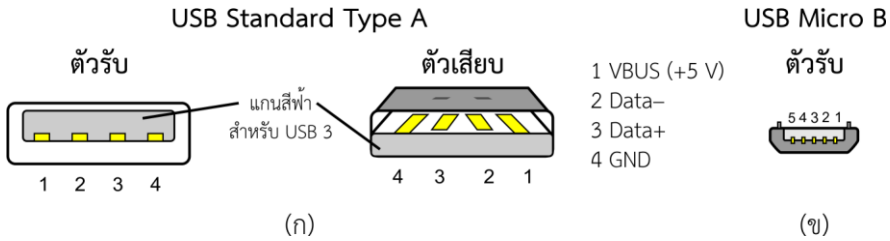
ในรูปที่ 5.5 ข้อมูลจะประกอบด้วย

- บิตเริ่มต้น (Start Bit) บอกจุดเริ่มต้นข้อมูล มีขนาด 1 บิต
- บิตข้อมูล (Data Bit) คือค่าข้อมูลมีได้ระหว่าง 5 ถึง 8 บิต
- พาริตี้บิต (Parity Bit) คือบิตที่ใช้สำหรับตรวจสอบความผิดพลาดของข้อมูล อาจไม่มีหรือมี 1 บิต
- บิตหยุด (Stop Bit) บิตที่บอกจุดสิ้นสุดข้อมูล มีได้ 1, 1.5 หรือ 2 บิต

USB (Universal Serial Bus)

USB เป็นมาตรฐานการเชื่อมต่อสำหรับการรับ-ส่งข้อมูลแบบอนุกรมสำหรับคอมพิวเตอร์และอุปกรณ์ต่อพ่วงต่าง ๆ ในระยะสั้น ๆ ไม่เกิน 2-5 เมตร โดย USB ได้ถูกเริ่มนำมาใช้ครั้งแรกในปี ค.ศ. 1996 (เรียก USB 1.0) และได้มีการพัฒนาต่อเนื่องมา โดยเน้นการพัฒนาด้านความเร็วในการส่งข้อมูล ปัจจุบัน มาตรฐาน USB ที่ใช้อย่างแพร่หลาย จะเป็น USB 2.0 และ USB 3.x (x = 0, 1 หรือ 2) โดย USB 2 จะมีความเร็วสูงสุดในการส่งข้อมูลอยู่ที่ 480 Mbit/s และพอร์ต USB 3 ถูกออกแบบมาให้มีลักษณะหน้าตาคล้ายกับ USB 2 เพื่อให้เราสามารถใช้งานอุปกรณ์ที่ใช้ USB 2 ร่วมกับพอร์ต USB 3 ได้ โดย USB 3 มีความเร็วสูงสุดสูงถึง 10 Gbit/s (USB 3.2) และเราสามารถสังเกตพอร์ต USB 3 ได้ง่าย จากสีของแกนพลาสติกภายในคือ USB 3.0 จะใช้แกนด้านในสีฟ้า รูปที่ 5.6 แสดงภาพหัวต่อ USB 2 แบบมาตรฐาน Type A และแบบไมโคร B ซึ่งพอร์ตแบบไมโคร B นี้ ถูกนำมาใช้ในบอร์ด NodeMCU

สาย USB ทั่วไป จะประกอบด้วยสาย 4 เส้น คือ แรงดันไฟตรง +5 V, Data–, Data+ และ กราวด์ โดยข้อมูลจะส่งแบบอนุกรมผ่านสาย Data– และ Data+ เท่านั้น สำหรับแรงดันไฟตรง +5 V จะสามารถนำมาใช้ส่งพลังงานไฟฟ้าได้ด้วย โดยส่งกระแสได้สูงสุด 0.5 A สำหรับ USB 2 และ 0.9 A สำหรับ USB 3 ดังนั้นในการใช้งานพอร์ต USB เป็นแหล่งพลังงานให้กับไมโครคอนโทรลเลอร์ ผู้ใช้จะต้องคำนึงถึงข้อจำกัดด้านการจ่ายกระแสนี้ด้วย



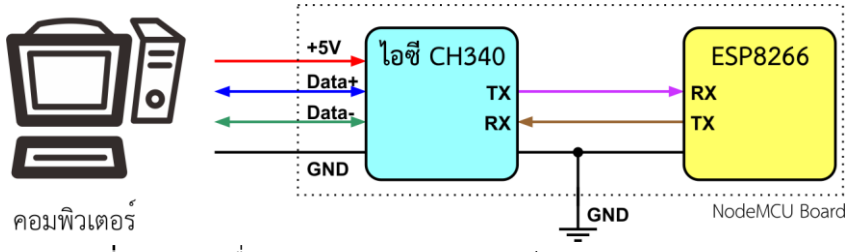
รูปที่ 5.6 ลักษณะของพอร์ต USB แบบ (ก) มาตรฐาน Type A และ (ข) ตัวรับแบบไมโคร B

อัตราการส่งข้อมูล

อัตราการส่งข้อมูลหรือ บอดเรท (Baud Rate) คือความเร็วของการรับ-ส่งข้อมูล เป็นจำนวนบิตต่อวินาที (bit per second, bps) เช่น 300, 1200, 2400, 4800, **9600**, 19200, 38400, 57600, 74880, **115200** bps เป็นต้น การเลือกอัตราการส่งข้อมูลขึ้นอยู่กับ ชนิดของสายสัญญาณ, ระยะทาง และปริมาณสัญญาณรบกวน โดยการส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์และอุปกรณ์อื่น ๆ นั้น จะทำได้ก็ต่อเมื่อเรากำหนดให้อัตราการส่งข้อมูลมีค่าตรงกันเท่านั้น

ไอซีเบอร์ CH340

ในการแปลงข้อมูลที่รับ-ส่งผ่านพอร์ต USB นั้น NodeMCU (ESP8266) จะใช้ไอซีเบอร์ CH340 ในการแปลงระดับสัญญาณแรงดัน ซึ่งบางครั้งจะเรียกว่า USB-UART Interface IC โดยในการใช้งานชิปนี้ เราอาจต้องลงไดรเวอร์ให้กับระบบปฏิบัติการวินโดว์ก่อน



รูปที่ 5.7 การเชื่อมต่อระหว่างคอมพิวเตอร์ (ระดับสัญญาณ USB) และ NodeMCU (ระดับสัญญาณ LVTTL) ผ่าน CH340

5.2 การเขียนโปรแกรมติดต่อสื่อสาร

ในการสื่อสารแบบอนุกรม ระหว่าง NodeMCU และคอมพิวเตอร์ เราสามารถทำได้โดยการเชื่อมต่อผ่านพอร์ต USB ดังที่เราได้ทดลองกันไปแล้วในบทที่ 3 และ 4 สำหรับฟังก์ชันในภาษา Arduino ที่ใช้ในการสื่อสารแบบอนุกรมนี้ จะอยู่ภายในวัตถุชื่อ Serial และมีดังนี้

- ฟังก์ชันกำหนด Buad rate

  - Serial.begin(int baud)

  โดย baud คือ อัตราเร็วที่ใช้ในการสื่อสาร มีค่าที่ใช้บ่อยคือ 9600 และ 115200

- ฟังก์ชันตรวจสอบข้อมูลที่ส่งมายังพอร์ตอนุกรม

  - Serial.available()

  ฟังก์ชันนี้จะส่งค่าคืนเป็นจำนวนไบต์ของข้อมูล (เป็นค่าจำนวนเต็ม) ที่ส่งผ่านพอร์ตอนุกรมมายังด้านรับ

- ฟังก์ชันอ่านค่าจากพอร์ตอนุกรม

  - Serial.read()

  ฟังก์ชันนี้จะส่งค่าคืนเป็นข้อมูลที่ส่งมายังด้านรับ

- ฟังก์ชันส่งข้อมูลออกไปยังพอร์ตอนุกรม

  - Serial.print(data), Serial.write(data) และ Serial.println(data)

  ฟังก์ชันนี้จะส่งค่าให้แก่พอร์ตอนุกรมโดย Serial.print จะส่งรหัสตัวอักษรแบบแอสกี (ASCII) ซึ่งสามารถอ่านทำความเข้าใจได้ง่าย Serial.write มักจะใช้ส่งค่าไบนารี และ Serial.println จะเหมือนกันกับ Serial.print แต่จะมีการเพิ่มตัวอักขระขึ้นบรรทัดใหม่ (newline character, '\n') หลังส่งค่าตัวอักษรตามคำสั่งนี้เสร็จ

5.3 การทดลองสื่อสารผ่านพอร์ตอนุกรม

วัตถุประสงค์

1. สามารถเขียนโปรแกรมให้ NodeMCU ส่งค่าไปแสดงผลบนคอมพิวเตอร์ได้
2. สามารถเขียนโปรแกรมให้ NodeMCU รับค่าจากคอมพิวเตอร์มาควบคุมรีเลย์ได้

อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดรีเลย์ชนิด 4 ช่อง	1 บอร์ด
5.	สาย USB	1 เส้น
6.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	6 เส้น

วิธีการทดลอง

ตอนที่ 1 การเขียนโปรแกรมเพื่อแสดงผลผ่านพอร์ตอนุกรม

1. เขียนโค้ดโปรแกรมแสดงข้อความผ่านพอร์ตอนุกรมดังแสดงด้านล่างนี้ จากนั้นจึงอัปโหลดลงสู่บอร์ด NodeMCU โดยเมื่อรันโค้ดโปรแกรมนี้ NodeMCU จะส่งค่าที่คำนวณได้ ไปแสดงผลยังพอร์ตอนุกรมของคอมพิวเตอร์ทุก ๆ 2 วินาที

```
1 // Basic Serial Communication with NodeMCU ESP8266
2 // Temperature Conversion
3
4 float Celsius = 23.32;
5 float Fahrenheit = 0;
6
7 void setup() {
8     Serial.begin(9600);
9 }
```

```
10
11 void loop() {
12     Serial.print("Temperature");
13     Serial.print("\t");
14     Serial.print(Celsius);
15     Serial.print(" C   -> \t");
16     Serial.print((9/5)*Celsius+32);
17     Serial.println(" F");
18     delay(2000);
19 }
20
```

2. เปิด Serial Monitor (ในเมนู Tools) เพื่อสังเกตผลลัพธ์ที่ได้

3. หากลองคำนวณค่าอุณหภูมิที่แสดงด้วยสมการในบรรทัดที่ 16 ด้วยเครื่องคิดเลข จะพบว่าค่าที่ได้ไม่ตรงกับค่าที่แสดงด้วยคอมพิวเตอร์ นั่นคือ การคำนวณด้วยโปรแกรมมีข้อผิดพลาดหรือบัค (bug) อยู่ ขอให้แก้ไขโดยการเปลี่ยนชนิดตัวเลขที่ใช้คำนวณเป็นเลขทศนิยมโดยการเพิ่มจุดทศนิยมและศูนย์ลงในโค้ดที่คำนวณ คือ แก้ไขโค้ดบรรทัดที่ 16 เป็น

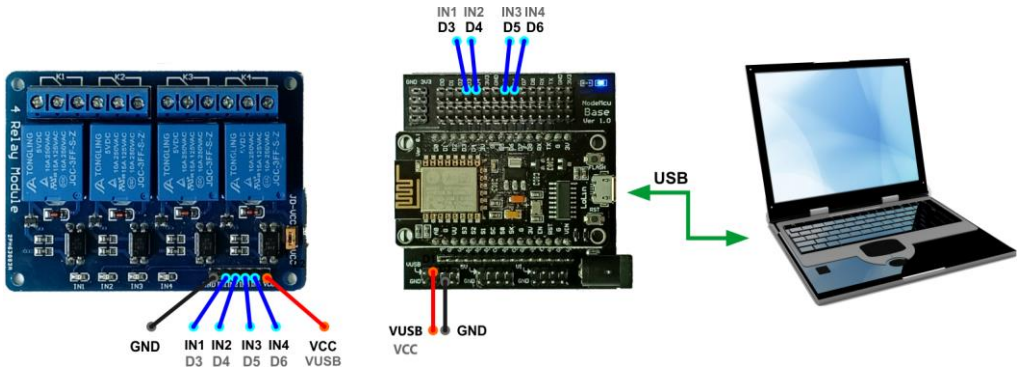
```
16     Serial.print((9.0/5.0)*Celsius+32);
```

จากนั้นจึงอัปโหลดใหม่แล้วสังเกตผลลัพธ์ที่ได้

ตอนที่ 2 การเขียนโปรแกรมเพื่อควบคุมรีเลย์ผ่านพอร์ตอนุกรม

ในการทดลองตอนนี้ เป็นการใช้คีย์บอร์ดควบคุมการทำงานของรีเลย์ผ่านการส่งข้อมูลผ่านพอร์ตอนุกรม โดยการกำหนดให้กดปุ่มบนคีย์บอร์ด เพื่อให้รีเลย์เปิดหรือปิดวงจร

1. ต่อวงจรดังรูปที่ 5.9



รูปที่ 5.9 วงจรควบคุมการทำงานอุปกรณ์ไฟฟ้าผ่านการสื่อสารแบบอนุกรม

2. เขียนโค้ดโปรแกรมควบคุมการทำงานของรีเลย์ทั้ง 4 ตัว

```
// Control Relays via Serial with NodeMCU ESP8266

#define ON LOW
#define OFF HIGH

char buff;

int Relay1 = D3;
int Relay2 = D4;
int Relay3 = D5;
int Relay4 = D6;

void setup() {
    Serial.begin(9600);
    pinMode(Relay1, OUTPUT);
    pinMode(Relay2, OUTPUT);
    pinMode(Relay3, OUTPUT);
    pinMode(Relay4, OUTPUT);
}

void loop() {
    buff = Serial.read();

    if(buff == '1')    {
        digitalWrite(Relay1, ON);
    }

```

```
28      if(buff == '5')     {
29          digitalWrite(Relay1, OFF);
30      }
31
32      if(buff == '2')     {
33          digitalWrite(Relay2, ON);
34      }
35
36      if(buff == '6')     {
37          digitalWrite(Relay2, OFF);
38      }
39
40      if(buff == '3')     {
41          digitalWrite(Relay3, ON);
42      }
43
44      if(buff == '7')     {
45          digitalWrite(Relay3, OFF);
46      }
47
48      if(buff == '4')     {
49          digitalWrite(Relay4, ON);
50      }
51
52      if(buff == '8')     {
53          digitalWrite(Relay4, OFF);
54      }
55  }
56
```

3. ทดสอบการควบคุม โดยการป้อนข้อมูลจากคีย์บอร์ดเข้าทางพอร์ตอนุกรม

แบบฝึกหัดท้ายการทดลอง

จงเขียนโปรแกรมเพิ่มเติมจากโปรแกรมในการทดลองตอนที่ 2 โดยกำหนดเงื่อนไขว่า หากเรากดปุ่ม ‘0’ บนคีย์บอร์ดแล้วรีเลย์ทุกตัวจะ OFF และหากเรากดปุ่ม ‘9’ แล้วรีเลย์ทุกตัวจะ ON

บทที่ 6

การแสดงผลด้วยจอแอลอีดีขนาด 128x64 พิกเซล

6.1 จอแอลอีดีและการติดตั้งไลบรารี

จอแสดงผล OLED (Organic Light-Emitting Diode) display เป็นจอแสดงผลที่สร้างจากวัสดุสารกึ่งตัวนำอินทรีย์ (Organic Semiconductor) ที่มีลักษณะเป็นชั้นสารกึ่ง-ตัวนำบาง ๆ อยู่ระหว่างขั้วบวก (Anode) และขั้วลบ (Cathode) และสามารถเปล่งแสงได้เมื่อมีกระแสไฟฟ้าไหลผ่าน การเปล่งแสงนี้เรียกว่ากระบวนการ อิเล็คโทรลูมิเนสเซนส์ (Electroluminescence) จอแสดงผล OLED มีข้อดีซึ่งแตกต่างจากจอแสดงผล LCD (Liquid Crystal Display) ทั่วไปคือ จอ OLED ไม่มีวงจรที่สร้างแสง Backlight จึงทำให้จอมีความหนาน้อยกว่าและเบากว่า ใช้กำลังไฟฟ้าต่ำ นอกจากนั้นจะไม่มีการเปล่งแสงในบริเวณที่ต้องการให้เป็นสีดำ ในปัจจุบันอุปกรณ์หลายอย่างเช่น โทรทัศน์, สมาร์ทโฟน (Smartphones) และ แท็บเล็ต (Tablets) ได้เริ่มเปลี่ยนไปใช้จอแสดงผล OLED กันมากขึ้น ในการทดลองนี้จะกล่าวถึง การทดลองใช้งานโมดูลจอแสดงผลกราฟิกแบบ OLED ขนาดเล็ก ซึ่งมีขนาด 128x64 พิกเซล ให้แสงเพียงสีเดียว (Monochrome) โดยทดลองเขียนโค้ดและใช้งานร่วมกับบอร์ด NodeMCU v.3 และใช้วิธีเชื่อมต่อแบบบัส I2C (หรือ I^2C, Inter-Integrated Circuit, อ่านว่า ไอสแควร์ซี)

ลักษณะโครงสร้างโมดูล OLED 128x64 พิกเซล

โมดูล OLED ที่ได้เลือกมาทดลองใช้งานนี้ มีขนาด 1.3" (วัดตามเส้นทแยงมุม) มีความละเอียด 128x64 พิกเซล สามารถแสดงผลได้แบบสีเดียว ซึ่งถือว่าเป็นจอขนาดเล็ก และใช้แรงดันไฟเลี้ยง +3.3 V ได้ จอขนาดเล็กนี้เหมาะสำหรับนำไปใช้กับระบบสมองกลฝังตัว ที่

ต้องการส่วนแสดงผลกราฟิกบนพื้นที่จำกัด ภายในโมดูลจะมีชิป SH1106 เป็นตัวควบคุมการทำงาน สามารถเชื่อมต่อแบบอนุกรมโดยใช้บัส SPI (Serial Peripheral Interface) หรือ I2C ถ้าต้องการแสดงข้อความจะต้องมีข้อมูลของฟอนต์ตัวอักษร แต่ละตัว ซึ่งจะมีลักษณะเป็นข้อมูลภาพแบบบิตแมปเก็บไว้ในหน่วยความจำ

การจัดการหน่วยความจำภายในของ SH1106 ที่เรียกว่า Graphic Display Data RAM (GDDRAM) สำหรับขนาด 128x64 พิกเซล แบ่งเป็นคอลัมน์ (Column) และ เพจ (Page) ซึ่งมีทั้งหมด 128 คอลัมน์ (หมายเลข 0..127) และมีทั้งหมด 8 เพจ (หมายเลข 0,1, 2...7) แต่ละเพจ จะประกอบด้วย 8 บรรทัด ดังนั้นจึงมี 8x8 = 64 บรรทัดหรือแถว (rows) การแสดงผลของจอภาพขึ้นอยู่กับค่าบิต (0 หรือ 1) สำหรับแต่ละพิกเซลที่ได้เขียนข้อมูลลงในหน่วยความจำ GDDRAM

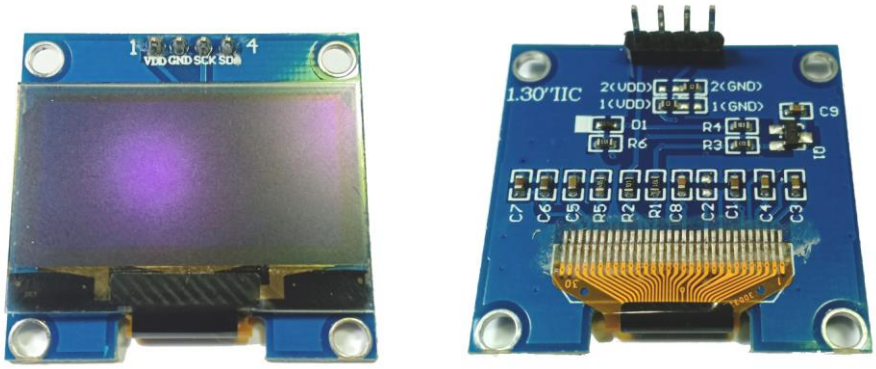
การสื่อสารข้อมูลกับชิป SH1106 ทำได้โดยผ่านบัส SPI หรือ I2C โดยต้องส่งคำสั่งเพื่อกำหนดค่าต่าง ๆ ในการทำงาน เช่น คำสั่งเปิด/ปิดการแสดงผล (turn on/off display) คำสั่งเลือกโหมดการเข้าถึงหน่วยความจำ (memory addressing mode) คำสั่งกำหนดค่าตำแหน่งเริ่มต้น (Start Address) และตำแหน่งสุดท้าย (End Address) สำหรับคอลัมน์และเพจ และการเขียนข้อมูล (Data) เป็นการเขียนข้อมูลลงใน GDDRAM

การเชื่อมต่อขาของโมดูล OLED แบบ I2C

รูปที่ 6.1 แสดงรูปร่างของโมดูล OLED ที่ใช้การเชื่อมต่อแบบ I2C โดยมีขาที่ต้องเชื่อมต่อดังนี้

- VDD (หรือ VCC) เป็นขาสำหรับรับแรงดันไฟเลี้ยง +3.3 V
- GND เป็นขากราวด์ (Ground)
- SCK เป็นขา I/O สำหรับสัญญาณ **S**erial **C**loc**K** (หรือ SCL (**S**erial **CL**ock)
- SDA เป็นขา I/O สำหรับสัญญาณ SDA (**S**erial **DA**ta) สำหรับการส่งข้อมูล

หมายเหตุ OLED บางรุ่น อาจมีการเรียงตำแหน่งขา สลับกันระหว่าง SCK และ SDA ดังนั้นขอให้ผู้ใช้ศึกษาแผ่นข้อมูลของอุปกรณ์ก่อนทำการทดลอง

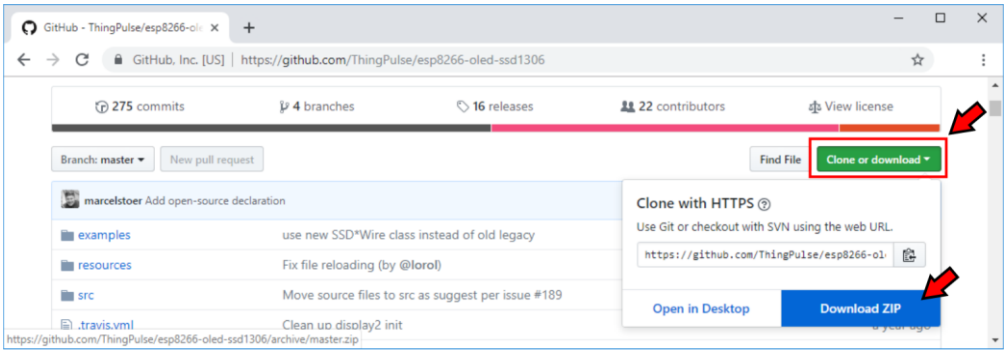


(ก) (ข)

รูปที่ 6.1 รูปร่างของโมดูล OLED (ก) ด้านหน้า และ (ข) ด้านหลัง

การติดตั้งไลบรารี OLED I2C

เนื่องจากโปรแกรม Arduino IDE ที่ติดตั้งมาในตอนเริ่มต้นไม่มีไลบรารีของ OLED นี้ เราจึงจำเป็นต้องมีการติดตั้งเพิ่มเติมเข้าไปในโปรแกรม ซึ่งจะต้องใช้ไลบรารี สามารถดาวน์โหลดจากลิงค์ https://github.com/ThingPulse/esp8266-oled-ssd1306 ดังรูปที่ 6.2 โดยให้ทำการคัดลอกโฟลเดอร์ที่โหลดมาไปเก็บไว้ที่ **C:\Arduino18\libraries** ดังรูปที่ 6.3 เมื่อคัดลอกลงมาแล้ว Arduino IDE จะสามารถเรียกใช้ฟังก์ชันสำหรับสั่งงาน OLED นี้ได้ โดยเราจะต้องเพิ่มคำสั่งเรียกไลบรารี คือ #include <Wire.h> และ #include <SH1106.h> เข้าไปที่ด้านบนของโค้ด (โดยฟังก์ชันใน Wire.h จะถูกเรียกใช้โดยฟังก์ชันใน SH1106.h อีกทีหนึ่ง)



รูปที่ 6.2 การดาวน์โหลดไลบรารีของ OLED จาก github

รูปที่ 6.3 การติดตั้งไลบรารีของ OLED ลงใน Arduino IDE

6.2 คำสั่งพื้นฐานควบคุมการแสดงผลของจอ OLED

1. คำสั่งกำหนดตำแหน่งขาที่ต่อใช้งานกับจอ OLED แบบ I2C

SH1106 display(0x3c, D1, D2);

เป็นการกำหนด Instance ของวัตถุชื่อ display และการกำหนดหน่วยความจำตำแหน่ง 0x3c ให้กับอุปกรณ์แสดงผลนี้ โดยต่อขา D1 กับ SDA และขา D2 ต่อกับ SCK

2. คำสั่งตรวจสอบการเชื่อมต่อกับจอและจัดการหน่วยความจำ

display.init();

เป็นคำสั่งที่ใช้ก่อนการเริ่มต้นให้จอแสดงผล (เรียกใช้ในฟังก์ชัน setup()) โดยวัตถุชื่อ display จะต้องถูกสร้างไว้ก่อนด้วยคำสั่งในข้อ 1 และอาจใช้ชื่ออื่นก็ได้

3. การสั่งให้แสดงผลบนหน้าจอ OLED

display.display();

เป็นฟังก์ชันที่ใช้สั่งให้จอแสดงข้อมูลที่อยู่ในหน่วยความจำ โดย display ตัวแรกคือชื่อวัตถุที่ตั้งในข้อ 1 (อาจตั้งเป็นชื่ออื่นก็ได้) และตัวที่สองคือชื่อฟังก์ชัน/เมธอด

4. การลบการแสดงผลบนหน้าจอ

display.clear();

สำหรับข้อ 5-7 จะขอละคำว่า display ไว้

5. การแสดงตัวอักษร เราจะเริ่มจากการกำหนดฟอนต์ด้วยคำสั่ง

setFont(fontData);

โดย fontData ที่รองรับคือ “ArialMT_Plain_10”, “ArialMT_Plain_16” และ ArialMT_Plain_24 แล้วจากนั้นจึงกำหนดตำแหน่ง (x,y) ของข้อความและข้อความ (String) ด้วยคำสั่ง

drawString(int x, int y, String);

6. การแสดงรูปเราขาคณิตพื้นฐานต่าง ๆ จะมีคำสั่งที่สามารถเลือกใช้ได้ ได้แก่

6.1 คำสั่งวาดเส้นตรง

6.1.1 เส้นตรง drawLine(x1, y1, x2, y2)

6.1.2 เส้นตามนอน drawHorizontalLine(x, y, length)

6.1.3 เส้นตามตั้ง drawVerticalLine(x, y, length)

6.2 คำสั่งวาดรูปสี่เหลี่ยม

6.2.1 สี่เหลี่ยมเปิด drawRect(x, y, width, height)

6.2.2 สี่เหลี่ยมทึบ fillRect(x, y, width, height)

6.3 คำสั่งวาดรูปวงกลม

6.3.1 วงกลมเปิด drawCircle(x, y, radius)

6.3.2 วงกลมทึบ fillCircle(x, y, radius)

โดยตัวแปรทุกตัวที่ส่งค่าไปจะต้องเป็นจำนวนเต็มค่าบวก (unsigned integer, uint) เท่านั้น

7. การวาดภาพแบบพิกเซล

7.1 คำสั่งวาดแถบ Progress Bar

drawProgressBar(x, y, width, height, progress)

โดยตัวแปร progress จะมีค่าได้ระหว่าง 0 ถึง 100

7.2 คำสั่งวาดภาพบิตแมป

drawXbm(x, y, width, height, picture);

โดยตัวแปร picture คือภาพที่เก็บเป็นอาเรย์ของรหัสแอสกี (ชนิด char)

6.3 การทดลองแสดงผลบนจอ OLED

วัตถุประสงค์

1. สามารถต่อวงจรไมโครคอนโทรลเลอร์ NodeMCU v.3 กับจอ OLED เพื่อแสดงผลได้
2. สามารถเขียนโปรแกรมแสดงข้อความและข้อมูลชนิดต่าง ๆ บนจอ OLED ได้อย่างถูกต้อง

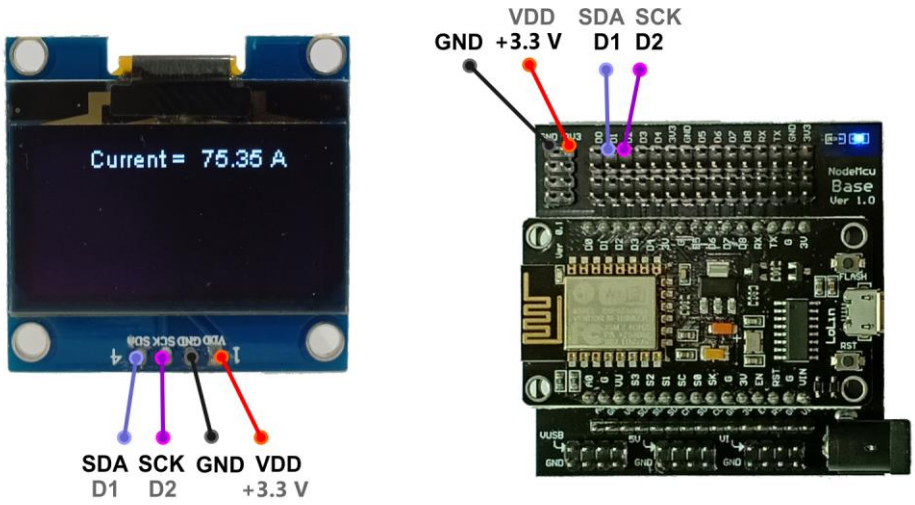
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดโมดูล OLED Display ขนาด 128×64 พิกเซล	1 บอร์ด
5.	สาย USB	1 เส้น
6.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	4 เส้น

วิธีการทดลอง

ตอนที่ 1

1. ตอวงจรตามรูปที่ 6.4 โดยให้ต่อขา D1 เข้ากับขา SDA ของโมดูล OLED, ขา D2 เข้ากับขา SCK (หรือ SCL), ต่อขาไฟเลี้ยง +3.3 V (3V3) ให้แก่โมดูล OLED ที่ขา VDD (หรือ VCC) และต่อขากราวด์ (GND) ร่วมกัน
2. เขียนโปรแกรมตามโค้ดที่แสดงในหน้าถัดไป



รูปที่ 6.4 การเชื่อมต่อ NodeMCU v.3 กับจอแสดงผล OLED แบบ I2C

```
// Control OLED by NodeMCU ESP8266

#include <Wire.h>
#include <SH1106.h>

//SSD1306 display(Addr,SDA,SCL);
SH1106 display(0x3c, D1, D2);

void setup() {
    display.init();
}

void loop() {
    display.setFont(ArialMT_Plain_10);
    display.drawString(25, 0, "Current = ");
    display.drawString(75, 0, String(75.35)+" A");
    display.display();
    delay(200);
}
```

3. เมื่อเขียนโปรแกรมตามโค้ดแล้ว ให้ทำการอัปโหลด จอจะแสดงผล “Current = 75.35 A” ดังรูปที่ 6.4

ตอนที่ 2 เขียนโปรแกรมดังแสดงในโค้ดด้านล่างแล้วสังเกตผลลัพธ์ที่แสดงบนจอ

```
// Control OLED by NodeMCU ESP8266

#include <Wire.h>
#include <SH1106.h>

// SSD1306 display(Addr,SDA,SCL);
SH1106 display(0x3c, D1, D2);

void setup() {
    display.init();
}

void loop() {

    // Display a line
    display.drawVerticalLine(120, 5, 30);

    // Display a circle
    display.drawCircle(64, 32, 18);

    // Display a rectangle
    display.drawRect(0, 0, 128, 64);
    display.drawRect(0, 33, 100, 10);
    display.fillRect(0, 35, 45, 6);

    // Display a progress bar
    display.drawProgressBar(0, 48, 100, 10, 50);

    display.display();

    delay(200);
}

```

ผลลัพธ์ของโค้ดในตอนที่ นี้แสดงในรูปที่ 6.5(ก)

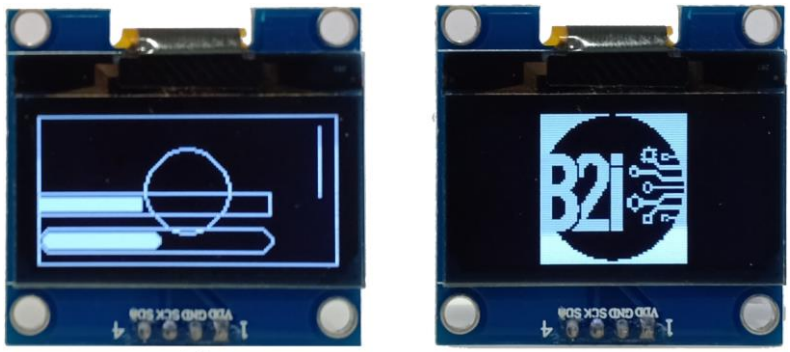
ตอนที่ 3 เขียนโปรแกรมดังแสดงในโค้ดด้านล่างแล้วสังเกตผลลัพธ์ที่แสดงบนจอ โดยมีข้อสังเกตคือ การเติม 0x หน้าตัวเลขเป็นการบ่งชี้ในภาษา C ว่าข้อมูลที่ตามมานั้นแสดงอยู่ในรูปเลขฐานสิบหก

```
// Control OLED by NodeMCU ESP8266

#include <Wire.h>
#include <SH1106.h>

// SSD1306 display(Addr, SDA, SCL);
SH1106 display(0x3c, D1, D2);

#define width 64
#define height 64

static unsigned char b2i[] = {
0xFF, 0xFF, 0xFF, 0x3F, 0xFE, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF,  0x0,  0x0, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x1F,  0x0,  0x0, 0xF8, 0xFF, 0xFF,
0xFF, 0xFF,  0x3,  0x0,  0x0, 0xE0, 0xFF, 0xFF,
0xFF, 0xFF,  0x0,  0x0,  0x0, 0x80, 0xFF, 0xFF,
0xFF, 0x7F,  0x0,  0x0,  0x0,  0x0, 0xFF, 0xFF,
0xFF, 0x1F,  0x0,  0x0,  0x0,  0x0, 0xFC, 0xFF,
0xFF,  0xF,  0x0,  0x0,  0x0,  0x0, 0xF8, 0xFF,
0xFF,  0x7,  0x0,  0x0,  0x0,  0x0, 0xF0, 0xFF,
0xFF,  0x3,  0x0,  0x0,  0x0,  0x0, 0xE0, 0xFF,
0xFF,  0x1,  0x0,  0x0,  0x0,  0x0, 0xC0, 0xFF,
0xFF,  0x0,  0x0,  0x0,  0x0,  0x0, 0x80, 0xFF,
0x7F,  0x0,  0x0,  0x0,  0x0,  0x0,  0x0, 0xFF,
0x3F,  0x0,  0x0,  0x0,  0x0,  0x0,  0x0, 0xFE,
0x3F,  0x0,  0x0,  0x0,  0x0, 0x80,  0x0, 0xFE,
0x1F,  0x0,  0x0,  0x0,  0x0, 0xE0,  0x7, 0xFC,
 0xF,  0x0,  0x0,  0x0,  0x0, 0x20,  0x2, 0xF8,
0xFF,  0x7, 0xFC, 0x83,  0xF, 0x30, 0xE6, 0xF8,
0xFF, 0x1F, 0xFE, 0x87,  0xF, 0x20, 0x22, 0xFF,
0xFF, 0x3F, 0xFF, 0x8F,  0xF, 0xF0, 0xE7, 0xF0,
0xFF, 0x7F, 0xFF, 0x9F,  0xF, 0xA0,  0x7, 0xE0,
0xFF, 0x7F, 0x9E, 0xBF,  0xF,  0x0,  0xC, 0xE0,
0x1F, 0xFC,  0xE, 0x3E,  0x0,  0x3, 0xF8, 0xFF,
0x1F, 0xFC,  0x6, 0xBE, 0x8F,  0x7, 0xF0, 0xFF,
0x1F, 0xF8,  0x6, 0xBE, 0x8F, 0xFC,  0x7, 0xC0,
```

```
0x1F, 0xF8,  0x6, 0xBE, 0x8F, 0xFC,  0xF, 0xC0,
0x1F, 0xF8,  0x6, 0xBE,  0xF,  0x7, 0x18, 0xC0,
0x1F, 0xF8,  0x0, 0xBE,  0xF,  0x0, 0xF0, 0xFF,
0x1F, 0xFC,  0x0, 0xBE,  0xF,  0x0, 0xE0, 0xFF,
0x1F, 0x7E,  0x0, 0x9F,  0xF, 0xC0,  0x0, 0x80,
0xFF, 0x7F,  0x0, 0x9F,  0xF, 0x20,  0x1, 0x80,
0xFF, 0x3F, 0x80, 0x9F,  0xF, 0x20, 0xFF, 0xFF,
0xFF, 0x1F, 0xC0, 0x8F,  0xF, 0x20,  0x1, 0x80,
0xFF, 0x3F, 0xC0, 0x87,  0xF, 0xC0,  0x0, 0x80,
0xFF, 0x7F, 0xE0, 0x87,  0xF,  0x2, 0xC0, 0xFF,
0x1F, 0xFE, 0xF0, 0x83,  0xF,  0x7, 0xE0, 0xFF,
0x1F, 0xFC, 0xF0, 0x81, 0xCF, 0x18, 0x30, 0xC0,
0x1F, 0xF8, 0xF8, 0x81, 0xCF, 0xF8, 0x1F, 0xC0,
0x1F, 0xF8, 0xFC, 0x80, 0xCF, 0xF8,  0xF, 0xC0,
0x1F, 0xF8, 0x7D, 0x80, 0xCF, 0x18, 0xF8, 0xFF,
0x1F, 0xF8, 0x7D, 0x80,  0xF,  0x7, 0xFC, 0xFF,
0x1F, 0xF8, 0x3D, 0x80,  0xF,  0x2,  0x6, 0xE0,
0x1F, 0xF8, 0x1C, 0x80,  0xF,  0x0,  0x3, 0xE0,
0x1F, 0xFC, 0x1C, 0x80,  0xF, 0xE0, 0xE1, 0xF0,
0x1F, 0xFE,  0xC, 0x80,  0xF, 0x20, 0x21, 0xFF,
0xFF, 0xFF, 0xFE, 0xBF,  0xF, 0x20, 0xA1, 0xF8,
0xFF, 0x7F, 0xFE, 0xBF,  0xF, 0xE0,  0x1, 0xF8,
0xFF, 0x3F, 0xFF, 0xBF,  0xF,  0x0,  0x0, 0xFC,
0xFF, 0x9F, 0xFF, 0xBF,  0xF,  0x0,  0x0, 0xFC,
0xFF, 0xC7, 0xFF, 0xBF, 0xF,  0x0,  0x0,  0xFE,
0x7F,  0x0,  0x0,  0x0,  0x0,  0x0,  0x0, 0xFF,
0xFF,  0x0,  0x0,  0x0,  0x0,  0x0, 0x80, 0xFF,
0xFF,  0x1,  0x0,  0x0,  0x0,  0x0, 0xC0, 0xFF,
0xFF,  0x1,  0x0,  0x0,  0x0,  0x0, 0xC0, 0xFF,
0xFF,  0x7,  0x0,  0x0,  0x0,  0x0, 0xF0, 0xFF,
0xFF,  0xF,  0x0,  0x0,  0x0,  0x0, 0xF8, 0xFF,
0xFF, 0x1F,  0x0,  0x0,  0x0,  0x0, 0xFC, 0xFF,
0xFF, 0x3F,  0x0,  0x0,  0x0,  0x0, 0xFE, 0xFF,
0xFF, 0xFF,  0x0,  0x0,  0x0, 0x80, 0xFF, 0xFF,
0xFF, 0xFF,  0x3,  0x0,  0x0, 0xE0, 0xFF, 0xFF,
0xFF, 0xFF,  0xF,  0x0,  0x0, 0xF8, 0xFF, 0xFF,
0xFF, 0xFF, 0x7F,  0x0,  0x0, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0x1F, 0xFC, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF  };

void setup() {
    display.init();
}

```

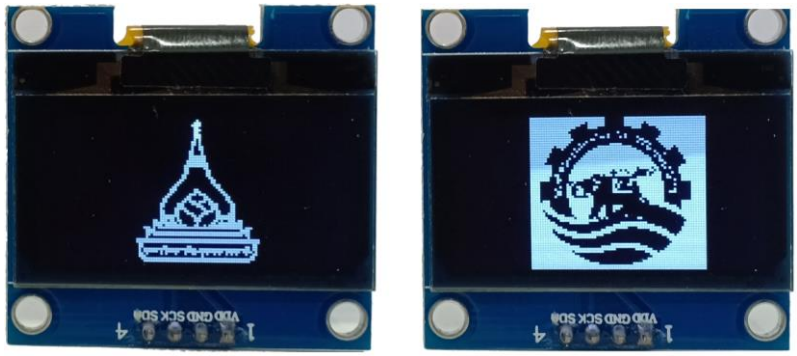
```
82 void loop() {
83     display.clear();
84     display.drawXbm(32, 0, width, height, b2i);
85     display.display();
86     delay(1000);
87 }
88
```

ผลลัพธ์ของโค้ดในตอนที่ นี้แสดงในรูปที่ 6.5(ข) ด้วยวิธีการแสดงผลแบบกราฟฟิกทำให้ผู้ใช้สามารถนำข้อมูลภาพใด ๆ มาแสดงก็ได้ ดังตัวอย่างในรูปที่ 6.6



(ก) (ข)

รูปที่ 6.5 ผลลัพธ์ที่ได้จากการรันโค้ดการทดลอง (ก) ตอนที่ 2 และ (ข) ตอนที่ 3



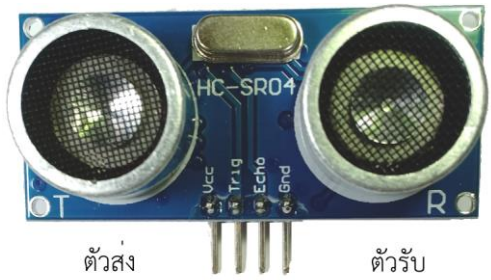
รูปที่ 6.6 ตัวอย่างภาพโลโก้ที่แสดงด้วยจอ OLED

บทที่ 7

การวัดระยะทางด้วยโมดูลอัลตราโซนิกส์

7.1 หลักการวัดระยะทางด้วยคลื่นอัลตราโซนิกส์

การวัดระยะทางด้วยคลื่นเหนือเสียงหรือคลื่นอัลตราโซนิกส์ ใช้หลักการของการสะท้อนคลื่น โดยโมดูลส่งและรับคลื่นอัลตราโซนิกส์ HC-SR04 (รูปที่ 7.1) เป็นโมดูลที่ใช้ในการวัดระยะทางโดยอาศัยหลักการสะท้อนของคลื่นอัลตราโซนิกส์ ที่ความถี่ประมาณ 40 kHz ในโมดูลนี้มีแหล่งกำเนิดคลื่นอัลตราโซนิกส์ส่งไปสะท้อนวัตถุแล้วสะท้อนกลับยังตัวรับสัญญาณ โดยระยะทางจะสัมพันธ์กับระยะเวลาที่คลื่นเดินทางตัวส่งไปสะท้อนวัตถุแล้วกลับมายังตัวรับ เมื่อได้เวลาในการเดินทางของคลื่นแล้วจึงนำมาคำนวณหาระยะทางระหว่างโมดูลกับวัตถุที่สะท้อนคลื่น



รูปที่ 5.1 โมดูลวัดระยะทางด้วยคลื่นอัลตราโซนิกส์ HC-SR04

หลักการทำงานของโมดูล HC-SR04

โมดูล HC-SR04 ทำงานที่แรงดันประมาณ 5 V โดยป้อนแรงดันแหล่งจ่ายให้ขา VCC และ GND โมดูลนี้มีขาสัญญาณดิจิทัล TRIG (อินพุต) และ ECHO (เอาต์พุต) ที่นำไปเชื่อมต่อ

กับไมโครคอนโทรลเลอร์ต่าง ๆ ในการวัดระยะห่างแต่ละครั้งโมดูลจะรับคำสั่งให้สร้างสัญญาณแบบพัลส์ที่มีความกว้าง (Pulse Width) อย่างน้อย 10 µsec ป้อนให้ขา TRIG และหลังจากนั้นให้วัดความกว้างของสัญญาณช่วง HIGH จากขา ECHO ถ้าวัตถุอยู่ใกล้ระยะเวลาที่จะได้รับสัญญาณพัลส์กลับมาจะมีค่าน้อย แต่ถ้าวัตถุอยู่ไกลออกไป ก็จะได้ค่าระยะเวลาที่สัญญาณพัลส์ใช้เดินทางมีค่ามากขึ้น การเลือกใช้งานโมดูลประเภทนี้ มีประเด็นที่สำคัญ เช่น ช่วงระยะห่างของวัตถุที่จะวัด ความกว้างของมุมเมื่อคลื่นความถี่เหนือเสียงเดินทางออกไปจากตัวส่ง (เรียกว่ามุมของลำคลื่นหรือ Beam Angle) นอกจากนั้น การสะท้อนกลับของคลื่นเสียงที่วัตถุกีดขวาง ขนาดและรูปทรงของวัตถุ และการสะท้อนกลับของเสียงจากหลายทิศทาง หรือต่างระยะกัน ก็มีผลต่อความถูกต้องในการวัดค่าระยะห่างได้เช่นกัน

ข้อมูลเชิงเทคนิคของโมดูล HC-SR04

- ใช้แรงดันประมาณ +5 V
- กินกระแสประมาณ 15 mA
- ช่วงการวัดระยะทาง (measurement range): ประมาณ 4 cm ถึง 4 m
- ความกว้างเชิงมุมในการวัด (measuring angle): 15 องศา
- ความกว้างของสัญญาณ Pulse สำหรับ Trigger: 10 µsec
- ระดับแรงดันลอจิกสำหรับขา TRIG และ ECHO คือ 5 V (TTL)

7.2 คำสั่งที่เกี่ยวข้องและไลบรารี Ultrasonic.h

ในการอ่านค่าระยะทางจากโมดูล HC-SR04 จะสามารถทำได้สองวิธีหลัก ๆ คือ การเขียนโค้ดควบคุมและคำนวณระยะทางด้วยตนเอง และการใช้ไลบรารี Ultrasonic.h ซึ่งในหัวข้อนี้ จะกล่าวถึงทั้งสองวิธี

การเขียนโค้ดควบคุมและคำนวณระยะทาง

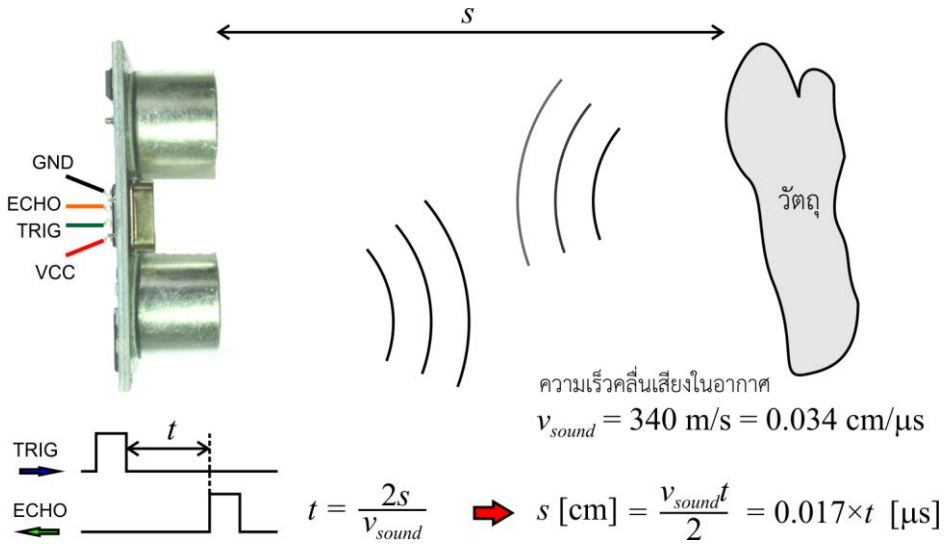
1. ให้ไมโครคอนโทรลเลอร์ส่งสัญญาณพัลส์ความกว้างอย่างน้อย 10 µsec ออกไปที่ขาเอาต์พุตที่ต่อกับขา TRIG ของโมดูล

2. วัดระยะเวลาที่คลื่นเดินทางโดยอ่านค่าจากสัญญาณพัลส์ที่เป็นอินพุตจากขา ECHO ของโมดูลโดยใช้คำสั่ง/ฟังก์ชัน pulseIn() ซึ่งเป็นฟังก์ชันภายในภาษา Arduino และจะได้ค่าเป็นจำนวนเต็ม (หน่วยเป็นไมโครวินาที)

3. นำค่าที่ได้มาคำนวณเป็นระยะทาง (โดยใช้ความเร็วคลื่นเสียง 340 m/s ประกอบ)

4. แสดงค่าของระยะทางที่คำนวณได้ แล้วเว้นระยะเวลาประมาณ 300 msec (0.3 วินาที) แล้วทำขั้นตอนซ้ำ

การคำนวณระยะทางสามารถทำได้ตามสูตรที่แสดงในรูปที่ 7.2



รูปที่ 7.2 การคำนวณค่าระยะทางจากค่าที่อ่านได้จากโมดูล HC-SR04

ฟังก์ชันหลักที่ใช้ในการคำนวณคือฟังก์ชัน pulseIn(pin, value) เป็นฟังก์ชันที่ใช้แสดงค่าความกว้างของพัลส์สัญญาณที่เข้ามาที่ขา pin โดยจะกำหนดพัลส์สัญญาณที่ค่า value (HIGH หรือ LOW)

pulseIn(pin, value, timeout) จะทำงานเหมือนฟังก์ชัน pulseIn ภายในช่วงเวลา timeout ที่กำหนด (ในหน่วยไมโครวินาที)

ตัวอย่างการใช้ฟังก์ชัน pulseIn

```
int pin = D0; // กำหนด pin ที่ต้องการอ่านค่าความกว้าง
unsigned long duration; // กำหนดตัวแปรที่เก็บค่าความกว้าง
void setup() {
    pinMode(pin, INPUT); // กำหนด pin ที่ใช้อ่านค่า เป็นอินพุต
}
void loop() {
    duration = pulseIn(pin, HIGH); //อ่านค่าความกว้างพัลส์
}
```

ฟังก์ชัน digitalWrite(pin, value) เป็นฟังก์ชันที่เขียนค่า HIGH หรือ LOW ไปที่ pin

ตัวอย่าง การสร้างสัญญาณ pulse ความกว้าง 12 µsec ไปที่ขาเอาต์พุต TRIG ของโมดูล HC-SR04

```
int TRIG = D1;    // กำหนด pin TRIG
                  // อาจเขียนเป็น #define TRIG D1 ก็ได้เช่นกัน
void setup() {
    pinMode(TRIG, OUTPUT); // กำหนด pin TRIG นี้เป็นเอาต์พุต
}

void loop() {
    digitalWrite(TRIG, HIGH); // เขียนค่า HIGH ไปที่ pin TRIG
    delayMicroseconds(12);    // หน่วงเวลา 12 ไมโครวินาที
    digitalWrite(TRIG, LOW);  // เขียนค่า LOW ไปที่ pin TRIG
}
```

การใช้ไลบรารี Ultrasonic.h

นอกจากการเขียนโปรแกรมเพื่อส่งพัลส์ อ่านค่าพัลส์ และ คำนวณระยะทางด้วยตนเองแล้ว เรายังสามารถใช้ไลบรารีสำเร็จรูปที่ชื่อ Ultrasonic.h มาอ่านค่าระยะทางได้ ไลบรารีนี้สามารถหาดาวน์โหลดได้ทั่วไป เช่น จาก https://github.com/hemalchevli/ultrasonic-library หลังจากโหลดแล้วก็คัดลอกลงในโฟลเดอร์ C:\Arduino18\libraries โดยค่าที่ได้จะมีหน่วยเป็นเซนติเมตรหรือนิ้วก็ได้ และมีรูปแบบการใช้คือ

```
#include <Ultrasonic.h>

// กำหนด instance ของวัตถุชื่อ ultrasonic
// กำหนดขา TRIG ต่อกับขา D6 และ ECHO ต่อกับขา D7

Ultrasonic ultrasonic(D6, D7);

// อ่านค่าระยะทางในหน่วย cm และเก็บไว้ในตัวแปรจำนวนเต็มชื่อ dist
int dist = ultrasonic.Ranging(CM);
```

7.3 การทดลองวัดระยะทาง

วัตถุประสงค์

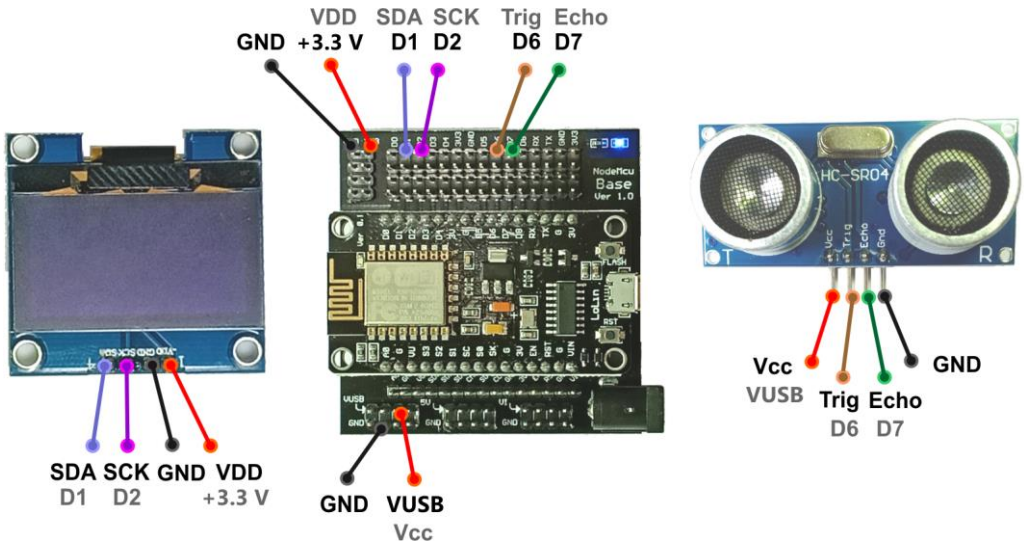
1. สามารถอธิบายหลักการวัดระยะทางด้วยคลื่นอัลตราโซนิกส์ได้
2. สามารถเขียนโปรแกรมให้ NodeMCU v.3 คำนวณค่าระยะทางได้อย่างถูกต้อง
3. สามารถเขียนโปรแกรมให้ NodeMCU v.3 แสดงค่าระยะทางบนจอ OLED และพอร์ตอนุกรมได้

อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดโมดูลอัลตราโซนิกส์ HC-SR04	1 บอร์ด
5.	บอร์ดโมดูล OLED Display ขนาด 128×64 พิกเซล	1 บอร์ด
6.	สาย USB	1 เส้น
7.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	8 เส้น

วิธีการทดลอง

1. ต่อวงจรดังรูปที่ 7.3



รูปที่ 7.3 การเชื่อมต่อ NodeMCU v.3 กับจอแสดงผล OLED และ โมดูล HC-SR04

2. เขียนโค้ดข้างล่างนี้แล้วอัปโหลดลง NodeMCU v.3 แล้วสังเกตสิ่งที่แสดงบนจอ OLED

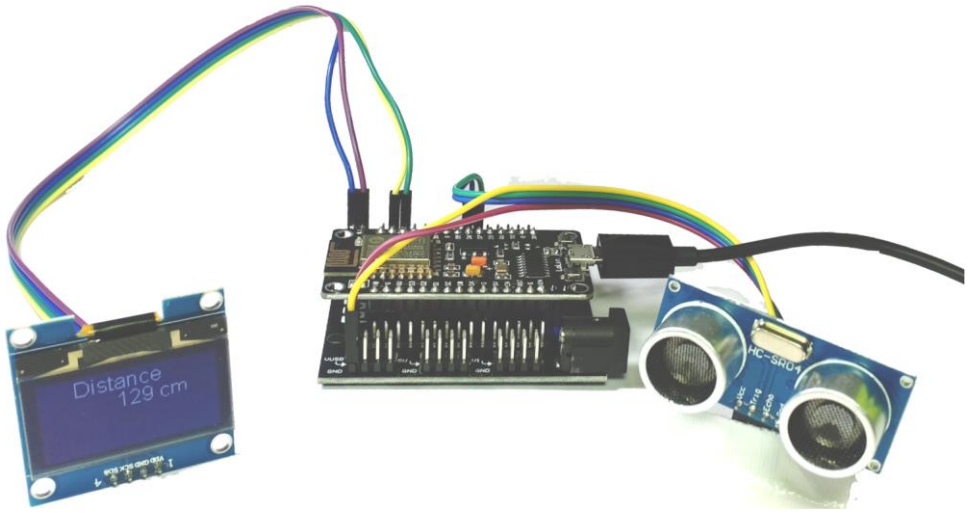
```
// Read Distance by HC-SR04 with NodeMCU ESP8266
// Display on OLED

#include <Wire.h>
#include <SH1106.h>

int TRIG_PIN = D6;
int ECHO_PIN = D7;

SH1106 display(0x3c, D1, D2);

void setup() {
    display.init();
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
}

unsigned long read_dist() {
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(12);
    digitalWrite(TRIG_PIN, LOW);
    unsigned long duration = pulseIn(ECHO_PIN,HIGH);
    unsigned long dist = (17.0*duration)/1000.;
    return(dist);
}

void loop() {
    unsigned long distance = read_dist();
    display.clear();
    display.setFont(ArialMT_Plain_16);
    display.drawString(25, 0, "Distance");
    display.drawString(54, 15, String(distance) +
" cm.");
    display.display();
    delay(200);
}

```

3. ทำการติดตั้งไลบรารี Ultrasonic.h แล้วเขียนโค้ดข้างล่างนี้แล้วอัปโหลดลง NodeMCU v.3 แล้วสังเกตสิ่งที่แสดงบนจอ OLED

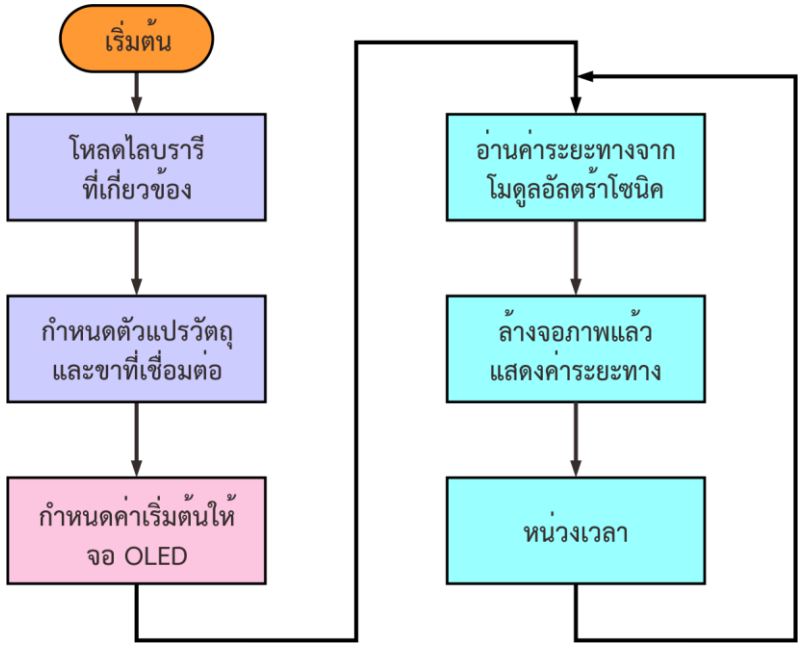
```
// Read Distance by HC-SR04 with NodeMCU ESP8266
// Use Ultrasonic.h library and Display on OLED

#include <Ultrasonic.h>
#include <Wire.h>
#include <SH1106.h>

// SH1106(Addr, SDA, SCL)
SH1106 display(0x3c, D1, D2);

// Ultrasonic(Trigger PIN, Echo PIN)
Ultrasonic ultrasonic(D6, D7);

void setup() {
    display.init();
}

void loop() {
    unsigned long distance = ultrasonic.Ranging(CM);
    display.clear();
    display.setFont(ArialMT_Plain_16);
    display.drawString(25, 0, "Distance");
    display.drawString(54, 15, String(distance) +
" cm.");
    display.display();
    delay(200);
}

```

รูปที่ 7.4 แสดงภาพผลลัพธ์ที่ได้จากโค้ดในการทดลองข้อที่ 2 และ 3 โดยเมื่อเราทำการวางวัตถุไว้ที่หน้าโมดูลอัลตราโซนิกส์ที่ระยะห่างต่าง ๆ จะพบว่า ค่าตัวเลข Distance เปลี่ยนแปลงไป

แผนผังการทำงาน (flowchart) ของโปรแกรมในการทดลองนี้ แสดงได้ดังรูปที่ 7.5



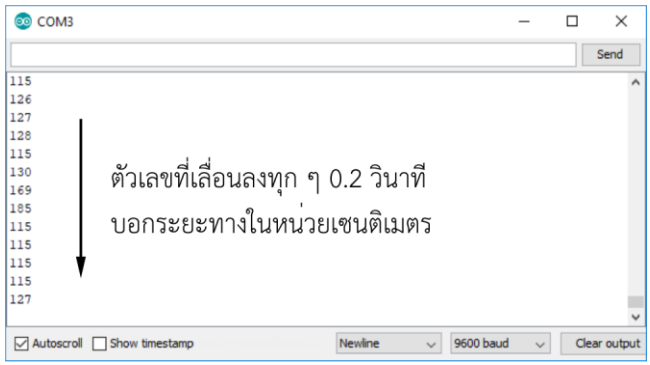
รูปที่ 7.4 ผลลัพธ์จากการทดสอบโปรแกรมตามการทดลองข้อที่ 2 และ 3



รูปที่ 7.5 แผนผังการทำงานของโปรแกรม

4. เขียนโค้ดข้างล่างนี้แล้วอัปโหลดลง NodeMCU v.3 แล้วสังเกตสิ่งที่แสดงผ่านพอร์ตอนุกรม โดยดูผ่าน Serial Monitor ใน Arduino IDE / Tools จะได้ผลดังแสดงตัวอย่างในรูปที่ 7.6

```
// Read Distance by HC-SR04 with NodeMCU ESP8266
// Display on Serial Monitor

#include <Ultrasonic.h>

// ultrasonic(Trigger PIN, Echo PIN)
Ultrasonic ultrasonic(D6, D7);

void setup() {
    Serial.begin(9600);
}

void loop() {
    unsigned long distance = ultrasonic.Ranging(CM);
    Serial.println(distance);
    delay(200);
}

```



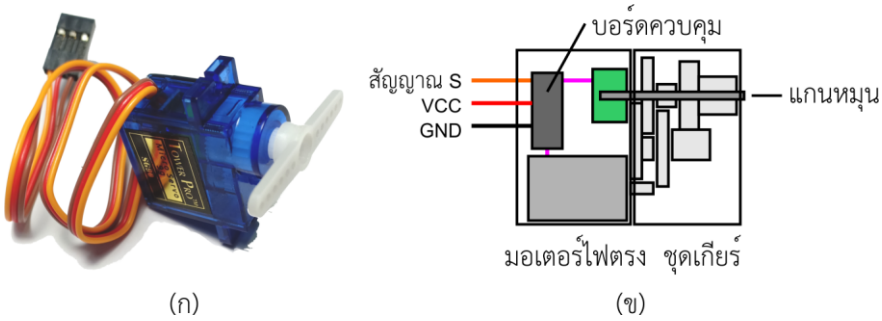
รูปที่ 7.6 ผลลัพธ์จากการทดสอบโปรแกรมตามการทดลองข้อที่ 4

บทที่ 8

การควบคุมเซอร์โวมอเตอร์

8.1 หลักการทำงานของเซอร์โวมอเตอร์

เซอร์โวมอเตอร์ (Servo Motor) คือ มอเตอร์ไฟฟ้ากระแสตรงที่มักถูกนำมาใช้ในการควบคุมมุมหรือตำแหน่งเชิงเส้นที่มีความละเอียดสูง โดยเซอร์โวมอเตอร์จะประกอบด้วยมอเตอร์ ชุดเกียร์ และบอร์ดควบคุม รวมไว้เป็นโมดูลเดียวกัน และจะรับสัญญาณควบคุม (signal, S) เพียง 1 เส้น ไฟเลี้ยง VCC และกราวด์ GND อีกอย่างละ 1 เส้น รวมเป็น 3 เส้น โดยทั่วไปเราสามารถควบคุมให้เซอร์โวมอเตอร์หมุนในทิศตามเข็มนาฬิกา (หมุนขวา) หรือทวนเข็มนาฬิกา (หมุนซ้าย) ได้ โดยมีมุมในการหมุนตั้งแต่ 0 องศา ถึง 180 องศา นั่นคือ เซอร์โวจะหมุนได้เพียง 180 องศาหรือครึ่งรอบเท่านั้น โดยมีตำแหน่งกึ่งกลางอยู่ที่ 90 องศา สัญญาณ S ที่ใช้ควบคุมมอเตอร์ชนิดนี้จะเป็นสัญญาณที่มีการมอดูเลตความกว้างพัลส์ (Pulse Width Modulation, PWM) และมีระดับแรงดันแบบ TTL ระดับแรงดัน VCC ที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 โวลต์ รูปที่ 8.1(ก) แสดงภาพถ่ายของเซอร์โวมอเตอร์ขนาดเล็กทั่วไปและ รูปที่ 8.1(ข) แสดงลักษณะภายในของเซอร์โวมอเตอร์นี้



รูปที่ 8.1 ลักษณะทั่วไปและโครงสร้างภายในของเซอร์โวมอเตอร์

การทำงานของเซอร์โวมอเตอร์

เมื่อจ่ายสัญญาณพัลส์เข้ามายังเซอร์โวมอเตอร์ในส่วนวงจรควบคุมภายในจะทำการอ่านและประมวลผลค่าความกว้างของสัญญาณพัลส์ที่ส่งเข้ามาเพื่อแปลค่าเป็นตำแหน่งองศาที่ต้องการให้มอเตอร์ หมุนเคลื่อนที่ไปยังตำแหน่งนั้น แล้วส่งคำสั่งไปทำการควบคุมให้มอเตอร์หมุนไปยังตำแหน่งที่ต้องการ โดยมีเซนเซอร์บอกตำแหน่งเป็นตัวป้อนกลับค่ามุมที่แกนหมุน ๆ ไป มาให้วงจรควบคุมเปรียบเทียบกับค่าอินพุตเพื่อควบคุมให้ได้ตำแหน่งที่ต้องการอย่างถูกต้องแม่นยำ

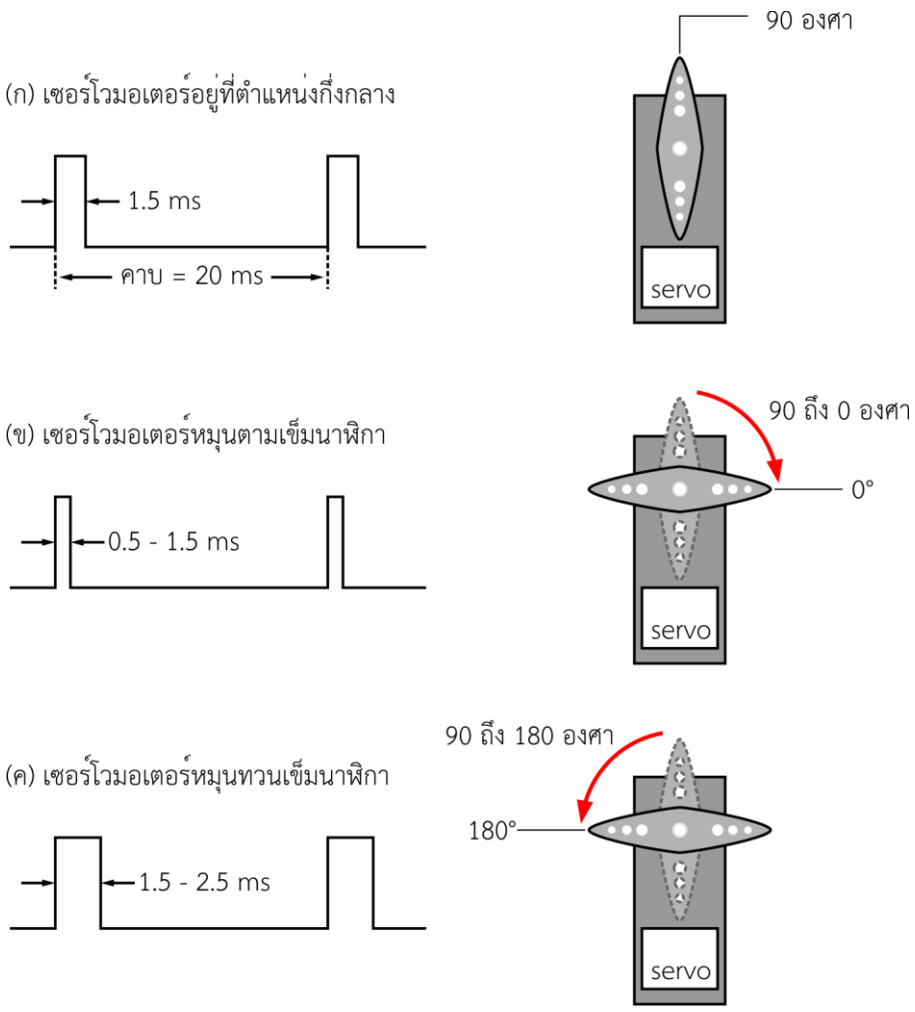
สัญญาณพัลส์สำหรับควบคุมเซอร์โวมอเตอร์

การควบคุมเซอร์โวมอเตอร์ทำได้โดยการสร้างสัญญาณพัลส์ป้อนให้กับวงจรควบคุมภายในเซอร์โวมอเตอร์ดังในรูปที่ 8.2 เริ่มต้นให้สร้างพัลส์คาบเวลา 20 มิลลิวินาที (ms) แล้วปรับความกว้างของพัลส์ให้เท่ากับ 1.5 ms จะได้ว่าแกนเซอร์โวจะอยู่ที่ตำแหน่งกึ่งกลาง (90 องศา) ดังรูปที่ 8.2(ก) การหมุนในช่วงมุม 0-90 องศา จะใช้พัลส์กว้างในช่วงประมาณ 0.5-1.5 ms มอเตอร์จะหมุนไปตำแหน่งขวามือ (ตามเข็มนาฬิกา, 0 ถึง 90 องศา) ดังรูปที่ 8.2(ข) และถ้าส่งพัลส์กว้างประมาณ 1.5-2.5 ms แกนหมุนของมอเตอร์จะหมุนไปด้านซ้ายมือ (ทวนเข็มนาฬิกา, 90 ถึง 180 องศา) ดังรูปที่ 8.2(ค)

การป้อนสัญญาณพัลส์ที่มีความกว้างตั้งแต่ ~0.5 ms ไปจนถึง 1.5 ms ทำให้เซอร์โวมอเตอร์หมุนทวนเข็มนาฬิกา ซึ่งถ้าค่าความกว้างพัลส์ยิ่งต่างจากค่า 1.5 ms มากเท่าใดมุมที่เซอร์โวมอเตอร์หมุนไปก็จะมากขึ้นเท่านั้น

การป้อนสัญญาณพัลส์ที่มีคาบเวลาช่วงบวกมากกว่า 1.5 ms ถึงประมาณ 2.5 ms จะทำให้เซอร์โวมอเตอร์หมุนทวนเข็มนาฬิกา โดยถ้าค่าความกว้างพัลส์ยิ่งห่างจาก 1.5 ms มากเท่าใดมุมที่เซอร์โวมอเตอร์หมุนไปก็จะมากขึ้นเท่านั้น

สำหรับการควบคุมความเร็วรอบในการหมุนเซอร์โวมอเตอร์นั้นเราก็สามารถทำได้โดยกำหนดขนาดของพัลส์และใช้การหน่วงเวลาประกอบ โดยการทดลองในตอนสุดท้าย (ตอนที่ 3) จะแสดงให้เห็นถึงวิธีการทางโปรแกรมในการปรับความเร็วของเซอร์โวมอเตอร์ให้หมุนช้าลง



รูปที่ 8.2 ลักษณะพัลส์ของสัญญาณที่ใช้ควบคุมและทิศทางการหมุนของเซอร์โวมอเตอร์ โดยพัลส์ที่แสดงจะควบคุมให้เซอร์โวมอเตอร์อยู่ในตำแหน่งที่ มุม (ก) 90 องศา (ข) 90-0 องศา และ (ค) 90 – 180 องศา

8.2 ไลบรารี Servo.h

ฟังก์ชันในไลบรารี Servo.h เป็นฟังก์ชันที่มาพร้อมกับ Arduino IDE โดยมีฟังก์ชันที่เกี่ยวข้องกับการควบคุมเซอร์โวมอเตอร์ คือ

ฟังก์ชัน attach() คือฟังก์ชันที่ใช้ในการกำหนดขาสัญญาณที่เซอร์โวมอเตอร์นั้นต่อกับบอร์ดไมโครคอนโทรลเลอร์และใช้กำหนดความกว้างของพัลส์ที่ 0 องศาและ 180 องศา การเรียกใช้ทำโดยใช้คำสั่ง

servo.attach(pin) หรือ servo.attach(pin, min, max)

โดย servo คือวัตถุที่เรากำหนดขึ้นก่อน และพารามิเตอร์ของคำสั่งนี้ ได้แก่ pin คือ ขาสัญญาณของบอร์ดไมโครคอนโทรลเลอร์ที่ใช้เชื่อมต่อกับเซอร์โวมอเตอร์ min คือ ความกว้างของพัลส์ที่ 0 องศาของเซอร์โวมอเตอร์ตัวที่ใช้ โดยกำหนดในหน่วยไมโครวินาที (µs) โดยปกติแล้วหากไม่มีการตั้งค่า โปรแกรมจะกำหนดค่าไว้ที่ 544, max คือ ความกว้างของพัลส์ที่ 180 องศาของ Servo ตัวที่ใช้ในหน่วยไมโครวินาที โดยปกติแล้วหากไม่มีการตั้งค่าโปรแกรมจะกำหนดค่าไว้ที่ 2400

ฟังก์ชัน write() คือฟังก์ชันที่ใช้ควบคุมตำแหน่งที่ต้องการให้เซอร์โวมอเตอร์หมุนไปยังองศาที่กำหนด สามารถกำหนดเป็นค่าองศาได้เลย คือ 0-180 องศา แต่ในเซอร์โวมอเตอร์ชนิดที่เป็น Full Rotation คำสั่ง write จะเป็นการกำหนดความเร็วในการหมุนโดยค่าเท่ากับ 90 คือคำสั่งให้มอเตอร์หยุดหมุน ค่าเท่ากับ 0 คือการหมุนด้วยความเร็วสูงสุดในทิศทางหนึ่ง และค่าเท่ากับ 180 คือการหมุนด้วยความเร็วสูงสุดในทิศทางตรงกันข้าม การเรียกใช้ทำโดยใช้คำสั่ง

servo.write(angle)

พารามิเตอร์ angle คือมุมที่ต้องการให้เซอร์โวมอเตอร์หมุนไป

ฟังก์ชัน writeMicroseconds() คือฟังก์ชันที่ใช้ควบคุมตำแหน่งที่ให้มอเตอร์หมุนไปยังตำแหน่งองศาที่กำหนดโดยกำหนดเป็นค่าความกว้างของพัลส์ในหน่วย µs ซึ่งปกติแล้ว

เซอร์โวมอเตอร์ทั่วไปจะใช้ความกว้างของพัลส์อยู่ที่ 1000-2000 µs ตามที่ได้กล่าวไปข้างต้นแล้ว มอเตอร์บางรุ่นหรือบางยี่ห้อไม่ได้ใช้ช่วงความกว้างของพัลส์ตามที่ได้กล่าวเอาไว้นี้ อาจจะใช้ช่วง 700-2300 แทนก็สามารถใช้ฟังก์ชัน writeMicroseconds นี้เพื่อกำหนดความกว้างพัลส์ได้เอง การใช้ฟังก์ชั่น writeMicroseconds สามารถกำหนดค่าได้อิสระ เราต้องระวังในการใช้งาน หากสั่งงานให้เซอร์โวมอเตอร์แบบ 0 - 180 องศา หมุนไปเกินจุดสิ้นสุดคือเกินทั้งฝั่ง 0 หรือ 180 องศา จะทำให้เกิดเสียงคราง ดังจากการหมุนไปต่อไม่ได้และมอเตอร์จะกินกระแสสูงขึ้นด้วยในเวลาเดียวกันนั้น ซึ่งอาจทำให้มอเตอร์เสียหายได้ การเรียกใช้ทำโดยใช้คำสั่ง

servo.writeMicroseconds(µs)

พารามิเตอร์ µs คือค่าความกว้างของพัลส์ที่ต้องการกำหนดในหน่วยไมโครวินาที (โดยตัวแปรจำนวนเต็ม)

ฟังก์ชัน read() คือฟังก์ชันอ่านค่าองศาที่สั่งเข้าไปด้วยฟังก์ชัน write() เพื่อให้ทราบว่าตำแหน่งองศาสุดท้ายที่เราสั่งเข้าไปนั้นมีค่าเท่าไหร่ซึ่งค่าที่อ่านออกมานั้นจะมีค่าอยู่ในช่วง 0 – 180 ฟังก์ชันนี้เรียกใช้โดยไม่มีพารามิเตอร์คือ

servo.read()

ฟังก์ชัน attached() คือฟังก์ชันที่ใช้ตรวจสอบว่าเซอร์โวมอเตอร์ที่เราต้องการใช้กำลังต่ออยู่กับขาสัญญาณของบอร์ดไมโครคอนโทรลเลอร์หรือไม่ การเรียกใช้ทำโดยใช้คำสั่ง servo.attached() โดยฟังก์ชันนี้จะส่งค่า True ออกมา หากเซอร์โวมอเตอร์เชื่อมต่ออยู่กับบอร์ด

ฟังก์ชัน detach() คือฟังก์ชันคืนสถานะของขาที่เรากำหนดให้เป็นขาควบคุมเซอร์โวมอเตอร์ด้วยคำสั่ง attach() ให้กลับคือสู่การใช้งานปกติ การเรียกใช้ทำโดยใช้คำสั่ง servo.detach()

8.3 การทดลองควบคุมเซอร์โวมอเตอร์

วัตถุประสงค์

1. เข้าใจการทำงานของเซอร์โวมอเตอร์
2. สามารถเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์ควบคุมเซอร์โวมอเตอร์ได้

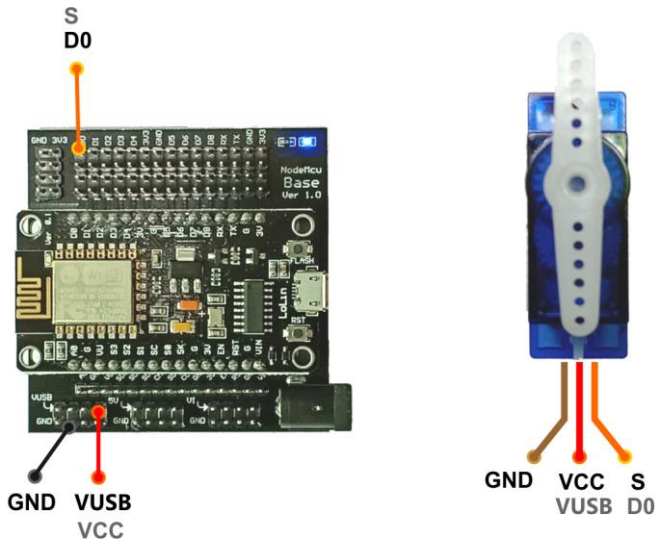
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	เซอร์โวมอเตอร์	1 ตัว
5.	อะแดปเตอร์ 9 V	1 ตัว
6.	สาย USB	1 เส้น
7.	สายต่อวงจร (สายจัมพ์ ผู้-เมีย)	3 เส้น

วิธีการทดลอง

ตอนที่ 1 การควบคุมทิศทางการหมุนของเซอร์โวมอเตอร์

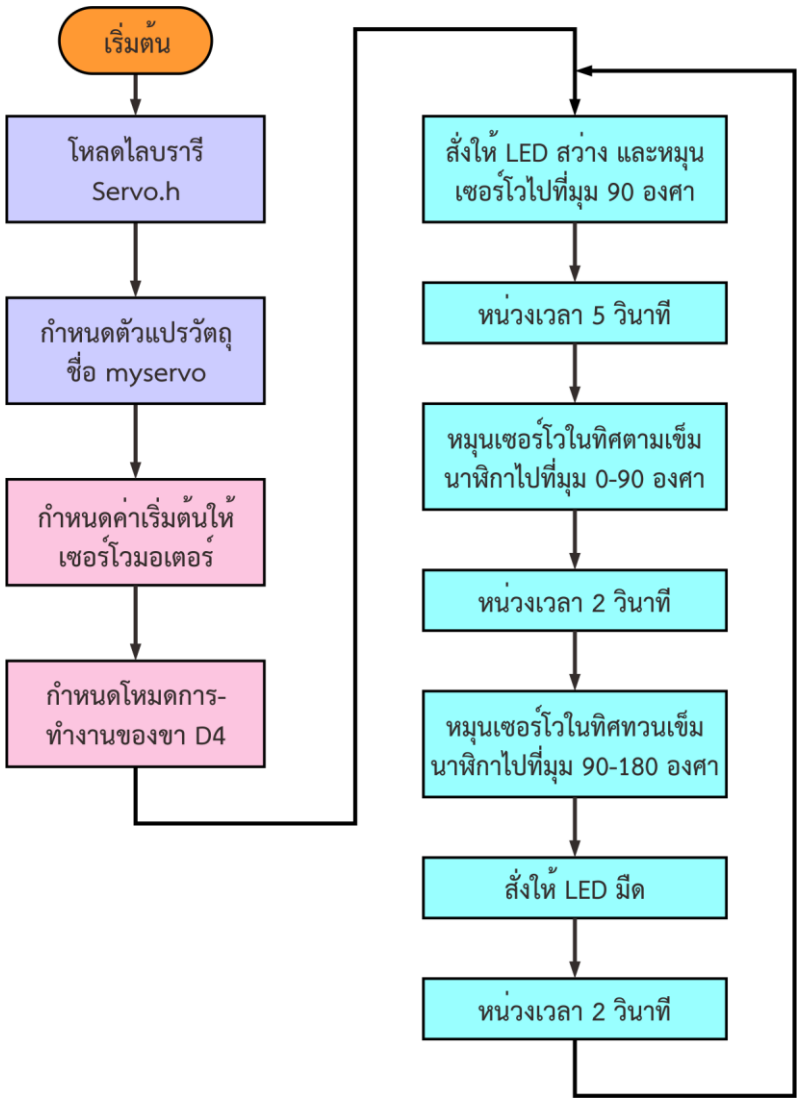
1. เชื่อมต่อเซอร์โวมอเตอร์เข้ากับบอร์ด NodeMCU v.3 ดังรูปที่ 8.3 โดยในการทดลองนี้ เราอาจจะต้องใช้อะแดปเตอร์ 9 V จ่ายไฟให้แก่บอร์ด NodeMCU Base เพื่อป้อนไฟให้แก่เซอร์โวมอเตอร์
2. เขียนโค้ดโปรแกรมดังที่แสดงในหน้าถัดไป จากนั้นจึงอัปโหลดแล้วสังเกตทิศทางการหมุนของเซอร์โวมอเตอร์
3. ทดลองเปลี่ยนค่าเวลาในบรรทัดที่ 17 จาก 1000 เป็น 600 แล้วสังเกตมุมสุดท้ายที่เซอร์โวมอเตอร์หมุนไปในทิศทางตามเข็มนาฬิกามากที่สุด



รูปที่ 8.3 การต่อวงจรในการทดลองควบคุมเซอร์โวมอเตอร์

```
// Control Servo Motor by NodeMCU ESP8266

#include <Servo.h>

Servo myservo;

void setup() {
    // Servo at PIN D0
    myservo.attach(D0);
    pinMode(D4, OUTPUT);   // On Board LED
}

void loop() {
    digitalWrite(D4,LOW);  // LED ON (active Low)
    myservo.writeMicroseconds(1500);  // Center
    delay(5000);
    myservo.writeMicroseconds(1000);  // Rotate CW
    delay(2000);
    myservo.writeMicroseconds(2000); // Rotate CCW
    digitalWrite(D4,HIGH);  // LED OFF (active high)
    delay(2000);
}

```

แผนผังการทำงานของโปรแกรมในตอนที่ 1 นี้แสดงดังรูปที่ 8.4



รูปที่ 8.4 แผนผังการทำงานของโปรแกรมที่แสดงในการทดลองตอนที่ 1

ตอนที่ 2 การควบคุมมุมของเซอร์โวมอเตอร์

1. เชื่อมต่อเซอร์โวมอเตอร์เข้ากับบอร์ด NodeMCU v.3 ดังรูปที่ 8.3

2. เขียนโค้ดโปรแกรมดังที่แสดงข้างล่างนี้ จากนั้นจึงอัปโหลดแล้วสังเกตตำแหน่งของเซอร์โวมอเตอร์

3. ทดลองเปลี่ยนค่ามุมในบรรทัดที่ 19 จาก 180 เป็น 135 แล้วสังเกตมุมสุดท้ายที่เซอร์โวมอเตอร์หมุนไปในทิศทางทวนเข็มนาฬิกามากที่สุด

```
1  // Control Servo Motor by NodeMCU ESP8266
2
3  #include <Servo.h>
4
5  Servo myservo;
6
7  void setup() {
8      // Servo at PIN D0
9      myservo.attach(D0);
10     pinMode(D4, OUTPUT);   // On Board LED
11 }
12
13 void loop() {
14     digitalWrite(D4,LOW);  // LED ON (active Low)
15     myservo.write(90);     // Center
16     delay(5000);
17     myservo.write(0);      // Rotate CW
18     delay(2000);
19     myservo.write(180);    // Rotate CCW
20     digitalWrite(D4,HIGH);  // LED OFF (active high)
21     delay(2000);
22 }
23
```

ตอนที่ 3 การควบคุมการเคลื่อนที่ของเซอร์โวมอเตอร์

1. เชื่อมต่อเซอร์โวมอเตอร์เข้ากับบอร์ด NodeMCU v.3 ดังรูปที่ 8.3

2. เขียนโค้ดโปรแกรมดังที่แสดงข้างล่างนี้ จากนั้นจึงอัปโหลดแล้วสังเกตการเคลื่อนที่ของเซอร์โวมอเตอร์

3. ทดลองเปลี่ยนค่าเวลาที่หน่วงในบรรทัดที่ 16 และ 21 จาก 15 เป็น 50 แล้วสังเกตความเร็วในการหมุนของเซอร์โวมอเตอร์หมุนไปในทิศทางทวนเข็มนาฬิกามากที่สุด

```
1  // Control Servo Motor by NodeMCU ESP8266
2
3  #include <Servo.h>
4
5  Servo myservo;
6
7  void setup() {
8      // Servo at PIN D0
9      myservo.attach(D0);
10 }
11
12 void loop() {
13     int pos;
14     for (pos = 0; pos < 180; pos += 1) {
15         myservo.write(pos);
16         delay(15);
17     }
18
19     for (pos = 180; pos >= 1; pos -= 1) {
20         myservo.write(pos);
21         delay(15);
22     }
23 }
24
```

บทที่ 9

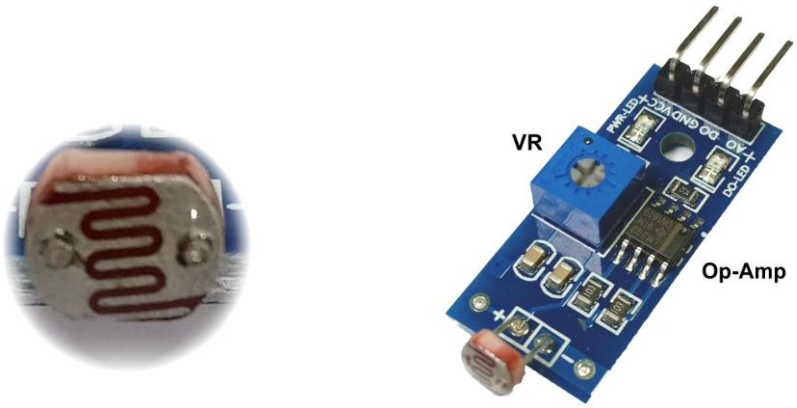
การวัดความสว่างด้วยเซนเซอร์ LDR

9.1 เซนเซอร์และโมดูลวัดความสว่าง

เราสามารถมองความเข้มแสงหรือความสว่างเป็นสัญญาณประเภทหนึ่งที่มนุษย์สามารถสัมผัสได้ด้วยดวงตา ความสว่างมีหน่วยเป็นลักซ์ (lux) เป็นหน่วยที่ใช้วัดค่าความสว่าง (Illuminance) ต่อพื้นที่ หรือคิดเป็นลูเมนต่อตารางเมตร โดยในปกติความสว่างตามสถานที่ต่าง ๆ นั้นได้มาจากแหล่งกำเนิดที่แตกต่างกัน เช่น จากหลอดไฟ ดวงอาทิตย์หรือแสงจากไฟบริเวณข้างเคียง เป็นต้น ในทางปฏิบัติจะมีการกำหนดค่าความสว่างที่เหมาะสม กับการใช้งานในสถานที่นั้น ๆ ซึ่งโดยทั่วไปจะมีค่าไม่น้อยกว่า 50-500 ลักซ์

ตัวต้านทานที่แปรค่าตามแสง (Light Dependent Resistor, LDR)

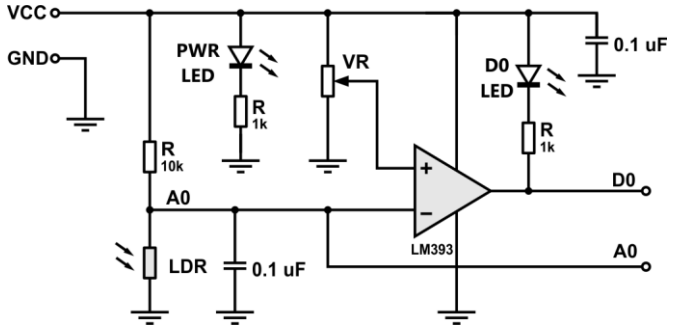
แอลดีอาร์ (LDR) หรือชื่อเต็ม ๆ คือ Light Dependent Resistor หรือตัวต้านทานที่แปรค่าตามแสง คือ ตัวต้านทานชนิดที่เปลี่ยนสภาพความนำไฟฟ้า (Conductance) ได้เมื่อมีแสงมาตกกระทบ ทำจากวัสดุสารกึ่งตัวนำที่ไวต่อแสง บางครั้งเราเรียก LDR เซนเซอร์ชนิดนี้ว่าโฟโตรีซีสเตอร์ (Photoresistor) หรือ โฟโตคอนดัคเตอร์ (Photoconductor) รูปที่ 9.1(ก) แสดงตัว LDR และ รูปที่ 9.1(ข) แสดงโมดูลวัดความสว่างที่ใช้ LDR เป็นเซนเซอร์ โมดูลนี้ให้สัญญาณเอาต์พุตได้ ทั้งแบบแอนะล็อกที่ช่อง (A0) ซึ่งมีค่าระหว่าง 0 - 1023 และแบบดิจิทัลที่ช่อง (D0) ค่า 0 กับ 1 โดยสามารถปรับระดับแรงดันที่นำเปรียบเทียบได้โดยการหมุนตัวต้านทานปรับค่าได้ (VR) บนบอร์ด และจะต้องป้อนใช้ไฟเลี้ยง 3.3-5V ให้กับวงจร ซึ่งบนบอร์ดจะมีแอลอีดีแสดงสัญญาณไฟเลี้ยง (PWR LED) และระดับสัญญาณที่เปรียบเทียบ (D0 LED) ด้วย



(ก) (ข)

รูปที่ 9.1 (ก) LDR และ (ข) โมดูลวัดความสว่างด้วยเซนเซอร์ LDR

โดยลักษณะวงจรภายในโมดูล LDR นี้ แสดงดังรูปที่ 9.2 โดยเมื่อมีการป้อนไฟเลี้ยง VCC และกราวด์ GND ให้กับบอร์ด ความต้านทานของ LDR จะถูกแปลงเป็นสัญญาณแรงดันผ่านวงจรแบ่งแรงดันเป็นสัญญาณแอนะล็อก A0 แล้วสัญญาณนี้จึงนำมาเปรียบเทียบกับสัญญาณอ้างอิงที่ได้จากวงจรแบ่งแรงดันอีกวงจรที่สร้างด้วยตัวต้านทานปรับค่าได้ VR โดยออปแอมป์ (LM393) จะเป็นไอซีที่ใช้เปรียบเทียบสัญญาณ จากนั้นจึงส่งสัญญาณดิจิทัลออกไปเป็นสัญญาณ D0 โดยเราสามารถสังเกตระดับสัญญาณ D0 ผ่าน D0 LED ที่ต่ออยู่กับด้านขาออกของออปแอมป์ได้ด้วย



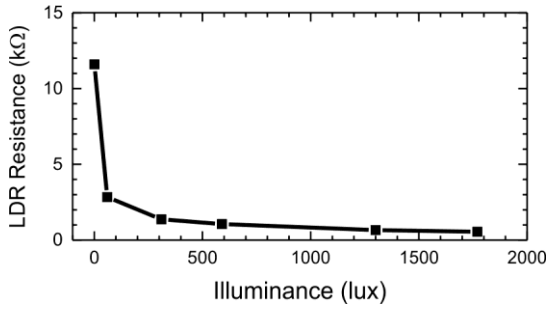
รูปที่ 9.2 ลักษณะวงจรภายในโมดูลวัดความสว่างด้วย LDR

9.2 การเทียบวัดค่าความสว่าง

ในการใช้โมดูลวัดความสว่างด้วยเซนเซอร์ LDR นี้ เพื่อให้การอ่านค่าความสว่าง (มีหน่วยเป็นลักซ์) ได้อย่างถูกต้อง เราควรจะทำการเทียบวัดค่า (calibration) เสียก่อน โดยหลักการเทียบวัดนั้น จะเริ่มจากการใช้เซนเซอร์อ่านค่าแสงที่ทราบค่าความสว่าง ซึ่งอาจวัดโดยใช้ลักซ์มิเตอร์วัดเปรียบเทียบ ตัวอย่างผลการทดลองกับ LDR ตัวหนึ่งแสดงในตารางที่ 9.1 และพล็อตเป็นกราฟได้ดังแสดงในรูปที่ 9.3

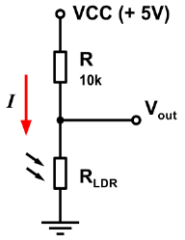
ตารางที่ 9.1 ความสัมพันธ์ระหว่างค่าความสว่างและความต้านทานของ LDR

ค่าความสว่าง (lux)	ค่าความต้านทาน (Ohm)
1	11600
60	2840
310	1374
590	1060
1300	664
1770	546



รูปที่ 9.3 ความสัมพันธ์ระหว่างค่าความสว่างและความต้านทานของ LDR

จากค่าความต้านทานที่เปลี่ยนแปลงตามความสว่าง เราสามารถแปลงข้อมูลสัญญาณเป็นแรงดันได้โดยใช้กฎการแบ่งแรงดัน ดังแสดงรูปวงจรในรูปที่ 9.4



รูปที่ 9.4 วงจรแบ่งแรงดัน

จากรูปวงจรที่ 9.4 เราสามารถคำนวณหาค่ากระแสไฟฟ้าที่ไหลในวงจรได้ดังสมการที่ (9.1) และแรงดันที่ตกคร่อม LDR หรือแรงดันที่ด้านออก (V_{out}) ดังสมการที่ (9.2)

$$I = \frac{V}{R + R_{LDR}} = \frac{5}{10\text{k}+R_{LDR}} \tag{9.1}$$

$$V_{out} = \frac{R_{LDR}}{10\text{k}+R_{LDR}} \times 5 \text{ V} \tag{9.2}$$

ตัวอย่าง ที่แสงความสว่าง 60 lux จงหาค่าแรงดัน V_{out} โดยแทนค่าความต้านทาน LDR ด้วยค่า 2.84 kΩ

$$V_{out} = \frac{R_{LDR}}{10\text{ k}+R_{LDR}} \times 5 \text{ V} = \frac{2.84\text{ k}}{10\text{k}+2.84\text{ k}} \times 5 \text{ V} = 1.106 \text{ V}$$

ตัวอย่าง ที่แสงความสว่าง 590 lux จงหาค่าแรงดัน V_{out} โดยแทนค่าความต้านทาน LDR ด้วยค่า 1.06 kΩ

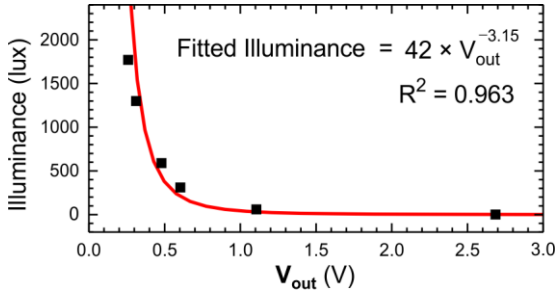
$$V_{out} = \frac{R_{LDR}}{10\text{ k}+R_{LDR}} \times 5 \text{ V} = \frac{1.06\text{ k}}{10\text{ k}+1.06\text{ k}} \times 5 \text{ V} = 0.479 \text{ V}$$

จากตารางที่ 9.1 และสมการที่ (9.2) เราสามารถสร้างตารางค่าแรงดันไฟฟ้าได้ดังตารางที่ 9.2

ตารางที่ 9.2 ความสัมพันธ์ระหว่างค่าความสว่างและแรงดันไฟฟ้าที่ตกคร่อมของ LDR

ค่าความสว่าง (lux)	ค่าความต้านทาน (Ohm)	แรงดันไฟฟ้า (Volt)
1	11600	2.69
60	2840	1.11
310	1374	0.60
590	1060	0.479
1300	664	0.311
1770	546	0.259

ในการหาค่าความสว่างจากข้อมูลที่แสดงในรูปที่ 9.3 เราจะฟิตข้อมูลและหาสมการความสัมพันธ์ของแสงและแรงดันไฟฟ้า ดังแสดงในรูปที่ 9.5



รูปที่ 9.5 การฟิตหาความสัมพันธ์ระหว่างค่าความสว่างและแรงดันไฟฟ้าที่วัดได้

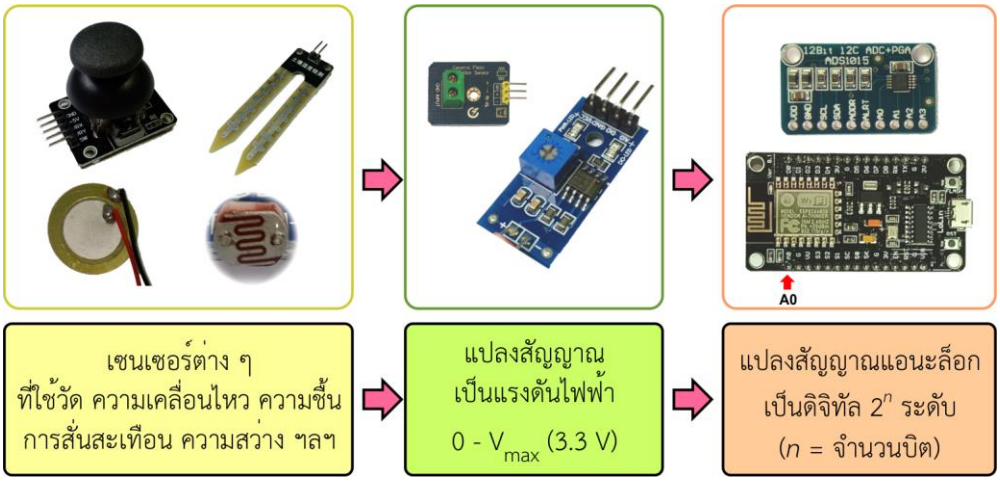
จากสมการที่ได้จากการฟิตกราฟในรูปที่ 9.5 ความสัมพันธ์ระหว่างแรงดันไฟฟ้ามีหน่วยเป็นโวลต์ (volt) ในแนวแกน x กับค่าความสว่างที่มีหน่วยเป็นลักซ์ (lux) ในแนวแกน y สมการที่ได้เป็นสมการส่วนกลับ ดังสมการที่ (9.3) และมีค่า R^2 เท่ากับ 0.963 ซึ่งเป็นค่าความเชื่อมั่นในระดับสูง

$$\text{Illuminance} = 42.0 \times V_{out}^{-3.15} \qquad (9.3)$$

9.3 การแปลงสัญญาณแอนะล็อกเป็นดิจิทัล

กระบวนการตรวจจับและแปลงสัญญาณหรือปริมาณทางกายภาพใด ๆ มาเป็นสัญญาณไฟฟ้าเป็นกระบวนการแรกในการประมวลสัญญาณ อุปกรณ์ที่สำคัญคือ ตัวตรวจจับหรือเซนเซอร์ หน้าที่สำคัญคือ การเปลี่ยนหรือแปลงปริมาณทางกายภาพมาเป็นสัญญาณทางไฟฟ้าเช่น จอยสติ๊ก (joystick) โพรบวัดความชื้น โพรบวัดความสั่นสะเทือนที่ทำจากวัสดุเพียโซอิเล็กทริก (piezoelectric material) ตัวต้านทานแปรค่าตามแสง (LDR) ใช้ในการตรวจจับความสว่าง เพื่อเปลี่ยนเป็นค่าความต้านทานทางไฟฟ้าแล้วจึงแปลงเป็นแรงดันไฟฟ้าตามลำดับ (ดูรูปที่ 9.6)

การปรับแต่งสัญญาณแรงดัน ซึ่งเป็นสัญญาณแอนะล็อกเป็นกระบวนการถัดมา โดยในส่วนนี้จะเป็นเรื่องของวงจรอิเล็กทรอนิกส์ที่จะทำให้สัญญาณไฟฟ้าที่ได้จากกระบวนการแรกเหมาะสมกับการเชื่อมต่อเข้ากับไมโครคอนโทรลเลอร์หรืออุปกรณ์แปลงสัญญาณแอนะล็อกเป็นดิจิทัล (Analog-to-Digital Conversion: ADC) โดยกระบวนการแปลงสัญญาณแอนะล็อกเป็นดิจิทัลเป็นขบวนการแปลงสัญญาณแอนะล็อกให้เป็นข้อมูลทางดิจิทัลเพื่อส่งไปประมวลผลยังไมโครคอนโทรลเลอร์ต่อไป



รูปที่ 9.6 ขบวนการแปลงสัญญาณแอนะล็อกเป็นดิจิทัล

ความละเอียดในการแปลงสัญญาณแอนะล็อก

ส่วนที่ต้องให้ความสนใจในส่วนนี้คือ ความละเอียดในการแปลงสัญญาณ วงจรแปลงสัญญาณแอนะล็อกเป็นสัญญาณดิจิทัลที่เหมาะสมในการนำมาใช้ในการทดลองทางวิทยาศาสตร์ควรมีความละเอียดไม่น้อยกว่า 8 บิต ซึ่งให้ความแตกต่างของข้อมูลไม่น้อยกว่า 256 ค่า (= 2^8) โดยเราสามารถหาจำนวนระดับความแตกต่างของข้อมูลได้โดยใช้สมการที่ (9.4) ส่วนค่าของข้อมูลดิจิทัล คือ 1024 สำหรับ NodeMCU หาได้จากสมการที่ (9.5) โดยกำหนดให้ความละเอียด *n* เท่ากับ 10 บิต ถ้าการแปลงสัญญาณนี้มีความละเอียดสูง (*n* มาก ๆ) เท่าใดก็จะยิ่งดี เพราะจะให้ผลการแปลงที่แม่นยำขึ้น สำหรับบอร์ด ADC ที่มีขายทั่วไปในท้องตลาดจะมีความละเอียด 8 – 24 บิต และสำหรับ NodeMCU วงจรแปลงสัญญาณภายในบอร์ดนี้มีความละเอียด 10 บิต

$$N = 2^n \text{ เมื่อ } n = 8 \text{ ได้ } N = 2^n = 256 \qquad (9.4)$$

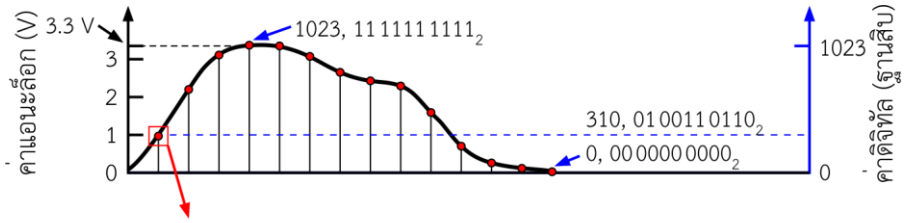
$$N = 2^n \text{ เมื่อ } n = 10 \text{ ได้ } N = 2^n = 1024 \qquad (9.5)$$

รูปที่ 9.7 แสดงตัวอย่างการแปลงสัญญาณแอนะล็อกเป็นดิจิทัลและการแปลงกลับ ความแยกชัด (Resolution) ของการแปลงสัญญาณกำหนดด้วย 1 LSB คือ Least Significant Bit หมายถึง ค่าแรงดันอินพุตในแต่ขั้นการเปลี่ยนข้อมูลการแปลง ช่วงห่างระหว่างขั้นของการเปลี่ยนแปลงแรงดันแอนะล็อกเอาต์พุตหาได้จากสมการที่ (9.6)

$$\text{Voltage Resolution} = \frac{\text{Full Scale Voltage}}{2^n} \qquad (9.6)$$

โดยเมื่อแรงดันอินพุทของการแปลงแอนะล็อกเป็นดิจิทัลเท่ากับ 3.3 V ที่ความละเอียด 10 บิต จะได้ Voltage resolution หรือ 1 LSB คือ 3.223 mV ดังแสดงในสมการที่ (9.7)

$$\text{Voltage Resolution} = 1 \text{ LSB} = \frac{3.3 \text{ V}}{1024} = 0.003223 \text{ V} \qquad (9.7)$$



แรงดัน **1.0** V คือ ค่าดิจิทัล = 1.0/Voltage Resolution $= \frac{1.0}{0.003223} =$ **310**

ค่าที่อ่านได้จาก ADC คือ **310** จะแปลงเป็นค่าแอนะล็อก $= \frac{310}{1024} \times 3.3\ V = 0.999 \approx$ **1.0** V

รูปที่ 9.7 ตัวอย่างการแปลงสัญญาณแอนะล็อกเป็นดิจิทัลและการแปลงกลับ

ตารางที่ 9.3 ตัวอย่างการแปลงแรงดันเป็นตัวเลขฐานสองขนาด 10 บิต

ค่าแรงดัน	ข้อมูลเลขฐานสิบ	ข้อมูลเลขฐานสอง
0.0 V	0	00 0000 0000
0.003223 V	1	00 0000 0001
0.3223 V	100	00 0110 0100
1.00 V	310	01 0011 0110
1.65 V	512	10 0000 0000
3.30 V	1023	11 1111 1111

คำสั่งในการอ่านค่าสัญญาณแอนะล็อกจาก NodeMCU

คำสั่ง analogRead(pin) โดย pin คือ ชื่อขา สำหรับ NodeMCU จะมีเพียงขาเดียว คือ ขา A0 (ดูรูปที่ 9.6 คอลัมน์ขวามือสุด)

โค้ดตัวอย่างการรับค่าสัญญาณแอนะล็อกแล้วคำนวณค่าแรงดันและค่าความสว่าง คือ

```
data_LDR = analogRead(A0);
volt_LDR = (3.3/1024) * data_LDR;
illu_LDR = 42.0 * pow(volt_LDR, -3.15);
```

9.4 การทดลองวัดค่าความสว่าง

วัตถุประสงค์

1. สามารถเข้าใจการแปลงข้อมูลจากสัญญาณแอนะล็อกเป็นดิจิทัลได้
2. สามารถต่อวงจรไมโครคอนโทรลเลอร์ NodeMCU v.3 กับโมดูลวัดแสงแบบแอนะล็อกได้
3. สามารถเขียนโปรแกรมแสดงการคำนวณค่าความสว่างและแสดงผลบน OLED Display ได้

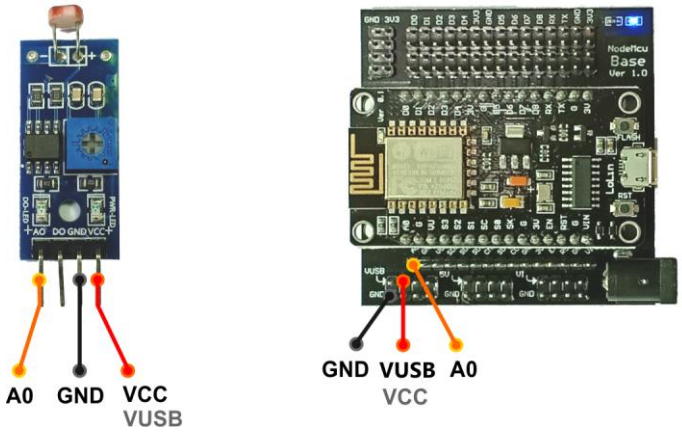
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	โมดูลวัดความสว่างด้วย LDR	1 โมดูล
5.	โมดูล OLED Display ขนาด 128×64 พิกเซล	1 โมดูล
6.	สาย USB	1 เส้น
7.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	7 เส้น

วิธีการทดลอง

ตอนที่ 1 การอ่านค่าแอนะล็อกและแสดงผลบนคอมพิวเตอร์

1. ต่อวงจรโดยป้อนแรงดัน 5 V และ กราวด์ให้แก่โมดูลวัดแสงแบบแอนะล็อกด้วย LDR และป้อนสัญญาณ A0 เข้าสู่ขา A0 ดังรูปที่ 9.8

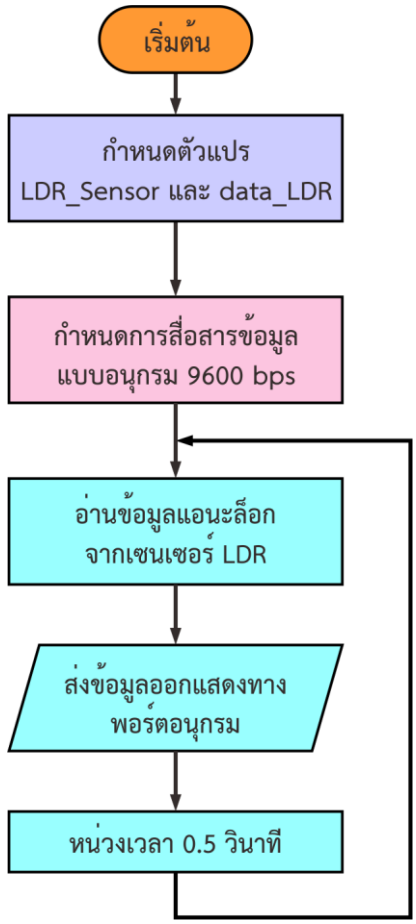


รูปที่ 9.8 การเชื่อมต่อ NodeMCU v.3 กับ โมดูลวัดแสงแบบแอนะล็อก

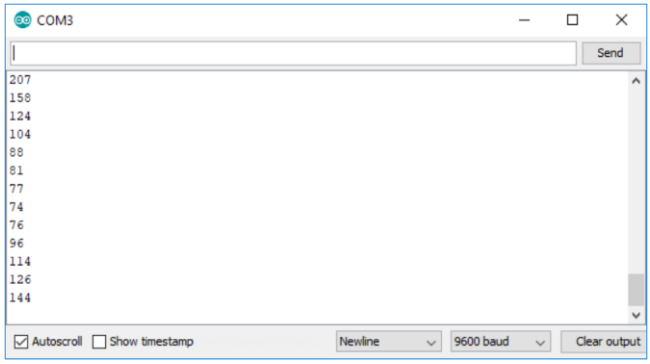
2. เขียนโค้ดโปรแกรมดังแสดงข้างล่างนี้ แล้วอัปโหลดและสังเกตผลที่แสดงบนคอมพิวเตอร์ผ่านพอร์ตอนุกรม ทั้งทาง Serial Monitor และ Serial Plotter ในเมนู Tools

```
1  // Read LDR by NodeMCU ESP8266
2  // Result is sent to serial port.
3
4  // Pin A0 = analog input
5  int LDR_Sensor = A0;
6  int data_LDR;
7
8  void setup() {
9      // Set serial speed to 9600 bps
10     Serial.begin(9600);
11 }
12
13 void loop() {
14     // Read Analog Data
15     data_LDR = analogRead(LDR_Sensor);
16     Serial.println(data_LDR);
17     delay(500);
18 }
19
```

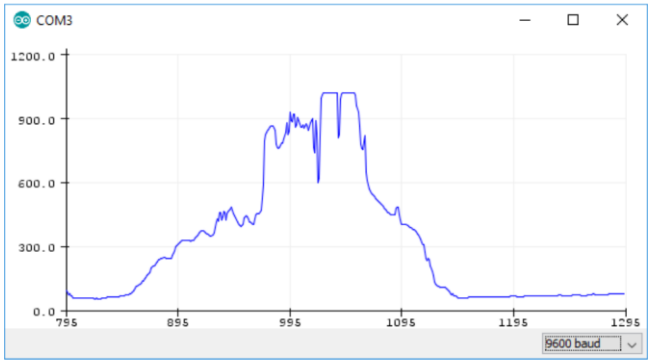
แผนผังการทำงานของโปรแกรมที่เขียนในตอนที่ 1 แสดงดังรูปที่ 9.9 และผลลัพธ์อาจแสดงเป็นตัวเลข (ผ่าน Serial Monitor) และเป็นกราฟ (ผ่าน Serial Plotter) ได้ดังรูปที่ 9.10 (ก) และ 9.10 (ข) ตามลำดับ



รูปที่ 9.9 แผนผังการทำงานของโปรแกรมในการทดลองตอนที่ 1



(ก)



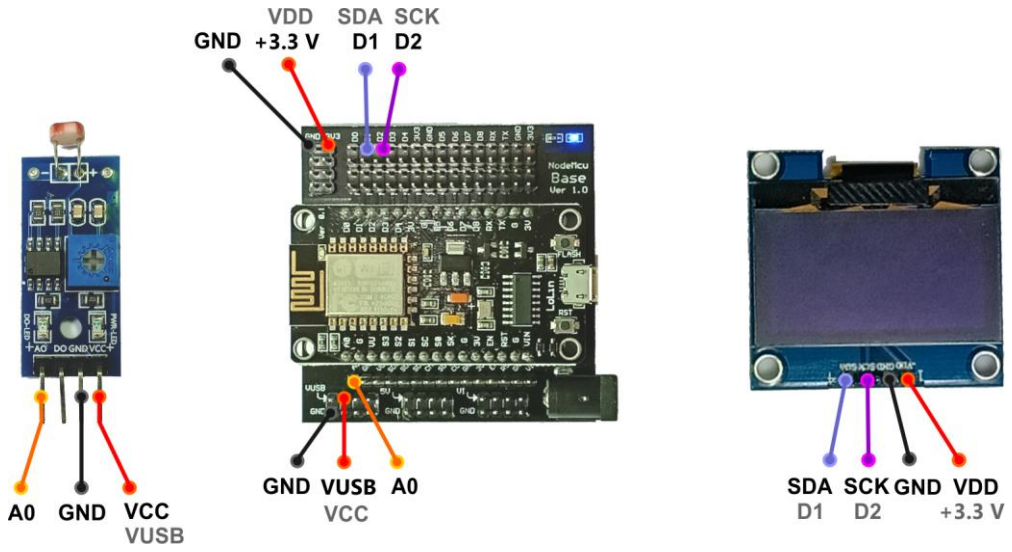
(ข)

รูปที่ 9.10 ผลลัพธ์ของโปรแกรมในการทดลองตอนที่ 1 ที่เปิดดูผ่าน (ก) Serial Monitor และ (ข) Serial Plotter

ตอนที่ 2 การอ่านค่าแอนะล็อกและแสดงผลบนจอ OLED

1. ต่อวงจรดังรูปที่ 9.11

2. เขียนโค้ดโปรแกรมดังที่แสดงข้างล่างนี้ แล้วอัปโหลดและสังเกตผลที่แสดงบนหน้าจอ OLED Display และผลบนคอมพิวเตอร์ (ผ่าน Serial Monitor)

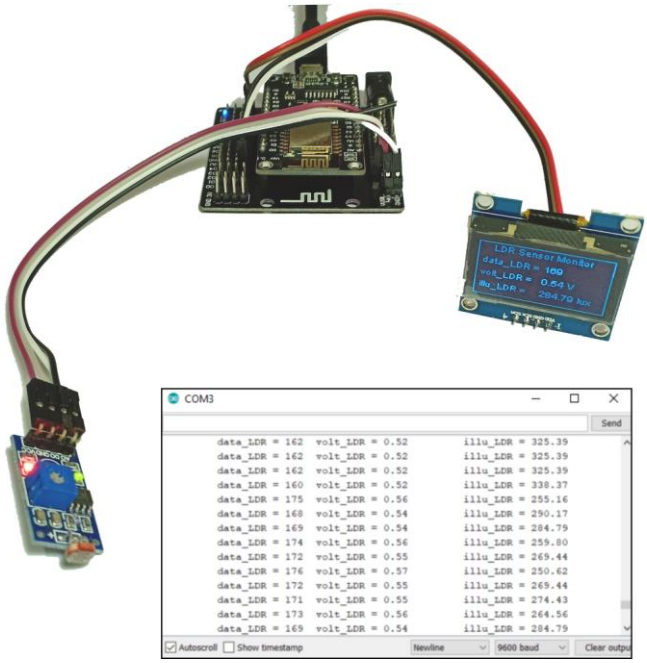


รูปที่ 9.11 การเชื่อมต่อ NodeMCU v.3 กับ โมดูลวัดแสงแบบแอนะล็อก และจอแสดงผล OLED Display

```
// Read LDR by NodeMCU ESP8266
// result is sent to OLED & Serial

#include <Wire.h>
#include <SH1106.h>

// SH1106(Addr, SDA, SCL)
SH1106 display(0x3c, D1, D2);  // OLED

// Pin A0 = analog input
int LDR_Sensor = A0;
int data_LDR;
float volt_LDR;
float illu_LDR;

void setup() {
    display.init();
    // Set serial speed to 9600 bps
    Serial.begin(9600);
}

```

```
22 void Calculate() {
23     data_LDR = analogRead(LDR_Sensor);
24     volt_LDR = (3.3/1024)*data_LDR;
25     illu_LDR = 42.0 * pow(volt_LDR, -3.15);
26 }
27
28 void show_OLED() {
29     display.clear();
30     display.drawRect(0, 0, 128, 64);
31     display.setFont(ArialMT_Plain_10);
32     display.drawString(15,0, "LDR Sensor
   Monitor");
33
34     display.drawString(5, 15, "data_LDR = ");
35     display.drawString(65, 15, String(data_LDR));
36
37     display.drawString(5, 30, "volt_LDR = ");
38     display.drawString(65, 30, String(volt_LDR) +
   " V");
39
40     display.drawString(5, 45, "illu_LDR = ");
41     display.drawString(65, 45, String(illu_LDR) +
   " lux");
42
43     display.display();
44 }
45
46 void show_Serial() {
47     Serial.print("\t data_LDR = ");
48     Serial.print(data_LDR);
49     Serial.print("\t volt_LDR = ");
50     Serial.print(volt_LDR);
51     Serial.print("\t illu_LDR = ");
52     Serial.println(illu_LDR);
53 }
54
55 void loop() {
56     Calculate();
57     show_OLED();
58     show_Serial();
59     delay(500);
60 }
61
```

รูปที่ 9.12 แสดงภาพถ่ายระบบที่ทดสอบและผลลัพธ์ที่ได้ บนหน้าจอ OLED และบนจอคอมพิวเตอร์ ผ่าน Serial Monitor



รูปที่ 9.12 การเชื่อมต่อ NodeMCU v.3 กับ โมดูลวัดแสงแบบแอนะล็อก และ OLED ที่แสดงพร้อมผลลัพธ์จากการทดลองตอนที่ 2

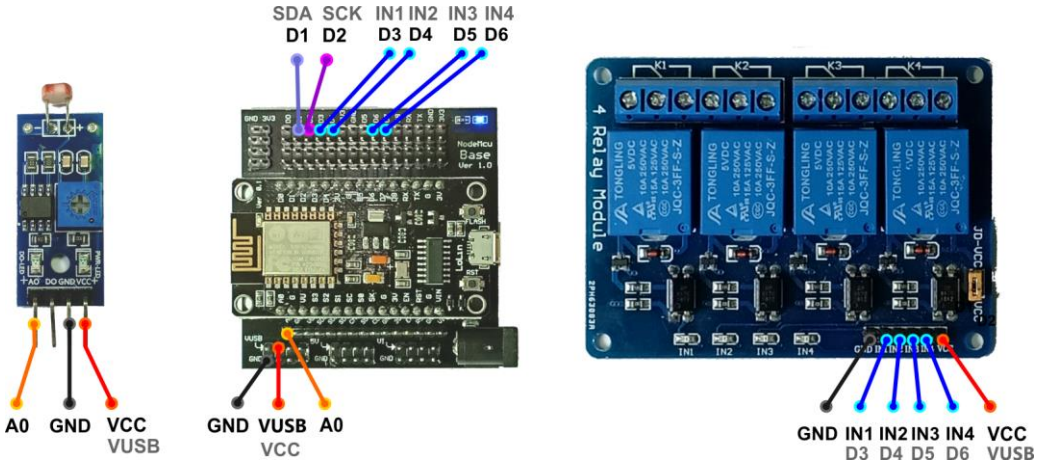
แบบฝึกหัดท้ายการทดลอง

จงเขียนโปรแกรมควบคุมการทำงานของรีเลย์ 1 – รีเลย์ 4 โดยมีการรับค่าจากเซนเซอร์วัดความสว่าง และมีเงื่อนไขการทำงานดังแสดงในตารางที่ 9.4

ข้อแนะนำ ควรต่อวงจรตามรูปที่ 9.13

ตารางที่ 9.4 เงื่อนไขของโปรแกรมควบคุมรีเลย์

เงื่อนไข	รีเลย์ 1	รีเลย์ 2	รีเลย์ 3	รีเลย์ 4
ความสว่าง $\leq$ 20 lux	ON	ON	ON	ON
20 < ความสว่าง $\leq$ 50	ON	ON	ON	OFF
50 < ความสว่าง $\leq$ 100	ON	ON	OFF	OFF
100 < ความสว่าง $\leq$ 200	ON	OFF	OFF	OFF
ความสว่าง > 200 lux	OFF	OFF	OFF	OFF



รูปที่ 9.13 การเชื่อมต่อ NodeMCU v.3 กับ โมดูลวัดแสงแบบแอนะล็อก และโมดูลรีเลย์

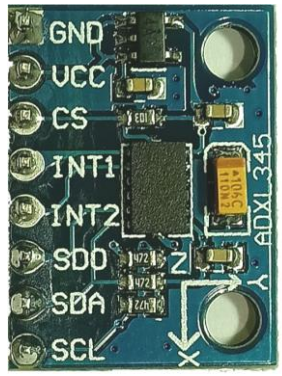
บทที่ 10

การใช้งานเซนเซอร์วัดความเร่งแบบสามแกน

10.1 โมดูลวัดความเร่งแบบสามแกน ADXL345

บอร์ดโมดูลที่กล่าวถึงในบทนี้คือ โมดูลวัดความเร่งแบบสามแกนแบบดิจิทัลที่ใช้ชิปเบอร์ ADXL345 เป็นเซนเซอร์วัดสัญญาณความเร่งในสามแกน ที่มีนิยามทิศทางแสดงอยู่บนบอร์ดอย่างชัดแจ้ง (รูปที่ 10.1) โดยโมดูลนี้ให้ให้สัญญาณขาออกเป็นข้อมูลดิจิทัลและติดต่อใช้งานผ่าน I2C ได้และสามารถกำหนดค่าความโน้มถ่วงสูงสุดที่อ่าน ได้สูงสุดถึง ±16g (±2g, ±4g, ±8g, และ ±16g) โดย g คือค่าความเร่งจากแรงดึงดูดของโลก (ประมาณ 10 m/s^2)

ชิป ADXL345 เป็นระบบวัดการเร่งความเร็ว แบบ 3 แกนที่ทำงานได้ในระดับการใช้พลังงานที่ต่ำมาก วัดได้ทั้งความเร่ง/ความเร็วแบบไดนามิกอันเป็นผลมาจากการเคลื่อนไหวหรือแรงกระแทกและการเร่งความเร็วแบบคงที่เช่นการเอียง ง่ายต่อการสื่อสารกับไมโครคอนโทรลเลอร์รุ่นใหม่ ๆ ผ่านการสื่อสารแบบ I2C ที่ใช้เพียงขา SCL และ SDA



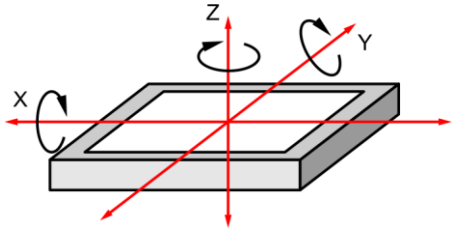
รูปที่ 10.1 โมดูลวัดความเร่งแบบ 3 แกน ADXL345

คุณสมบัติของชิปวัดค่าความเร่ง ADXL345

- ใช้ไฟเลี้ยงในช่วงกว้าง 2.0 - 3.6 V
- กินพลังงานต่ำมาก (23 μA ที่ไฟเลี้ยง 2.5 V)
- สามารถเลือกความละเอียดในการวัดผ่านคำสั่งที่รับผ่านการสื่อสารแบบอนุกรมได้
- ข้อมูลที่อ่านได้มีความละเอียดสูงถึง 0.004 g/LSB (13 บิต)
- สามารถนำมาเชื่อมต่อได้ทั้งแบบแบบ SPI และ I2C

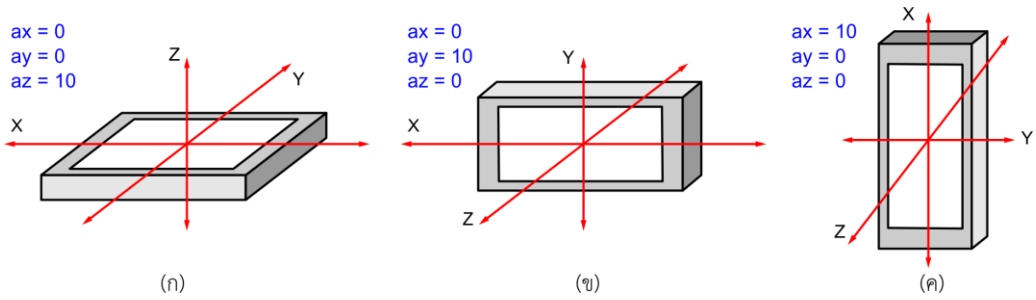
การวัดความเร่งแบบ 3 แกน

การวัดมุมเอียงของเซนเซอร์สามารถทำได้โดยการวัดความเร่งในแต่ละแกน โดยทั่วไปเราจะนิยามการเอียงเมื่อเทียบกับแกนหรือระบบพิกัดอ้างอิง โดยบอกความเอียงเป็นมุมที่หมุนเอียงไปในแกนหนึ่ง ๆ ซึ่งการเอียงในแต่ละทิศจะมีลักษณะการกำหนดทิศทางดังรูปที่ 10.2



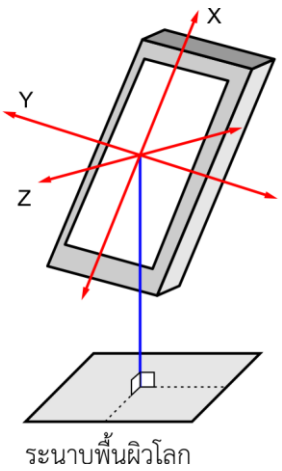
รูปที่ 10.2 ทิศทางการเอียงในมุมต่าง ๆ เสมือนการหมุนรอบแกน X, Y และ Z

โมดูลวัดความเร่งนี้จะวัดความเร่งในแต่ละแกน ก็คือ ขณะที่โมดูลอยู่นิ่ง ๆ ค่าความเร่งในแต่ละแกนน่าจะมีค่าเป็นศูนย์ แต่ในความเป็นจริงอย่าลืมว่ายังมีแรงโน้มถ่วงของโลกอยู่ด้วย ดังนั้นค่าที่อ่านได้จึงไม่ได้เป็นศูนย์ทั้งหมด คือ ถ้าเราตั้งเครื่องให้แกน Z ตั้งฉากกับพื้นโลก แกน X และ Y จะเป็น 0 แต่ว่าแกน Z จะไม่เป็น 0 เพราะมีแรงโน้มถ่วงของโลกกระทำอยู่ ดังนั้นค่าที่ได้จากแกน Z จึงมีค่า 9.81 m/s^2 หรือประมาณ 10 m/s^2 รูปที่ 10.3 แสดงลักษณะการวางโมดูลที่ทำให้อ่านค่าได้ค่าต่าง ๆ



รูปที่ 10.3 ลักษณะการวางโมดูลให้แกน (ก) Z (ข) Y และ (ค) X ตั้งฉากกับผิวโลก ตามลำดับ

ในกรณีที่อยู่นิ่ง ๆ และแกน X, Y และ Z ไม่มีแกนใดตั้งฉากกับพื้นโลก แรงโน้มถ่วงของโลกที่กระทำกับแต่ละแกนของโมดูลวัดความเร่งก็จะกระจายออกไปในแต่ละแกน ขึ้นอยู่กับการเอียงของเซนเซอร์ แต่เมื่อคิดเวกเตอร์ลัพธ์ที่ตั้งฉากกับพื้นโลกก็มีค่าประมาณ 10 m/s^2 ดังแสดงในรูปที่ 10.4



$$a = \sqrt{ax^2 + ay^2 + az^2}$$
$$= \sqrt{9.0^2 + 4.0^2 + 2.0^2}$$
$$\approx 10$$

รูปที่ 10.4 ตัวอย่างค่าที่อ่านได้เมื่อวางโมดูลในลักษณะไม่ตั้งฉากกับพื้นโลก

10.2 ไลบรารีและคำสั่งที่เกี่ยวข้อง

ในขั้นแรกของการใช้งานโมดูล ADXL345 เราจะต้องดาวน์โหลดไลบรารีมาลงในโปรแกรม Arduino IDE เสียก่อน โดยผู้ใช้จะต้องดาวน์โหลดไลบรารีสองตัวคือ Adafruit_Sensor จาก https://github.com/adafruit/Adafruit_Sensor และ Adafruit_ADXL345 จาก https://github.com/adafruit/Adafruit_ADXL345

โดยหลังจากติดตั้งไลบรารีลงใน Arduino IDE แล้ว ผู้ใช้สามารถเรียกใช้งานโมดูล ADXL345 ได้โดยเริ่มจากการเพิ่มไลบรารี 3 ตัวได้แก่ Wire.h, Adafruit_Sensor.h และ Adafruit_ADXL345_U.h แล้วจึงกำหนดวัตถุคือ โมดูล ADXL345 ที่เชื่อมต่อ โดยใช้คำสั่ง

Adafruit_ADXL345_Unified accel = **Adafruit_ADXL345_Unified(12345)**;

โดย accel คือชื่อวัตถุที่สร้างจากคลาส Adafruit_ADXL345_Unified ซึ่งคลาสนี้เป็นคลาสที่ใช้งานได้อย่างหลากหลาย มีฟังก์ชันที่เป็นเมธอดที่อ่านค่าความเร่งอยู่ภายในพร้อมให้เรียกใช้งาน โดยการเชื่อมต่อโมดูลนี้ ชิป ADXL345 กำหนดให้มีการเชื่อมต่อแบบ I2C โดยมี address คือ 0x53 และเชื่อมต่อผ่านขา D1 และ D2 ของ NodeMCU

ฟังก์ชัน begin() เป็นฟังก์ชันที่เรียกเพื่อใช้ตรวจสอบการเชื่อมต่อของโมดูลและไมโครคอนโทรลเลอร์ โดยหลังจากที่เชื่อมต่อได้แล้ว เราจะสามารถกำหนดขอบเขตของการอ่านค่าได้โดยใช้คำสั่ง accel.setRange(ADXL345_RANGE_xx_G) โดย xx = 2, 4, 8, 16 และกำหนดอัตราเร็วในการส่งข้อมูล ด้วยคำสั่ง setDataRate(dataRate) โดยฟังก์ชันที่กล่าวถึงนี้ จะอยู่ในส่วนของ setup() ของโปรแกรม

การเรียกใช้งาน (อ่านค่า) ความเร่งจากโมดูลทำได้โดยกำหนดตัวแปร เหตุการณ์ คือ sensors_event_t event; จากนั้นจึงใช้ฟังก์ชัน (เมธอด) getEvent เพื่ออ่านค่าความเร่งแล้วเราสามารถเรียกดูค่าความเร่งในแต่ละแกนได้จาก event.acceleration.x (.y และ .z)

ในการใช้งานโมดูลนี้ หากเราต้องการค่าที่ถูกต้องแม่นยำ เราอาจจะทำการเทียบวัด (calibration) ก่อน ด้วยวิธีการทางซอฟต์แวร์ โดยผู้ที่สนใจวิธีการนี้ ก็สามารถศึกษาเพิ่มเติม

ได้จากแหล่งความรู้ในอินเทอร์เน็ตเช่น https://learn.adafruit.com/adxl345-digital-accelerometer/programming

สำหรับขาที่ใช้ในการเชื่อมต่อเพื่อใช้งานโมดูล ADXL345 มีด้วยกัน 4 ขา คือ VCC, GND, SDA และ SCL โดยในการทดลองนี้ เราจะเชื่อมต่อขา D1 และ D2 ของ NodeMCU เข้ากับขา SCL และ SDA ตามลำดับ

10.3 การทดลองวัดค่าความเร่ง

วัตถุประสงค์

1. สามารถต่อวงจร NodeMCU v.3 กับโมดูลวัดความเร่ง 3 แกน ได้
2. สามารถเขียนโปรแกรมแสดงผลการวัดความเร่งได้
3. สามารถเขียนโปรแกรมเพื่อนำค่าความเร่งที่วัดได้ไปใช้งานต่อไปได้

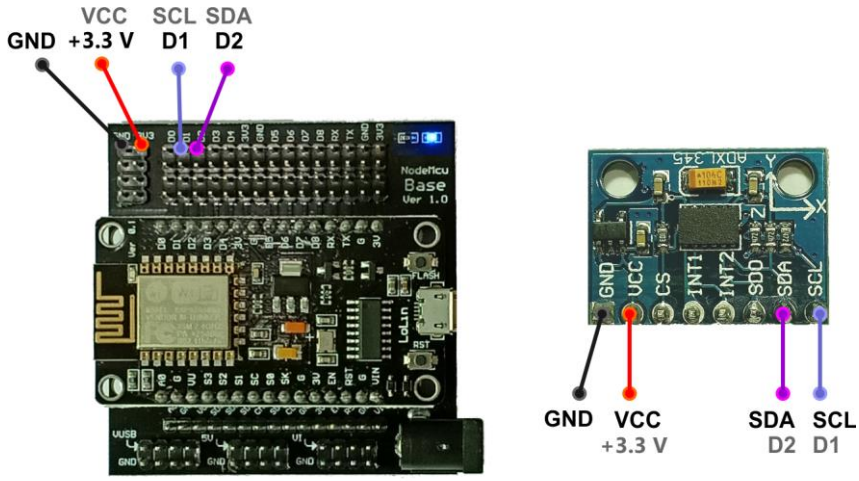
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดโมดูลวัดความเร่ง 3 แกน ด้วยไอซีเบอร์ ADXL345	1 บอร์ด
5.	สาย USB	1 เส้น
6.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	4 เส้น

วิธีการทดลอง

ตอนที่ 1

1. ต่อวงจรตามรูปที่ 10.5
2. เขียนโค้ดโปรแกรมเพื่ออ่านค่าความเร่งทั้งสามแกนและแสดงผลผ่านพอร์ตอนุกรม



รูปที่ 10.5 การเชื่อมต่อแบบ I2C ระหว่าง NodeMCU กับโมดูลวัดความเร่ง

```
// Read 3-axis accelerometer by NodeMCU ESP8266

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
Adafruit_ADXL345_Unified accel =
Adafruit_ADXL345_Unified(12345);

void setup() {
    Serial.begin(9600);
    Serial.println("Accelerometer Test");
    Serial.println("");
    if(!accel.begin()) {
       Serial.println("Oops, no ADXL345 detected.");
       while(1);
    }
    accel.setRange(ADXL345_RANGE_16_G);
}

void loop() {
    sensors_event_t event;
    accel.getEvent(&event);

```

```
    Serial.print("X: ");
    Serial.print(event.acceleration.x);
    Serial.print(" ");

    Serial.print("Y: ");
    Serial.print(event.acceleration.y);
    Serial.print(" ");

    Serial.print("Z: ");
    Serial.print(event.acceleration.z);
    Serial.print(" ");
    Serial.println("m/s^2");

    delay(500);
}
```

3. ทดสอบผลการวัดความเร่งในแนวแกน X, Y, และ Z โดยการหมุนเซนเซอร์ในทิศต่าง ๆ แล้วอ่านค่าที่แสดง

ตอนที่ 2

1. ต่อวงจรตามรูปที่ 10.5

2. เขียนโค้ดโปรแกรมเพื่อตรวจสอบเหตุการณ์การเคลื่อนที่อย่างรวดเร็ว ซึ่งเป็นเหตุการณ์ที่ทำให้มีขนาดความเร่งรวมเกิน 20 m/s^2 และแสดงผลเป็นข้อความผ่านพอร์ตอนุกรม

```
//Detect event by accelerometer + NodeMCU ESP8266

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
Adafruit_ADXL345_Unified accel =
Adafruit_ADXL345_Unified(12345);

// Threshold value for detecting accident event
float acth = 20.0;
```

```
10 float acx,acy,acz,ac;
11
12 void setup() {
13     Serial.begin(9600);
14     Serial.println("Accelerometer Test");
15     Serial.println("");
16     if(!accel.begin()) {
17       Serial.println("Oops, no ADXL345 detected.");
18       while(1);
19     }
20     accel.setRange(ADXL345_RANGE_16_G);
21 }
22
23 void loop() {
24     sensors_event_t event;
25     accel.getEvent(&event);
26
27     acx = event.acceleration.x;
28     acy = event.acceleration.y;
29     acz = event.acceleration.z;
30
31     ac = sqrt(pow(acx,2)+pow(acy,2)+pow(acz,2));
32
33     if(ac > acth) {
34       Serial.println("accident detected!");
35     }
36     delay(100);
37 }
38
```

3. ทดสอบผลโดยการขยับเซนเซอร์เร็ว ๆ แล้วสังเกตที่ Serial Monitor

บทที่ 11

การใช้งานโมดูลสื่อสารไร้สายด้วยเทคโนโลยีบลูทูธ

11.1 การสื่อสารไร้สายด้วยเทคโนโลยีบลูทูธ

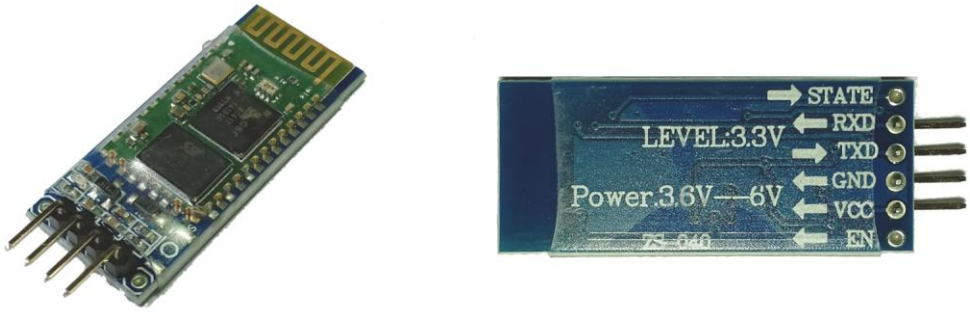
บลูทูธ (Bluetooth) คือ มาตรฐานของเทคโนโลยีการสื่อสารไร้สายที่ใช้ในการแลกเปลี่ยนข้อมูลในระยะทางสั้น ๆ ที่พัฒนาโดยกลุ่ม Bluetooth special Interest Group (BIG) โดยการสื่อสารนี้จะใช้คลื่นวิทยุในช่วง UHF (2.4-2.485 GHz) โดยทั่วไปบลูทูธจะถูกนำไปใช้กับอุปกรณ์เคลื่อนที่ โดยอุปกรณ์จะสร้างเครือข่ายส่วนตัว (personal area networks, PANs) ที่มักจะมีระยะทางในการส่งข้อมูลไม่เกิน 10 เมตร และจำนวนอุปกรณ์ไม่เกิน 7 ชิ้น จุดประสงค์ของการพัฒนาบลูทูธ เพื่อนำมาใช้แทนที่การส่งข้อมูลแบบ RS-232 ที่ใช้สาย โดยปัจจุบันเทคโนโลยีนี้ถูกพัฒนาไปถึง Bluetooth 5 ที่มีความสามารถในการส่งสัญญาณได้ในช่วงกว้าง (สูงสุด 40 – 400 เมตร) โลโก้สัญลักษณ์ของบลูทูธแสดงดังรูปที่ 11.1 โดยเรามักจะพบโลโก้นี้ในอุปกรณ์อิเล็กทรอนิกส์ที่สามารถสื่อสารด้วยเทคโนโลยีนี้ได้

ปัจจุบันเทคโนโลยีบลูทูธถูกนำมาประยุกต์ใช้งานอย่างแพร่หลาย เช่น การสื่อสารและควบคุมไร้สายระหว่างโทรศัพท์มือถือและหูฟัง/เครื่องเสียง การสื่อสารไร้สายในเมาส์ การส่งข้อมูลระหว่างโทรศัพท์และคอมพิวเตอร์ โดยเทคโนโลยีนี้เป็นเสมือนเทคโนโลยีทางเลือกในการสื่อสารผ่านเครือข่ายไร้สาย ซึ่งเหมาะกับการสร้างเครือข่ายขนาดเล็กที่มีการเชื่อมต่ออุปกรณ์เพียง 1-2 ชิ้น ในระยะเวลาสั้น ๆ โดยความเสถียรของการส่งข้อมูลขึ้นกับทั้งตัวอุปกรณ์และสภาพแวดล้อม

รูปที่ 11.1 โลโก้ของบลูทูธ

11.2 โมดูลสื่อสารไร้สายและโปรแกรมเชื่อมต่อ

บอร์ดโมดูลที่ใช้ในการทดลองนี้คือ บลูทูธโมดูล HC-06 เป็นบอร์ดสำหรับใช้รับ-ส่งสัญญาณกับอุปกรณ์ที่เชื่อมต่อผ่านบลูทูธได้ เช่น โทรศัพท์มือถือ โดยบอร์ดนี้จะสื่อสารกับไมโครคอนโทรลเลอร์ด้วยการสื่อสารแบบอนุกรมที่เรียกว่า USART (Universal Synchronous/Asynchronous Receiver/Transmitter) รูปที่ 11.2 แสดงบอร์ดโมดูลสื่อสารไร้สาย HC-06 ที่ใช้เทคโนโลยีบลูทูธในการเชื่อมต่อ



(ก) (ข)

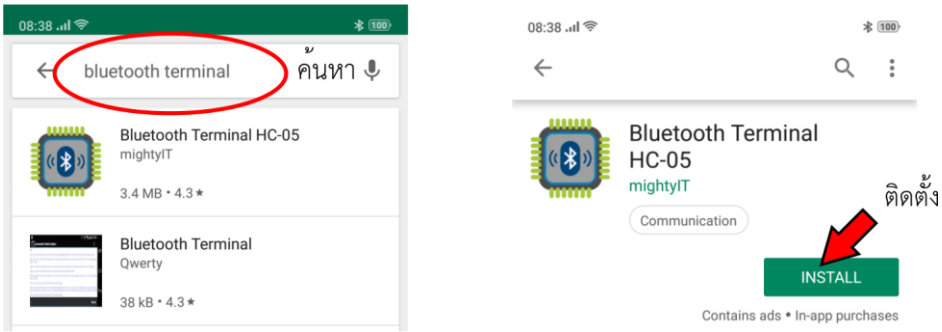
รูปที่ 11.2 บอร์ดโมดูลสื่อสารไร้สาย HC-06 (ก) ด้านหน้าและ (ข) ด้านหลัง

การเชื่อมต่อเพื่อใช้งานโมดูลนี้ในเบื้องต้น ทำได้โดยง่าย คือใช้เพียงขา RXD (Receive) และ TXD (Transmit) ในการรับส่งสัญญาณแบบอนุกรมในระดับ LVTTL (3.3 V) โดยเราจะต้องป้อนไฟเลี้ยง (รองรับไฟเลี้ยงระดับแรงดัน 3.6 – 6 V) ให้แก่บอร์ดด้วย

NodeMCU ที่ภายในมีชิป ESP8266 จะมีการสื่อสารแบบ UART อยู่ภายในบอร์ดอยู่แล้ว โดยเราสามารถส่งสัญญาณจาก NodeMCU ไปยังโมดูลบลูทูธ HC-06 ได้ด้วยการเชื่อมต่อขา RX บนบอร์ด NodeMCU ไปที่ขา TXD บนโมดูลบลูทูธ และขา TX บนบอร์ดไปที่ขา RXD บนโมดูล

สำหรับการรับข้อมูลจาก NodeMCU นั้น ผู้ใช้จะต้องมีโปรแกรมที่ใช้ในการเชื่อมต่อ โดยทั้งนี้เราอาจใช้คอมพิวเตอร์ที่สามารถเชื่อมต่อผ่านบลูทูธได้ (คือโน๊ตบุ๊คหรือคอมพิวเตอร์ตั้งโต๊ะที่มีโมดูลบลูทูธอยู่ภายในหรือใช้โทรศัพท์มือถือก็ได้เช่นกัน) โดยโปรแกรมพื้นฐานที่ขอ

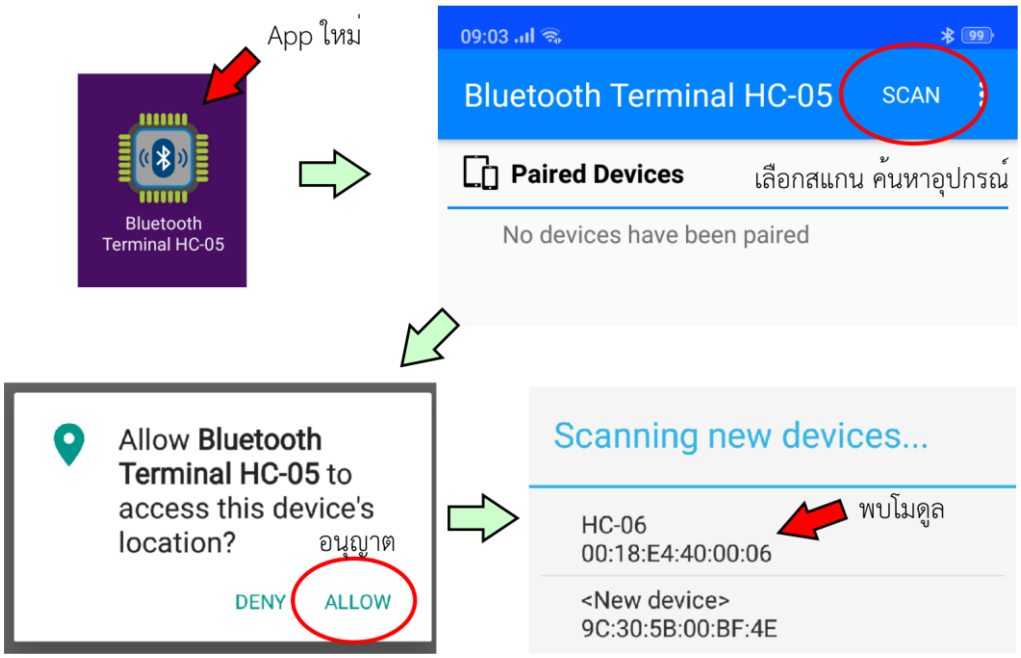
กล่าวถึงและแสดงการทดลองใช้ในที่นี้ คือ โปรแกรม Bluetooth Terminal HC-05 ซึ่งสามารถดาวน์โหลดและติดตั้งได้ฟรี ออกแบบมาสำหรับโมดูลบลูทูธนี้โดยเฉพาะ รูปที่ 11.3 แสดงหน้าต่างที่แสดงการดาวน์โหลดและติดตั้งผ่าน Google Play



รูปที่ 11.3 หน้าต่างแสดงการดาวน์โหลดและติดตั้งโปรแกรม Bluetooth Terminal HC-05 ผ่าน Google Play บนโทรศัพท์มือถือตระกูลแอนดรอยด์

เมื่อติดตั้งโปรแกรมเสร็จสมบูรณ์แล้ว บนโทรศัพท์จะแสดงไอคอนของโปรแกรม ดังรูปที่ 11.4 โดยเมื่อเข้าโปรแกรมแล้วเราจะต้องอนุญาตให้โปรแกรมเข้าถึงการเชื่อมต่อแบบบลูทูธในโทรศัพท์ และการกำหนด (Setting) บนโทรศัพท์จะต้อง Enable การสื่อสารชนิดนี้อยู่ด้วย จากนั้นจึงสแกนค้นหาอุปกรณ์บลูทูธ แล้วจะพบอุปกรณ์เมื่อโมดูลบลูทูธทำงานอยู่บริเวณใกล้เคียง โดยหากมีอุปกรณ์ที่มีชื่อเดียวกันอยู่ในย่านนั้นมากกว่า 1 ตัว (เช่น มีอุปกรณ์ชื่อ HC-06 หลายตัว) เราก็จะต้องสังเกตเบอร์ MAC Address ของอุปกรณ์ที่เราต้องการเชื่อมต่อ โดย MAC Address นี้คือ เบอร์ของฮาร์ดแวร์ ซึ่งเป็นเลขฐานสิบหก 6 ชุด

เมื่อเลือกอุปกรณ์ที่เชื่อมต่อแบบบลูทูธแล้วเราจะต้องทำการกรอกรหัสผ่านซึ่งโดยปกติจะตั้งค่าไว้เป็น “1234” แล้วจึงสามารถเชื่อมต่อได้ ดังแสดงในรูปที่ 11.5 (สำหรับการเปลี่ยนชื่อและพาสเวิร์ดของอุปกรณ์บลูทูธนี้ จะต้องใช้คำสั่ง AT (AT Command) ซึ่งมิกล่าวถึงในการใช้งานเบื้องต้นนี้



รูปที่ 11.4 หน้าต่างการเข้าใช้งานโปรแกรม Bluetooth Terminal HC-05

รูปที่ 11.5 หน้าต่างการเข้าใช้งานโปรแกรม Bluetooth Terminal HC-05

11.3 การทดลองรับส่งข้อมูล

วัตถุประสงค์

1. สามารถอธิบายวิธีการและทำการเชื่อมต่อโมดูลเพื่อสื่อสารไร้สายได้
2. สามารถเขียนโปรแกรมให้ NodeMCU v.3 ส่งข้อมูลแบบไร้สายได้อย่างถูกต้อง
3. สามารถเขียนโปรแกรมให้ NodeMCU v.3 สามารถรับข้อมูลจากการสื่อสารไร้สายและประมวลผลได้อย่างถูกต้อง

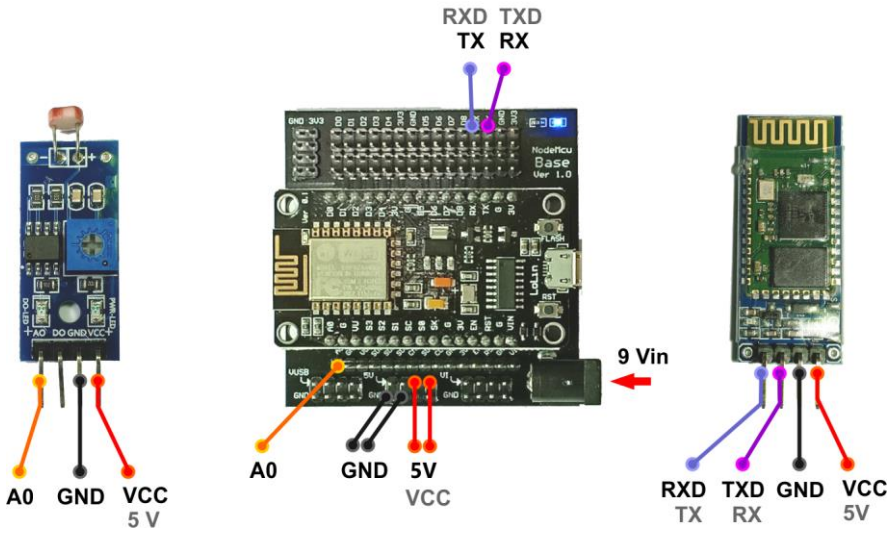
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดโมดูลสื่อสารไร้สาย HC-06	1 บอร์ด
5.	โมดูลวัดความสว่างด้วย LDR	1 บอร์ด
6.	อะแดปเตอร์ 9 V	1 ตัว
7.	สาย USB	1 เส้น
8.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	10 เส้น

วิธีการทดลอง

ตอนที่ 1

1. ต่อวงจรดังรูปที่ 11.6 โดยขอให้สังเกตว่า ขา TX ของ NodeMCU ต่อกับขา RXD ของโมดูล HC-06 และ ขา RX ของ NodeMCU ต่อกับขา RXD ของโมดูล HC-06 และนอกจากนี้เรายังต่อให้ไฟเลี้ยง (5 V) มาจากอะแดปเตอร์ 9 V แทนที่จะใช้ไฟจาก VUSB
2. เขียนโค้ดที่แสดงในหน้าถัดไปแล้วอัปโหลดลงบอร์ด NodeMCU v.3 (ข้อสังเกตโค้ดนี้คือโค้ดเดียวกันกับโค้ดที่แสดงในบทที่ 9)

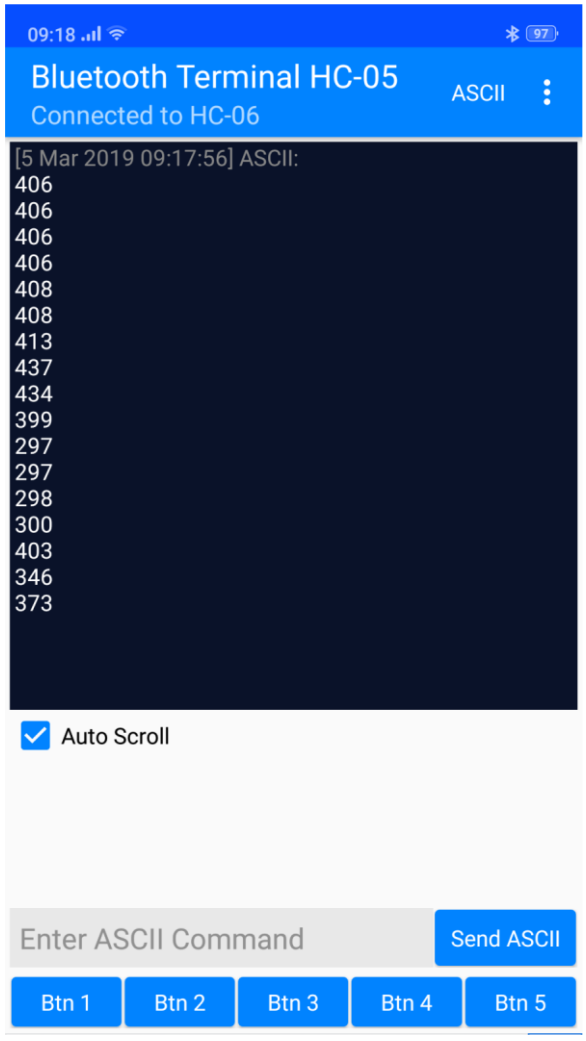


รูปที่ 11.6 การเชื่อมต่อ NodeMCU v.3 กับโมดูล HC-06 และโมดูลวัดความสว่างด้วย LDR

```
// Wireless Communication of NodeMCU ESP8266
// by using Bluetooth Module HC-06
//
// Read LDR by NodeMCU ESP8266
// Result is sent to Bluetooth Module.

// Pin A0 = analog input
int LDR_Sensor = A0;
int data_LDR;

void setup() {
    // Set serial speed to 9600 bps
    Serial.begin(9600);
}

void loop() {
    // Read Analog Data
    data_LDR = analogRead(LDR_Sensor);
    Serial.println(data_LDR);
    delay(500);
}

```

3. ทำการถอดสาย USB แล้วเชื่อมต่อบอร์ด NodeMCU กับโปรแกรม Bluetooth Terminal HC-05 แล้วทดลองปรับค่าความสว่างที่เซนเซอร์ได้รับ

รูปที่ 11.7 แสดงตัวอย่างผลการทดลองที่สังเกตเห็นผ่าน Bluetooth Terminal ซึ่งค่านี้คือค่าที่อ่านได้จากขา A0 ที่ต่อกับเซนเซอร์ LDR นั่นเอง



รูปที่ 11.7 ตัวอย่างหน้าจอผลการทดลองที่แสดงผ่าน Bluetooth Terminal

ตอนที่ 2

ในการทดลองนี้ เราจะทดสอบการส่งค่าจากโทรศัพท์มือถือผ่านบลูทูธ ไปควบคุมไฟแอลอีดีที่อยู่บนบอร์ด NodeMCU (ต่ออยู่กับขา D4) โดยมีขั้นตอนดังนี้

1. ต่อวงจรดังรูปที่ 11.6 โดยอาจนำโมดูลวัดความสว่างแสงออก และเชื่อมต่อ NodeMCU กับคอมพิวเตอร์เพื่อลงโปรแกรมต่อไป

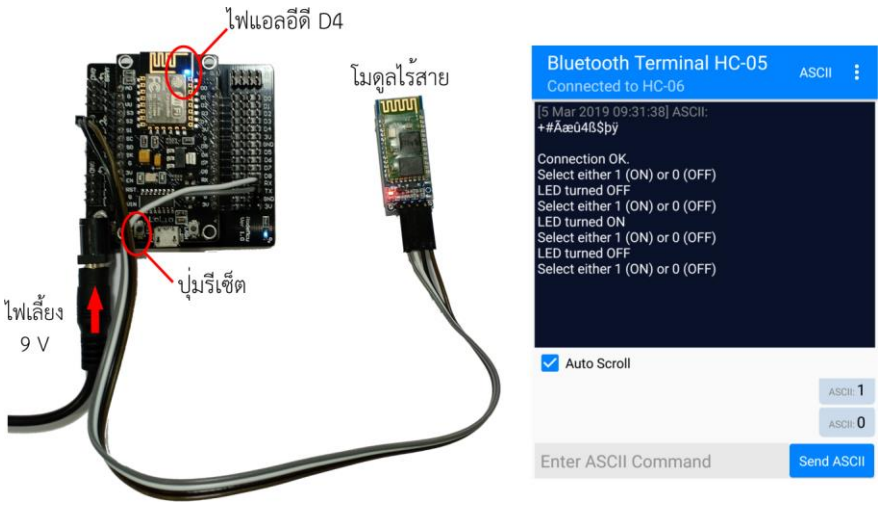
2. เขียนโค้ดที่แสดงด้านล่างนี้แล้วอัปโหลดลง NodeMCU v.3

```
// Wireless Communication of NodeMCU ESP8266
// by using Bluetooth Module HC-05

#define ON LOW
#define OFF HIGH

int LED = D4;  // On board LED

void setup() {
 pinMode(LED, OUTPUT);
 Serial.begin(9600);
 Serial.println("\n\nConnection OK.");
 Serial.println("Select either 1 (ON) or 0
(OFF)");
}

void loop() {

// If data is available on serial port
 if (Serial.available())
    {
      // Data received from bluetooth
      char data_received = Serial.read();

      if (data_received == '0')
      {
       digitalWrite(LED, OFF);
       Serial.println("LED turned OFF");
       Serial.println("Select either 1 (ON) or 0
(OFF)");
      }
      else if (data_received == '1')
```

```
31          {
32           digitalWrite(LED, ON);
33           Serial.println("LED turned ON");
34           Serial.println("Select either 1 (ON) or 0
    (OFF)");
35          }
36       }
37  }
38
```

3. ทำการถอดสาย USB แล้วเชื่อมต่อบอร์ด NodeMCU กับโปรแกรม Bluetooth Terminal HC-05 แล้ว กดปุ่มรีเซ็ตบนบอร์ด NodeMCU จากนั้นจึงทดลองเปลี่ยนสถานะของไฟแอลอีดี โดยส่ง ‘0’ หรือ ‘1’ ไปยังบอร์ด

รูปที่ 11.8 แสดงลักษณะการเชื่อมต่อและหน้าต่างโปรแกรม Bluetooth Terminal HC-05 ในขณะใช้งาน



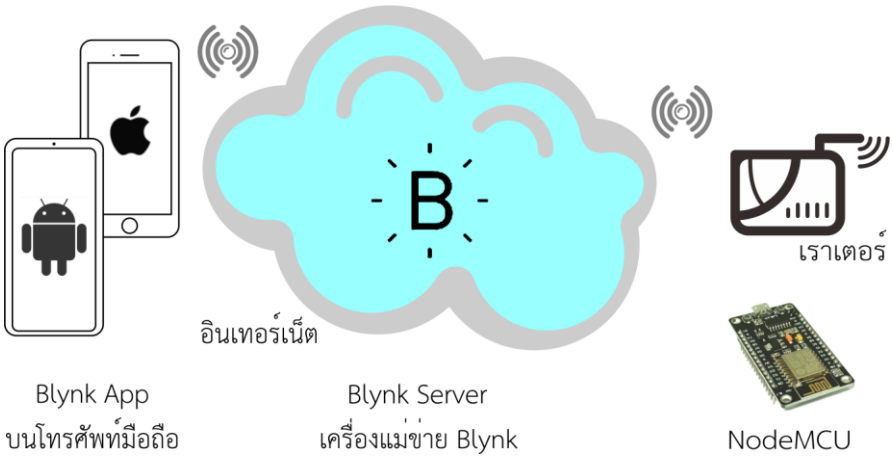
รูปที่ 11.8 ลักษณะการเชื่อมต่อและหน้าต่างโปรแกรม Bluetooth Terminal HC-05 ในขณะใช้งานตามการทดลองตอนที่ 2

บทที่ 12

การสร้างแอปพลิเคชันเพื่อควบคุมสิ่งประดิษฐ์ด้วย Blynk

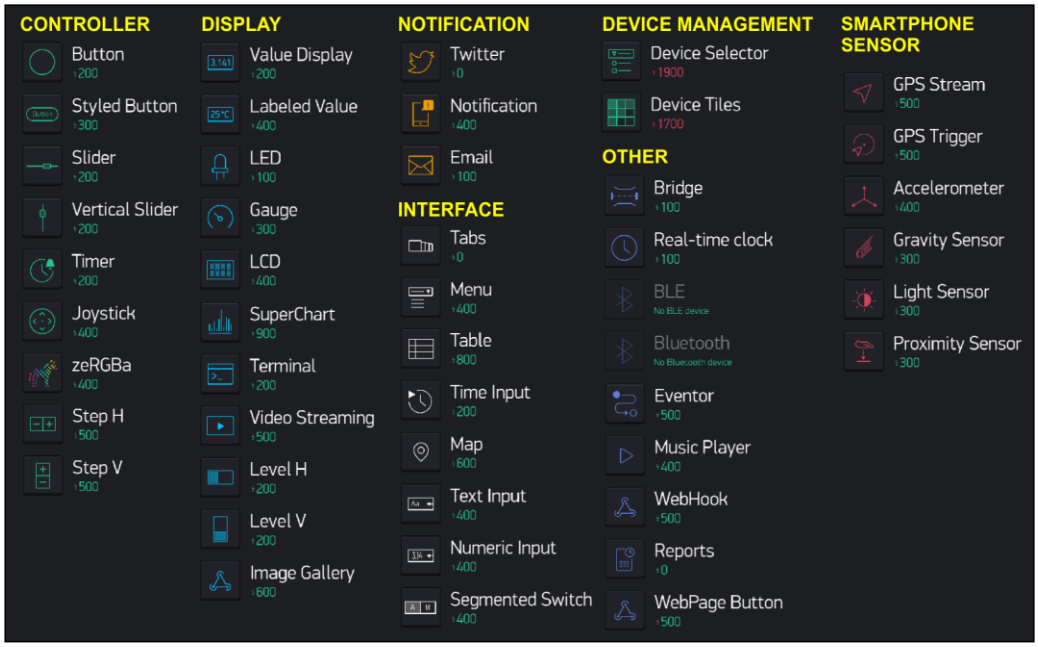
12.1 แนะนำ Blynk App

Blynk เป็นชื่อโดยรวมของการบริการให้ผู้ใช้งานได้ใช้งานเครื่องแม่ข่าย คือ Blynk Server ที่เป็น IoT Cloud ซึ่ง ถูกพัฒนามาจากภาษา Java ทำให้สามารถทำงานภายใต้ระบบปฏิบัติการที่หลากหลาย เช่น Windows, Mac หรือ Linux โดยเครื่องแม่ข่าย (Blynk Server) พัฒนาเป็นแบบเปิด (open-source) ภายใต้ลิขสิทธิ์แบบ GNU ทำให้เราสามารถนำ Blynk ไปใช้งานประกอบการสร้างนวัตกรรมเพื่อการค้า แก้ไข ดัดแปลง เผยแพร่ หรือแจกจ่ายได้ ซึ่งสามารถดูภาพรวมของระบบได้ตามรูปที่ 12.1



รูปที่ 12.1 ภาพรวมของการเชื่อมต่อผ่าน Blynk Server

Blynk App คือ แอปพลิเคชันสำเร็จรูปที่ใช้สำหรับงานที่เกี่ยวกับอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things, IoT) ที่ทำให้เราสามารถเชื่อมต่ออุปกรณ์ต่าง ๆ เข้ากับอินเทอร์เน็ตในลักษณะการเชื่อมต่อเครื่องแม่ข่าย (Server) ไปยังอุปกรณ์ลูกข่าย (Client) เช่น Arduino, ESP-8266, ESP-32, NodeMCU และ Raspberry Pi ซึ่งแอปพลิเคชัน Blynk สามารถใช้งานได้ฟรีและใช้งานได้ทั้งบนระบบปฏิบัติการ IOS และ Android รูปที่ 12.2 แสดงภาพรายการอุปกรณ์ต่าง ๆ ที่สามารถเชื่อมต่อ แสดงผล และ/หรือ ควบคุมด้วย Blynk App ได้ โดยเริ่มต้นหลังจากสมัครเข้าใช้งาน เราจะได้รับ “Energy” ซึ่งเปรียบเสมือนเงินในโปรแกรมนี้ ในการเรียกใช้งานอุปกรณ์แต่ละตัว เราจะต้องแลกด้วย “Energy” และหาก “Energy” นี้ไม่เพียงพอ เราก็สามารถซื้อเพิ่มเติมได้ภายหลัง

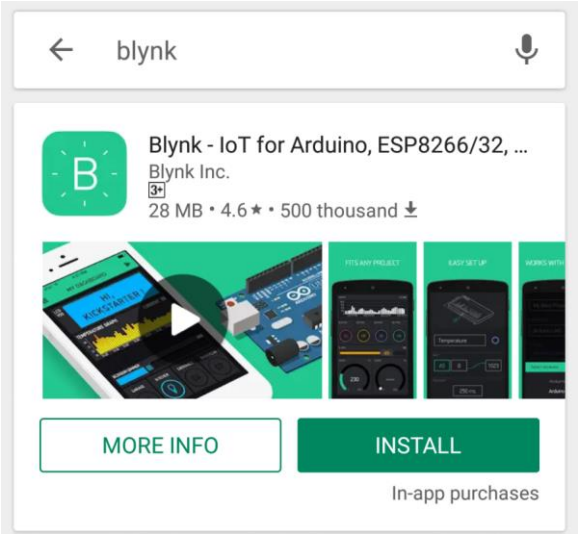


รูปที่ 12.2 รายการอุปกรณ์ต่าง ๆ ที่เชื่อมต่อ แสดงผล และ/หรือ ควบคุมด้วย Blynk App ได้

12.2 การเริ่มต้นใช้งาน Blynk App

ขั้นตอนการสมัครและเปิดใช้งานแอปพลิเคชัน Blynk App มีดังนี้

1. ค้นหาและติดตั้งแอปพลิเคชัน Blynk – IoT for Arduino ESP8266/32, Raspberry Pi (เรียกย่อ ๆ ว่า Blynk App) บนโทรศัพท์มือถือที่จะใช้ทดสอบ (ใช้ได้ทั้งระบบ Android และ IOS แต่ในการทดลองนี้เราจะแสดงเฉพาะหน้าจอบนระบบ Android เท่านั้น) ดังรูปที่ 12.3



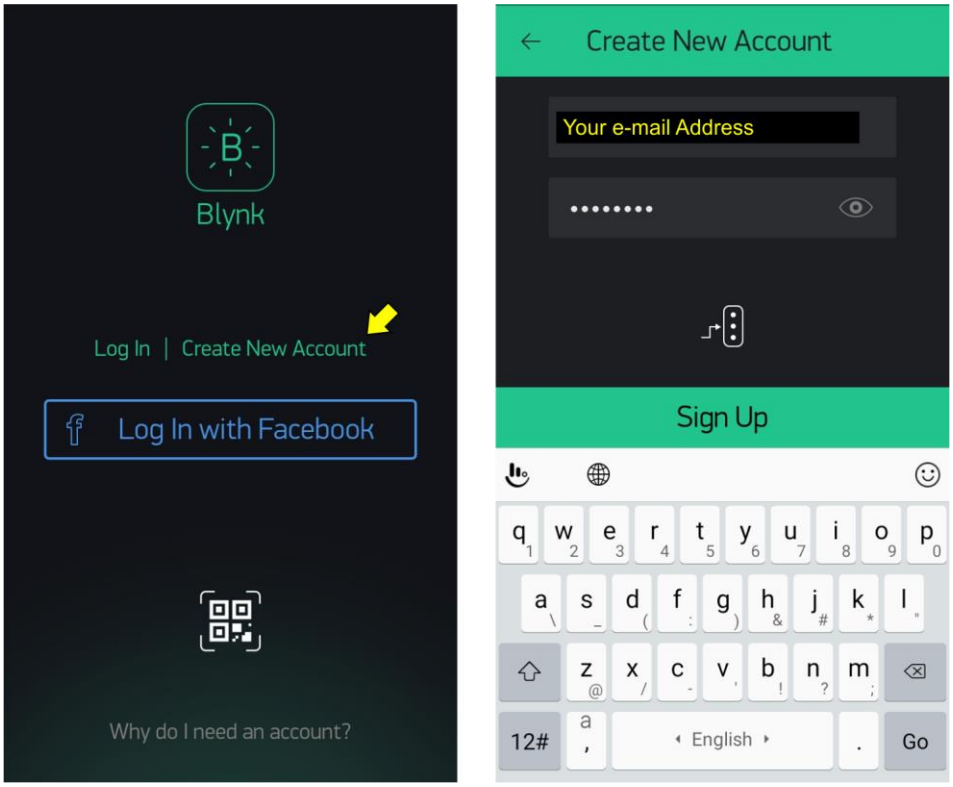
รูปที่ 12.3 หน้าต่างแสดงการดาวน์โหลดและติดตั้ง Blynk App บนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์

2. เปิดโปรแกรม Blynk App แล้วทำการสร้างบัญชีผู้ใช้ (account) แล้วเข้าใช้ (sign up) ดังรูปที่ 12.4

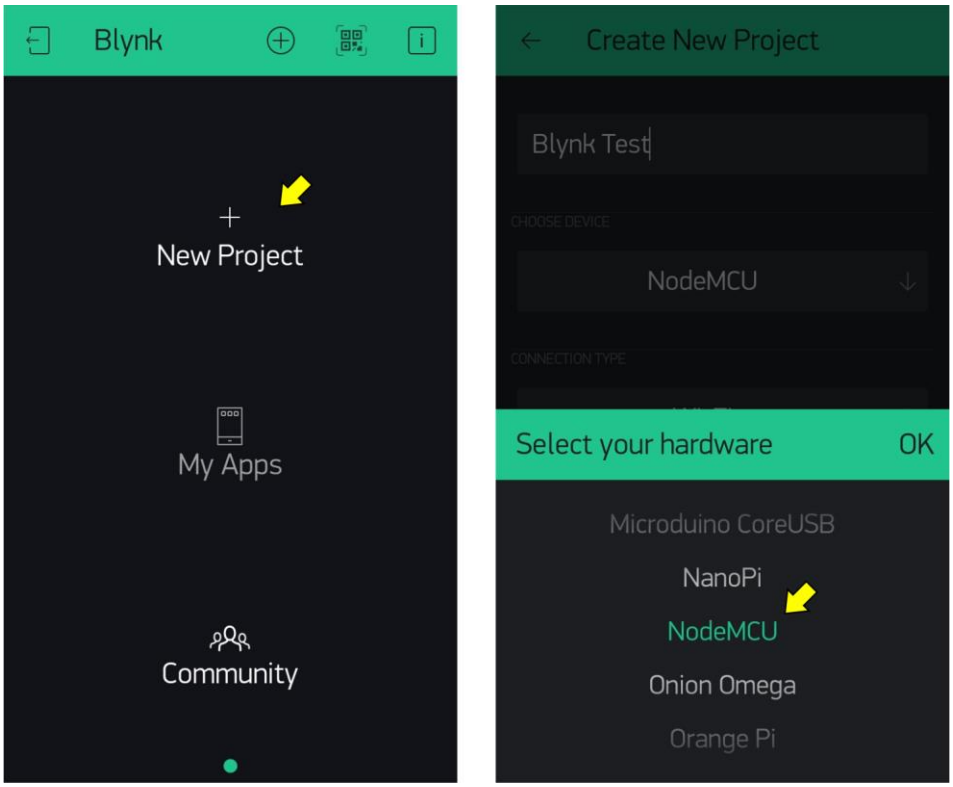
3. เมื่อเชื่อมต่อ Blynk Server ได้แล้ว โปรแกรมจะพร้อมให้เราสร้างโปรเจคใหม่ โดยการเลือก ‘+ New Project’ แล้วตั้งชื่อโปรเจค เป็น **Blynk Test** จากนั้นจึงเลือกอุปกรณ์เป็น **NodeMCU** และเลือกการเชื่อมต่อเป็นแบบ **Wi-Fi** (ดังแสดงในรูปที่ 12.5) จากนั้นจึงกดปุ่ม

Create ก็จะมีการส่งอีเมลรหัส Token มาให้ในอีเมลที่ลงทะเบียนไว้ เพื่อให้ผู้ใช้นำไปเป็นรหัสที่ใช้ในการเชื่อมต่อระหว่าง NodeMCU และ Blynk ต่อไป

4. ภายใน Blynk จะมีปุ่มต่าง ๆ ให้เลือกสร้าง App ได้ตามที่ผู้ใช้ต้องการ



รูปที่ 12.4 ขั้นตอนการสร้างบัญชีและเข้าใช้ Blynk App

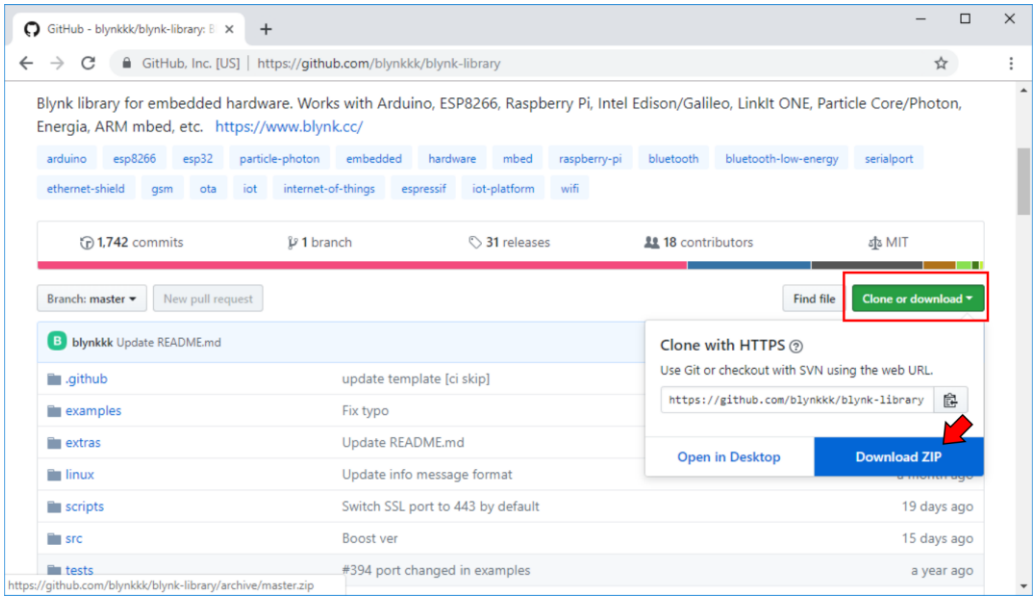


รูปที่ 12.5 ขั้นตอนการโปรเจคใหม่และเลือกฮาร์ดแวร์ที่เชื่อมต่อ Blynk App

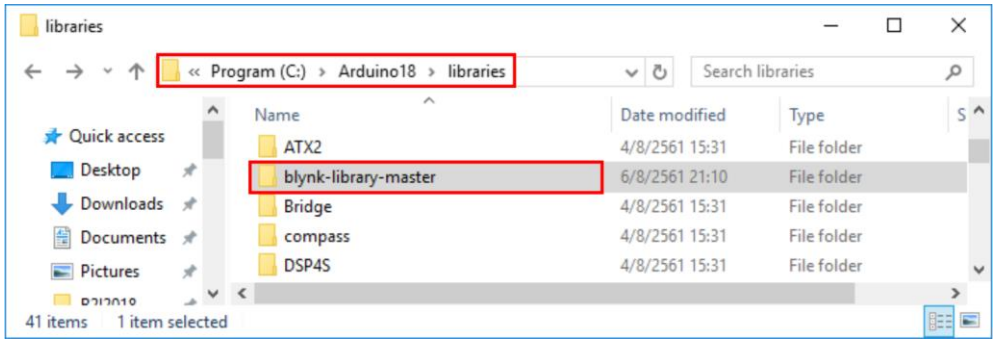
12.3 การติดตั้งไลบรารีเพื่อเชื่อมต่อกับ NodeMCU

การใช้งานแอปปลิเคชั่น Blynk กับ NodeMCU เราจำเป็นต้องติดตั้งไลบรารีบนโปรแกรมที่ใช้ในการเขียนโปรแกรมของไมโครคอนโทรลเลอร์ (ในที่นี้คือ Arduino IDE 1.8.8 IoT) เพื่อให้สามารถเรียกใช้งานคำสั่งเชื่อมต่อได้ ซึ่งสามารถดาวน์โหลดไลบรารี Blynk ได้จาก https://github.com/blynkkk/blynk-library ดังแสดงในรูปที่ 12.6

เมื่อดาวน์โหลดแล้วก็นำมาคัดลอก (ติดตั้ง) ลงในโปรแกรม Arduino IDE ที่ C:\Arduino18\libraries ดังรูปที่ 12.7



รูปที่ 12.6 หน้าต่างแสดงการดาวน์โหลดไลบรารี Blynk



รูปที่ 12.7 หน้าต่างแสดงโฟลเดอร์ที่ติดตั้งไลบรารี Blynk

12.4 การทดลองควบคุมสิ่งประดิษฐ์ผ่าน Blynk App

วัตถุประสงค์

1. สามารถเชื่อมต่อ NodeMCU กับ Blynk App ได้
2. สามารถเขียนโปรแกรมประยุกต์ใช้งานควบคุมด้วย Blynk App ได้

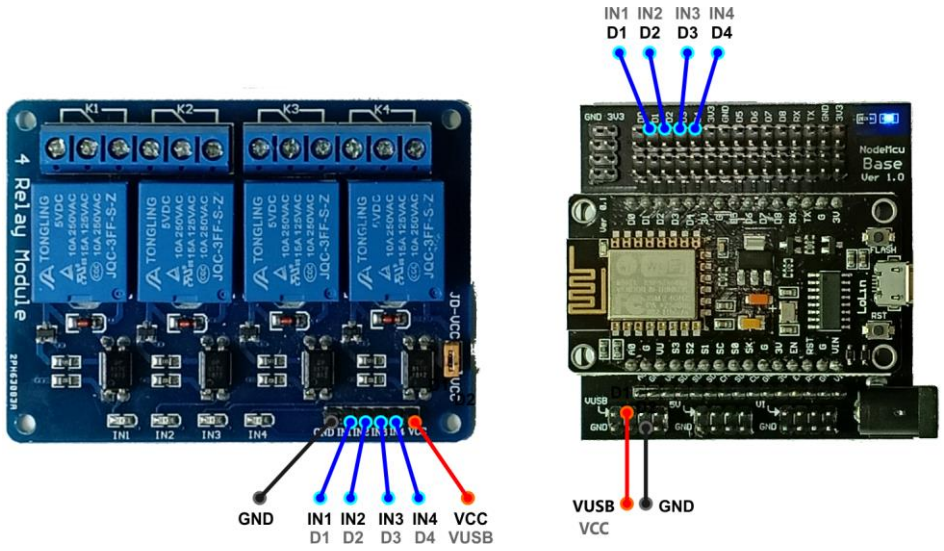
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดรีเลย์ชนิด 4 ช่อง	1 บอร์ด
5.	เซอร์โวมอเตอร์	1 ตัว
6.	อะแดปเตอร์ 9 V	1 ตัว
7.	สาย USB	1 เส้น
8.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	6 เส้น
9.	สายต่อวงจร (สายจัมพ์ ผู้-เมีย)	3 เส้น

วิธีการทดลอง

ตอนที่ 1 การสร้างแอปพลิเคชันบน Blynk App เพื่อควบคุมรีเลย์

1. ต่อวงจรตามรูปที่ 12.8 โดยให้ต่อขา D1, D2, D3, D4 ของ NodeMCU เข้ากับขา IN1, IN2, IN3, IN4 ของโมดูลรีเลย์ ตามลำดับ



รูปที่ 12.8 การต่อวงจรเพื่อควบคุมรีเลย์

2. เขียนโค้ดโปรแกรมดังแสดงข้างล่าง โดยในโค้ดมีตัวแปรข้อความ (สตริง, String) ที่ผู้เขียนต้องเปลี่ยนแปลงแก้ไข คือ

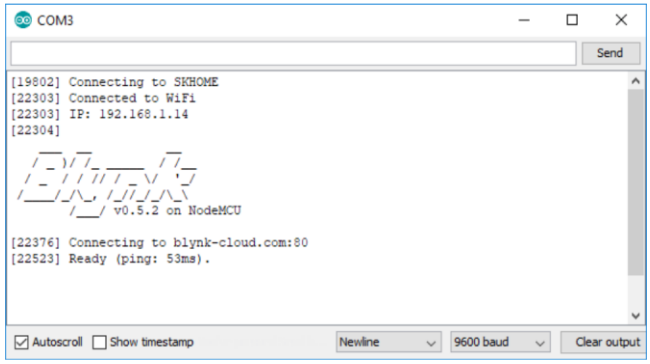
- auth คือรหัส Auth Token ที่ Blynk App ส่งให้ทางอีเมล สำหรับโปรเจคที่สร้าง (เป็นรหัสเลขฐาน 16) ตัวอย่างเช่น “960110e6cf014314bcb5fbdf05448a1e”
- ssid คือชื่อ WiFi ที่ใช้เชื่อมต่อ เช่น “WiFi NU”
- pass คือรหัส WiFi ที่ตั้งไว้ หากไม่มี ก็ใส่เป็นสตริงว่าง “”

```
1  // Control NodeMCU ESP8266 by Blynk
2  // Switch Relays
3
4  // Comment this out to disable prints
5  #define BLYNK_PRINT Serial
6
7  #include <ESP8266WiFi.h>
8  #include <BlynkSimpleEsp8266.h>
9
10 char auth[] = "9c687abf4897467992e46e7249af7209";
11 char ssid[] = "XXXXXXX";
12 char pass[] = "XXXXXXX";
```

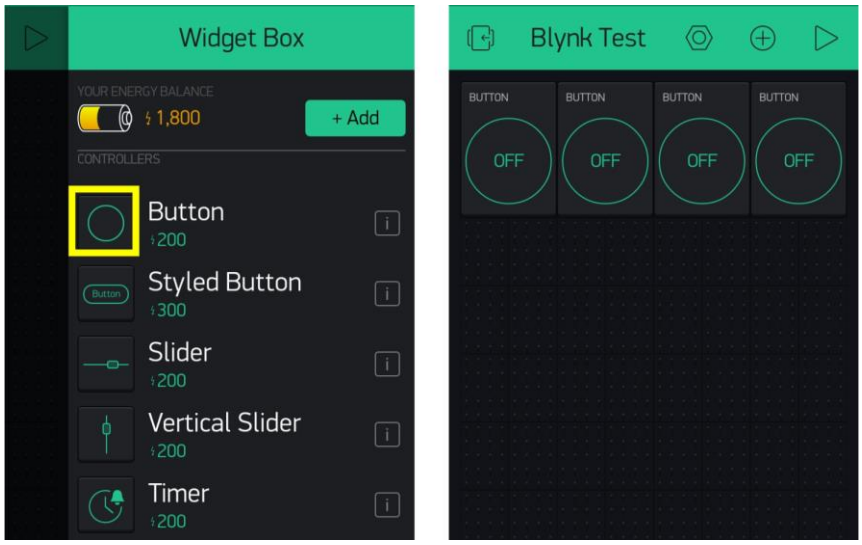
```
13
14 void setup() {
15     Serial.begin(9600);
16     Blynk.begin(auth, ssid, pass);
17 }
18
19 void loop() {
20     Blynk.run();
21     delay(100);
22 }
23
```

3. อัปโหลดโปรแกรมลงใน NodeMCU เพื่อให้ฮาร์ดแวร์รอรับคำสั่งเชื่อมต่อ โดยหากการเชื่อมต่อสำเร็จดี Blynk ก็จะส่งข้อมูลผ่าน Serial Monitor ดังแสดงในรูปที่ 12.9

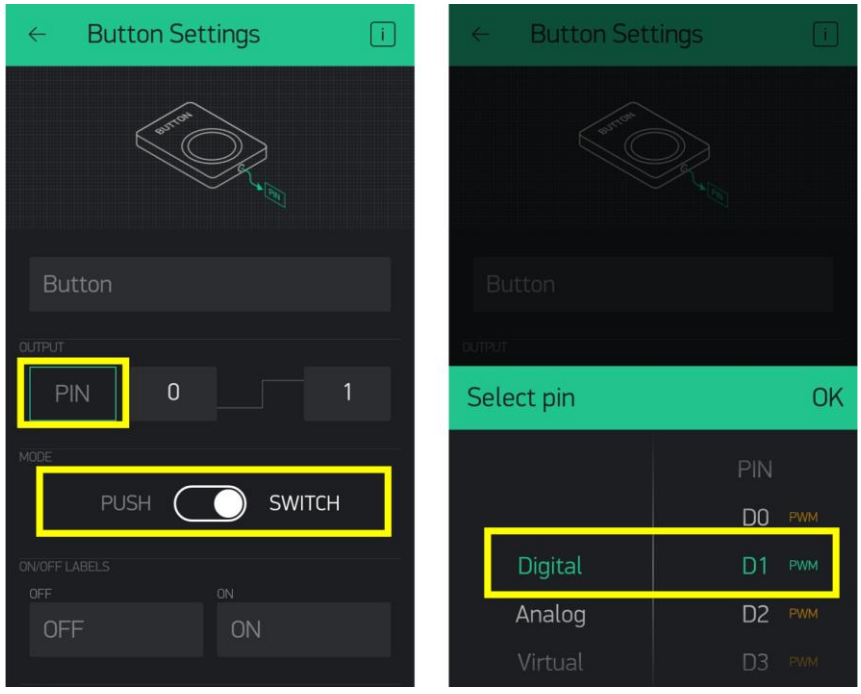
4. เปิด Blynk App สร้างปุ่ม (Button) 4 ปุ่ม (ดังรูปที่ 12.10) โดยกำหนด OUTPUT PIN เป็น D1, D2, D3 และ D4 ตามลำดับ และ เลือกโหมดของปุ่มเป็นแบบ SWITCH (ดังรูปที่ 12.11) โดยทำเหมือนกันทั้ง 4 ตัว



รูปที่ 12.9 หน้าต่างแสดงข้อความการเชื่อมต่อระหว่าง NodeMCU และ Blynk Server ที่แสดงผ่าน Serial Monitor

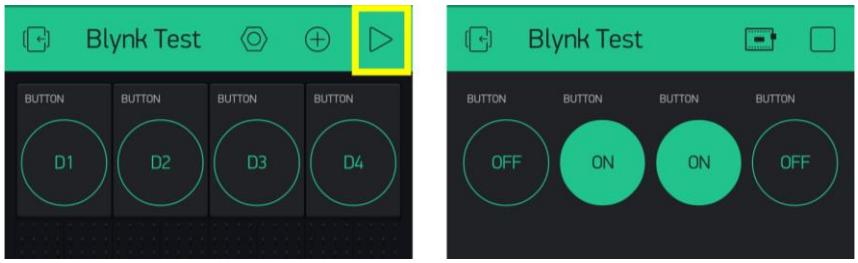


รูปที่ 12.10 ขั้นตอนการสร้างปุ่ม 4 ตัวสำหรับการควบคุมรีเลย์



รูปที่ 12.11 ขั้นตอนการกำหนดการเชื่อมต่อของปุ่ม 4 ตัวสำหรับการควบคุมรีเลย์

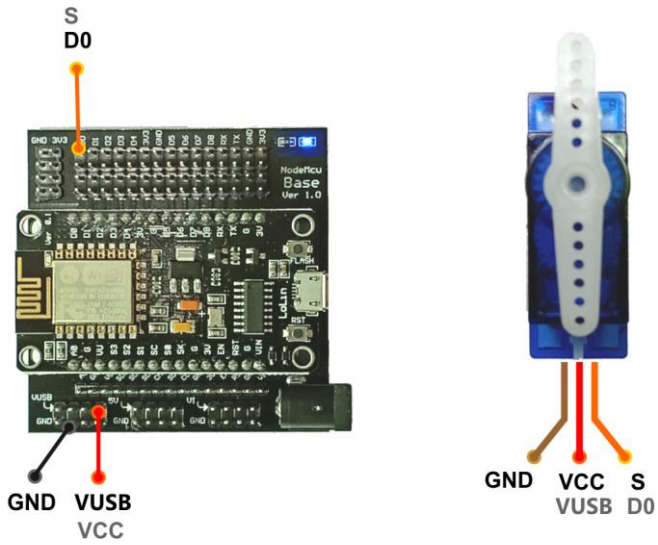
5. รันโปรแกรม ดังรูปที่ 12.12 แล้วทดสอบการควบคุมรีเลย์ โดยสังเกตไฟแอลอีดีที่บอร์ดรีเลย์ โดยบอร์ดรีเลย์ทำงานแบบ Active low ดังนั้น สถานะ ‘ON’ ใน Blynk หมายถึงไฟแอลอีดีไม่ติด



รูปที่ 12.12 การรันโปรแกรมควบคุมรีเลย์ผ่าน Blynk App

ตอนที่ 2 การสร้างแอปพลิเคชันบน Blynk App เพื่อควบคุมเซอร์โวมอเตอร์

1. ต่อวงจรดังรูปที่ 12.13



รูปที่ 12.13 การเชื่อมต่อ NodeMCU กับเซอร์โวมอเตอร์ เพื่อควบคุมผ่าน Blynk App

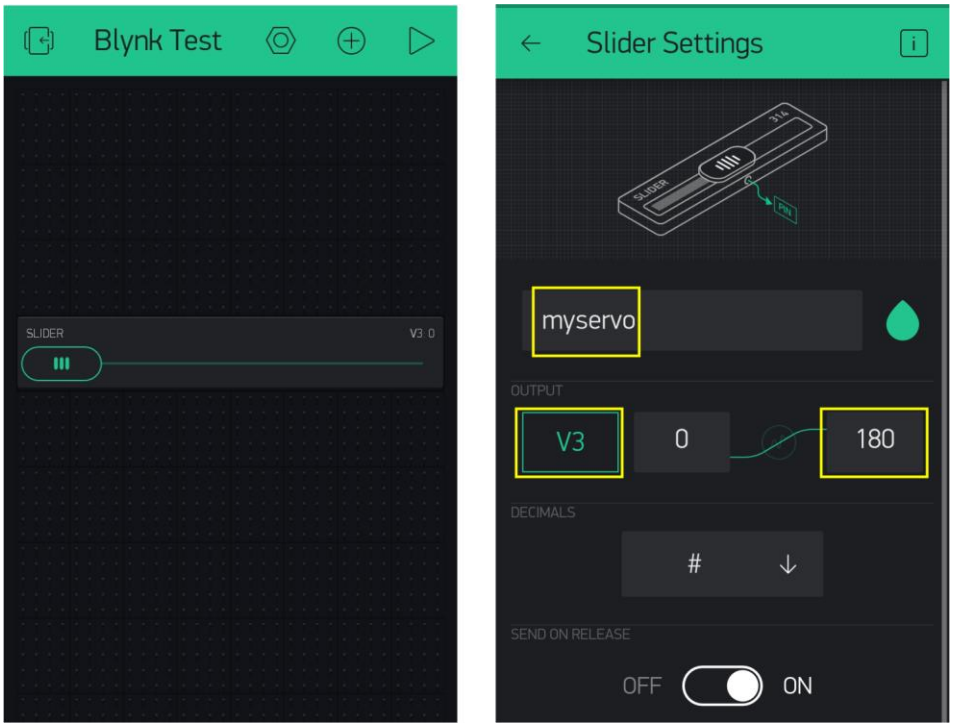
2. เขียนโค้ดโปรแกรมดังที่แสดงข้างล่างนี้ โดยอาจจะต้องแก้ไขค่าในสตริง auth ซึ่งก็คือรหัส Auth Token ที่ Blynk App ส่งให้ทางอีเมล หลังจากกดปุ่มสร้างโปรเจคใหม่

```
// Control NodeMCU ESP8266 by Blynk
// Control Servo motor

// Comment this out to disable prints
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Servo.h>

Servo myservo;
char auth[] = "9c687abf4897467992e46e7249af7209";
char ssid[] = "XXXXXXX";
char pass[] = "XXXXXXX";

void setup() {
    Serial.begin(9600);
    Blynk.begin(auth, ssid, pass);
    myservo.attach(D0);
}

// This function gets called every time the Virtual Button on V3 changes state

BLYNK_WRITE(V3) {
    myservo.write(param.asInt());
}

void loop() {
    Blynk.run();
    delay(100);
}

```

ในโค้ดนี้จะมีการรับค่าจากปุ่ม V3 ใน Blynk App มา แล้วส่งไปยังเซอร์โวมอเตอร์ชื่อ myservo ผ่านฟังก์ชัน BLYNK_WRITE ซึ่งจะถูกเรียกทุกครั้งที่มีการเปลี่ยนแปลงค่า V3

3. อัปโหลดโปรแกรมลงใน NodeMCU เพื่อให้ฮาร์ดแวร์รอรับคำสั่งเชื่อมต่อ

4. เปิด Blynk App โดยอาจสร้างโปรเจคใหม่ (ขอ Token ใหม่) หรือลบอุปกรณ์ในโปรเจคเดิมก็ได้ จากนั้นจึงสร้างแถบเลื่อน (Slider) 1 แถบ แล้วขยายขนาด (ดังรูปที่ 12.14) โดยกำหนดชื่อเป็น myservo, OUTPUT PIN เป็นปุ่มเสมือน (Virtual pin) V3 และเปลี่ยนค่าสูงสุดจาก 1023 เป็น 180 (ซึ่งคือมุมการหมุนสูงสุดของเซอร์โวมอเตอร์) ดังรูปที่ 12.14 จากนั้นจึงรันโปรแกรมเพื่อทดสอบการควบคุมเซอร์โวมอเตอร์

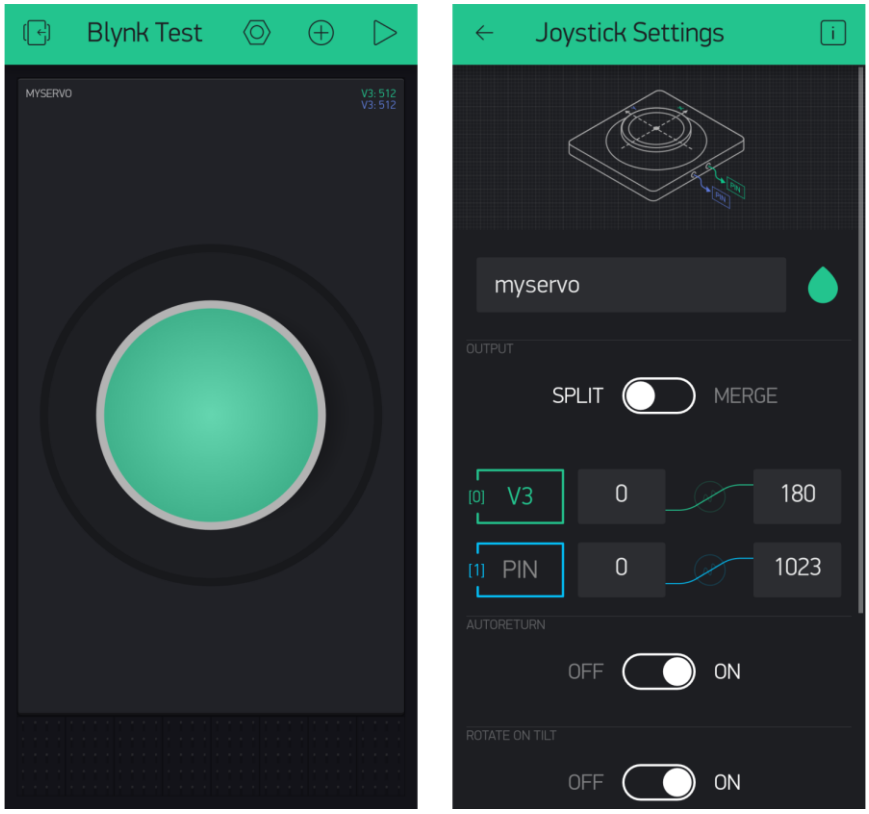


รูปที่ 12.14 ขั้นตอนการสร้างและกำหนดคุณสมบัติของแถบเลื่อน
เพื่อควบคุมเซอร์โวมอเตอร์ผ่าน Blynk App

แบบฝึกหัดท้ายการทดลอง

การสร้างแอปพลิเคชันบน Blynk App เพื่อควบคุมเซอร์โวมอเตอร์ผ่านจอยสติ๊ก

หากเราจะใช้วงจรควบคุมเซอร์โวมอเตอร์เดิม (รูปที่ 12.13) โดยเปลี่ยนวิธีการควบคุมจากแถบเลื่อนเป็นจอยสติ๊ก (Joystick) เราสามารถทำได้โดยตั้งค่าการเชื่อมต่อ ปุ่มเสมือน V3 กับจอยสติ๊กที่เรียกขึ้นมาใช้แทนแถบเลื่อน ดังรูปที่ 12.15 ซึ่งเราสามารถเลือกว่าจะควบคุมแบบแยกแกน (SPLIT คือ แยกแกนตั้งและแกนนอน) หรือ รวมแกน (MERGE) ก็ได้ สำหรับค่าสูงสุดของปุ่ม เราจะกำหนดให้เป็น 180 เนื่องจากเซอร์โวที่ใช้ รับค่าเป็นมุมองศา และมีค่าสูงสุดประมาณ 180° เท่านั้น



รูปที่ 12.15 การตั้งค่าจอยสติ๊กใน Blynk App เพื่อควบคุมเซอร์โวมอเตอร์

บทที่ 13

การสร้างแอปพลิเคชันเพื่อรับค่าและแสดงผลด้วย Blynk

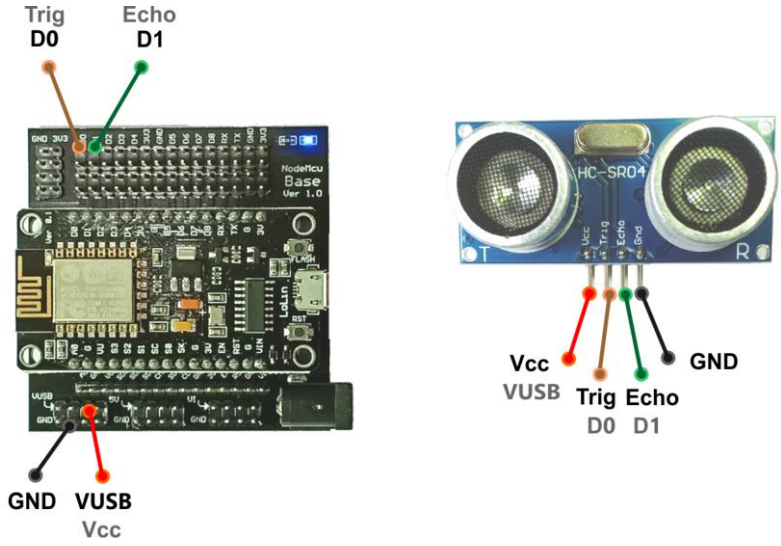
ในบทนี้ จะเป็นการประยุกต์ใช้งาน NodeMCU ร่วมกับ Blynk App เพิ่มเติมจากบทที่แล้ว โดยจะแบ่งเป็น 3 ตอน คือ การสร้างแอปพลิเคชันบน Blynk App เพื่อแสดงค่าระยะทาง การสร้างแอปพลิเคชันบน Blynk App เพื่อแสดงค่าความสว่าง และ การสร้างแอปพลิเคชันบน Blynk App เพื่อควบคุมรีเลย์จากความสว่าง โดยการทดลองที่แสดงในบทนี้ มีวัตถุประสงค์เพื่อให้ผู้เรียนสามารถเชื่อมต่อ NodeMCU กับแอพพลิเคชัน Blynk และรับค่าต่าง ๆ จากเซนเซอร์ได้ และผู้เรียนสามารถเขียนโปรแกรมประยุกต์ใช้งานควบคุมร่วมกันระหว่าง NodeMCU และแอปพลิเคชัน Blynk App ได้

อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดโมดูลอัลตราโซนิกส์ HC-SR04	1 บอร์ด
5.	บอร์ดโมดูลโมดูลวัดแสงแบบแอนะล็อกด้วย LDR	1 บอร์ด
6.	บอร์ดรีเลย์ชนิด 4 ช่อง	1 บอร์ด
7.	อะแดปเตอร์ 9 V	1 ตัว
8.	สาย USB	1 เส้น
9.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	6 เส้น

13.1 การสร้างแอปพลิเคชันบน Blynk App เพื่อแสดงค่าระยะทาง

1. ต่อวงจรตามรูปที่ 13.1 โดยให้ต่อขา D0 และ D1 ของ NodeMCU เข้ากับขา TRIG และ ECHO ของบอร์ดโมดูลอัลตราโซนิกส์ HC-SR04 ตามลำดับ



รูปที่ 13.1 การเชื่อมต่อเพื่อใช้งานโมดูลอัลตราโซนิกส์

2. เขียนโค้ดโปรแกรมดังแสดงข้างล่าง (โดยอาจจะต้องแก้ไข auth, ssid และ pass)

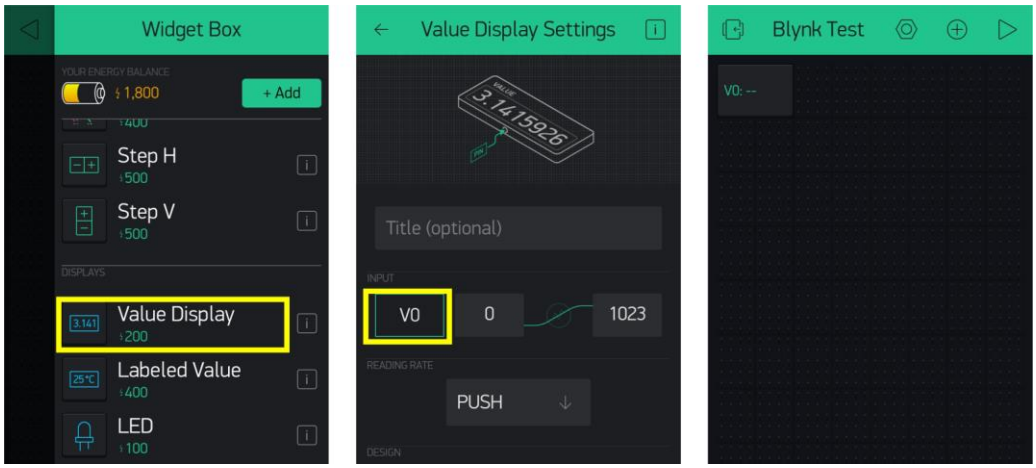
```
// Control NodeMCU ESP8266 by Blynk
// Read and display value from ultrasonic sensor

// Comment this out to disable prints
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Ultrasonic.h>

// ultrasonic(Trigger PIN, Echo PIN)
Ultrasonic ultrasonic(D0,D1);
```

```
13
14 char auth[] = "9c687abf4897467992e46e7249af7209";
15 char ssid[] = "XXXXXXX";
16 char pass[] = "XXXXXXX";
17
18 void setup() {
19     Serial.begin(9600);
20     Blynk.begin(auth, ssid, pass);
21 }
22
23 void sendUltra() {
24     unsigned long ultra_value =
   ultrasonic.Ranging(CM);
25     Blynk.virtualWrite(V0, ultra_value);
26 }
27
28 void loop() {
29     Blynk.run();
30     delay(100);
31     sendUltra();
32 }
33
```

3. อัปโหลดโปรแกรมลงใน NodeMCU เพื่อให้ฮาร์ดแวร์รอรับคำสั่งเชื่อมต่อ

4. เปิด Blynk App โดยอาจสร้างโปรเจคใหม่ (ขอ Token ใหม่) หรือลบอุปกรณ์ในโปรเจคเดิมก็ได้ จากนั้นจึงสร้างตัวแสดงค่า (Value Display) 1 ตัว แล้วขยายขนาด (ดังรูปที่ 13.2) โดยกำหนด INPUT PIN เป็นปุ่มเสมือน V0 ดังรูปที่ 13.2

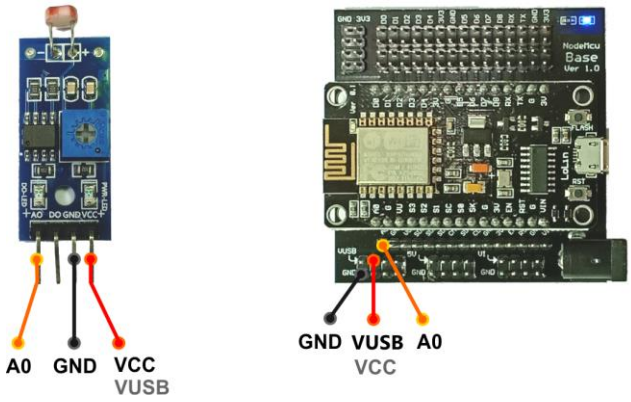
5. รันโปรแกรมเพื่อทดสอบการรับค่าเข้าจากโมดูลอัลตราโซนิกส์ และทดลองเปลี่ยนระยะห่างเพื่อตรวจสอบความถูกต้องของการอ่านค่าและการส่งข้อมูล



รูปที่ 13.2 ขั้นตอนการสร้างและกำหนดคุณสมบัติของ Value Display เพื่อแสดงค่าจากโมดูลอัลตราโซนิกส์

13.2 การสร้างแอปพลิเคชันบน Blynk App เพื่อแสดงค่าความสว่าง

1. ต่อวงจรตามรูปที่ 13.3 โดยให้ต่อขา A0 ของ NodeMCU ต่อเข้ากับขา A0 ของบอร์ดโมดูลวัดค่าความสว่างแสงแบบแอนะล็อก



รูปที่ 13.3 การเชื่อมต่อเพื่อใช้งานโมดูลวัดแสงแบบแอนะล็อก

2. เขียนโค้ดโปรแกรมดังแสดงข้างล่าง (โดยอาจจะต้องแก้ไข auth, ssid และ pass)

```
// Control NodeMCU ESP8266 by Blynk
// Read value from LDR

// Comment this out to disable prints
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "9c687abf4897467992e46e7249af7209";
char ssid[] = "XXXXXXX";
char pass[] = "XXXXXXX";

int LDR_Sensor = A0;

float Cal_illuminance() {
    float illu_LDR;
    float volt_LDR;
    int data_LDR;
    data_LDR = analogRead(LDR_Sensor);
    volt_LDR = (3.3/1024)*data_LDR;
    illu_LDR = (42.0 * pow(volt_LDR, -3.15));
    return illu_LDR;
}

void setup() {
    Serial.begin(9600);
    Blynk.begin(auth, ssid, pass);
}

void loop() {
    float illu_LDR;
    illu_LDR = Cal_illuminance();
    Blynk.virtualWrite(V0, String(illu_LDR));
    Blynk.run();
    delay(100);
}

```

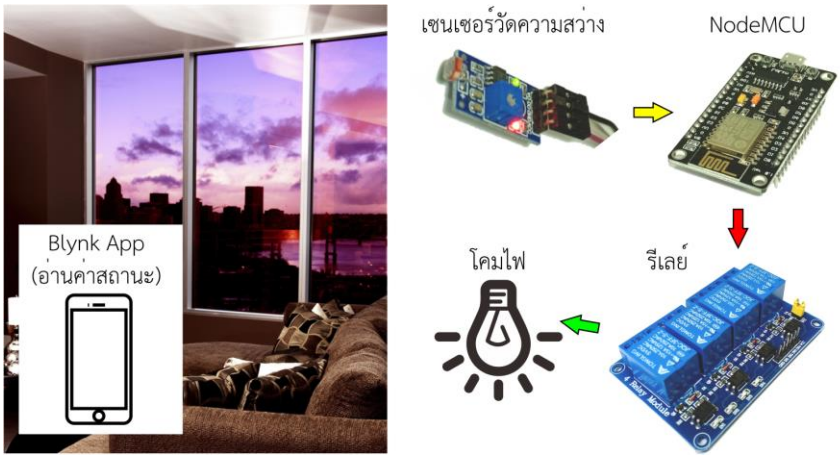
3. อัปโหลดโปรแกรมลงใน NodeMCU เพื่อให้ฮาร์ดแวร์รอรับคำสั่งเชื่อมต่อ

4. เปิด Blynk App โดยอาจสร้างโปรเจคใหม่ (ขอ Token ใหม่) หรือลบอุปกรณ์ในโปรเจคเดิมก็ได้ จากนั้นจึงสร้างตัวแสดงค่า (Value Display) 1 ตัว โดยกำหนด INPUT PIN เป็นปุ่มเสมือน V0 ดังรูปที่ 13.2 (เหมือนกับกับการทดลองตอนที่ 1)

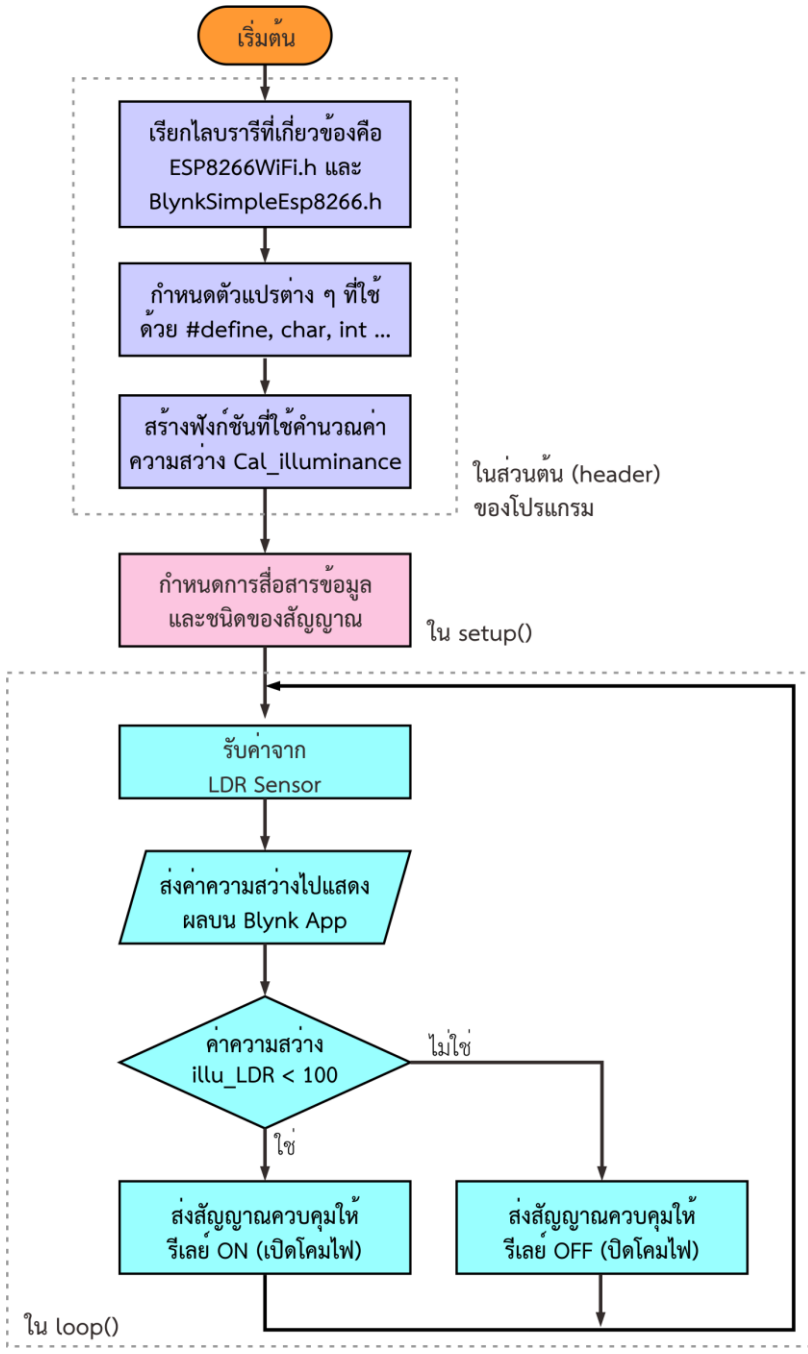
5. รันโปรแกรมเพื่อทดสอบการรับค่าเข้าจากโมดูลวัดความสว่างแสง โดยสังเกตว่า ค่าที่แสดงเป็นค่าทศนิยมในหน่วยลักซ์ที่คำนวณจากโค้ดที่เขียนในฟังก์ชัน Cal_illuminance (ดูบทที่ 9 สำหรับการเทียบวัดค่าความสว่างแสง)

13.3 การสร้างแอปพลิเคชันบน Blynk App เพื่อควบคุมรีเลย์จากความสว่าง

ในการทดลองตอนนี้ เราจะสมมติว่า เราต้องการสร้าง Blynk App ที่ทำหน้าที่เปิด-ปิดไฟอัตโนมัติเมื่อมีความสว่างน้อยเกินไป โดยกำหนดให้ค่าความสว่างต้องมากกว่า **100 ลักซ์** นั่นคือ หากแสงที่ตรวจวัดได้มีค่าความสว่างน้อยกว่า 100 ลักซ์แล้ว รีเลย์จะควบคุมให้เปิดโคมไฟ (ดูรูปที่ 13.4) ขั้นตอนการทดลองสร้าง เริ่มจากการกำหนดแผนผังการทำงานของโปรแกรมเสียก่อน แผนผังอาจเขียนได้ดังแสดงในรูปที่ 13.5



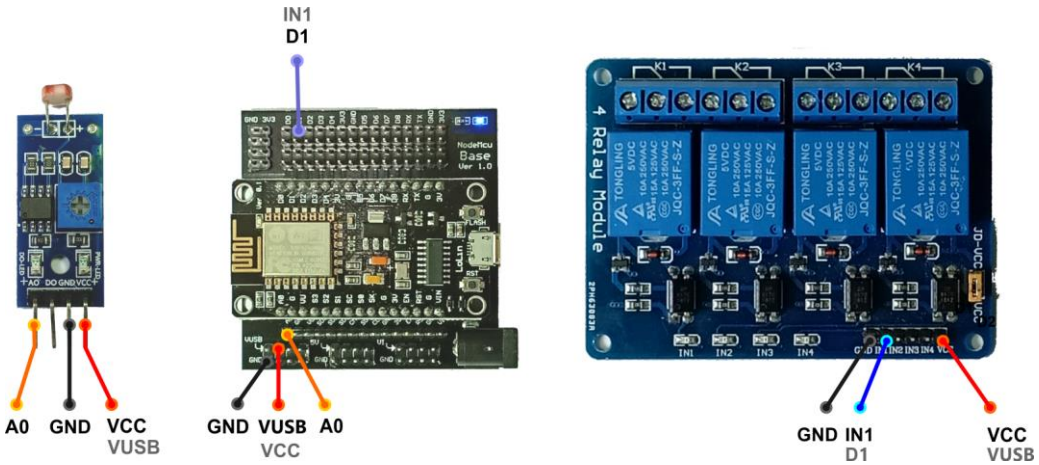
รูปที่ 13.4 แนวคิดการออกแบบระบบควบคุมอัตโนมัติในการทดลองตอนที่ 3



รูปที่ 13.5 แผนผังการทำงานของโปรแกรมที่แสดงในตอนที่ 3

ขั้นตอนการสร้าง/การทดลองตามแผนผังที่ออกแบบไว้ มีดังนี้

1. ต่อวงจรดังรูปที่ 13.6 โดยต่อรีเลย์เข้ากับขา D1 ของ NodeMCU เพียงขาเดียว และจ่ายไฟเลี้ยงให้กับทั้งบอร์ดโมดูลวัดความสว่างและบอร์ดรีเลย์



รูปที่ 13.6 การเชื่อมต่อเพื่อใช้สร้างระบบควบคุมการทำงานโคมไฟด้วยแสง

2. เขียนโค้ดโปรแกรมดังแสดงข้างล่าง (อย่าลืมแก้ไข auth, ssid และ pass)

```
// Control NodeMCU ESP8266 by Blynk
// Read value from LDR and control a relay

// Comment this out to disable prints
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "9c687abf4897467992e46e7249af7209";
char ssid[] = "XXXXXXX";
char pass[] = "XXXXXXX";

#define OFF HIGH
#define ON LOW

```

```
int Relay1 = D1;
int LDR_Sensor = A0;

float Cal_illuminance() {
    float illu_LDR;
    float volt_LDR;
    int data_LDR;
    data_LDR = analogRead(LDR_Sensor);
    volt_LDR = (3.3/1024)*data_LDR;
    illu_LDR = (42.0 * pow(volt_LDR, -3.15));
    return illu_LDR;
}

void setup() {
    Serial.begin(9600);    // for debugging
    Blynk.begin(auth, ssid, pass);
    pinMode(Relay1, OUTPUT);
    digitalWrite(Relay1, OFF);
}

void loop() {
    int state1;  // State of virtual button V1
    float illu_LDR;
    illu_LDR = Cal_illuminance();
    // Serial.println(String(illu_LDR));
    Blynk.virtualWrite(V0, String(illu_LDR));
    if (illu_LDR < 100.0) {
        digitalWrite(Relay1, ON);
        delay(1000);
    } else {
        digitalWrite(Relay1, OFF);
        delay(1000);
    }
    Blynk.run();
    delay(100);
}
```

3. อัปโหลดโปรแกรมลงใน NodeMCU เพื่อให้ฮาร์ดแวร์รอรับคำสั่งเชื่อมต่อ

4. เปิด Blynk App โดยอาจสร้างโปรเจคใหม่ (ขอ Token ใหม่) หรือลบอุปกรณ์ในโปรเจคเดิมก็ได้ จากนั้นจึงสร้างตัวแสดงค่า (Value Display) 1 ตัว โดยกำหนด INPUT PIN เป็นปุ่มเสมือน V0 ดังรูปที่ 13.2 (เหมือนกับการทดลองตอนที่ 1)

5. รันโปรแกรมเพื่อทดสอบการรับค่าเข้าจากโมดูลวัดความสว่างแสง โดยสังเกตว่า เมื่อค่าความสว่างมีค่าน้อยกว่า 100 ลักซ์แล้ว รีเลย์ 1 จะทำงาน (นำกระแส)

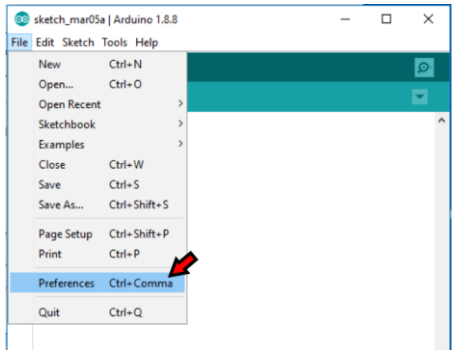
ภาคผนวก ก
การเพิ่มบอร์ดและไลบรารี ในโปรแกรม Arduino IDE

ก.1 การเพิ่มบอร์ด NodeMCU (ชิป ESP8266)

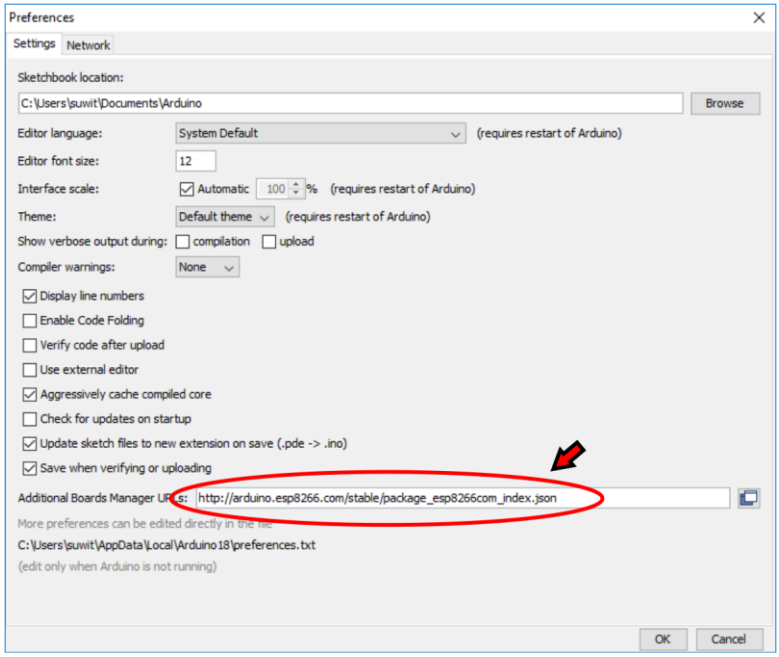
ในบทที่ 3 เราได้กล่าวถึง การติดตั้งโปรแกรม Arduino IDE สำหรับบอร์ด Node-MCU v.3 ที่ใช้ในการทดลองที่แสดงในหนังสือเล่มนี้ แต่สำหรับผู้ที่มีโปรแกรม Arduino IDE อยู่ก่อนแล้ว อาจไม่จำเป็นต้องติดตั้งโปรแกรมใหม่ แต่จะสามารถเพิ่มบอร์ด NodeMCU ให้โปรแกรม Arduino IDE ได้ โดยเชื่อมต่อคอมพิวเตอร์กับอินเทอร์เน็ตแล้วทำตามขั้นตอนต่อไปนี้

เพิ่ม Board Manager URL

ในโปรแกรม Arduino IDE ไปที่เมนู File แล้วเลือก Preferences ดังรูปที่ ก.1 แล้วทำการพิมพ์ URL ของแพ็กเกตสำหรับ ชิป ESP8266 คือ **http://arduino.esp8266.com/stable/package_esp8266com_index.json** ลงในช่อง Additional Boards Manager URLs ดังรูปที่ ก.2

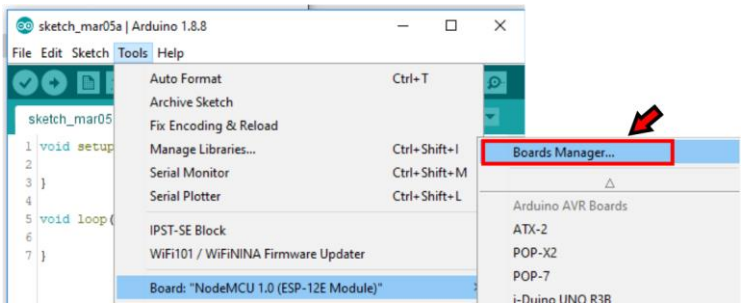


รูปที่ ก.1 เมนูสำหรับเลือก เพิ่ม Board Manager URL ใน Arduino IDE

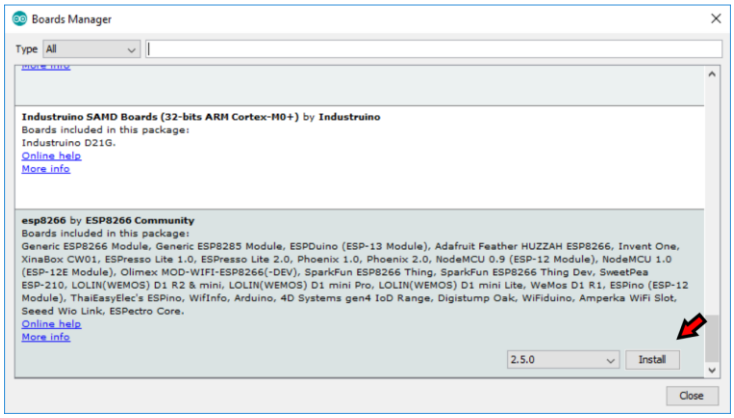


รูปที่ ก.2 บล็อกกำหนด Preference สำหรับเพิ่ม Board Manager URL

ไปที่เมนู Tools แล้วเลือก Board ดังรูปที่ ก.3 แล้วเลือก Boards Manager ... บล็อกข้อความเลือกบอร์ดจะปรากฏขึ้น ดังแสดงในรูปที่ ก.4 จากนั้นจึงเลื่อนลงมาล่างสุด จะพบ แพ็กเกต esp8266 by ESP8266 Community ขอให้เลือกติดตั้ง (Install) ดังรูปที่ ก.4

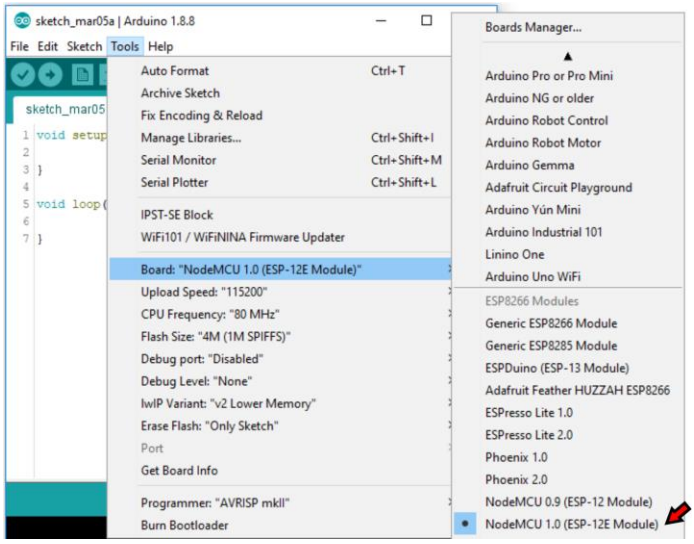


รูปที่ ก.3 เมนู Boards manager



รูปที่ ก.4 บล็อกกำหนด Preference สำหรับเพิ่ม Board Manager URL

โปรแกรม Arduino IDE จะทำการดาวน์โหลดและติดตั้งแพ็กเกตเสริมสำหรับบอร์ด NodeMCU v.3 ที่ใช้ชิป ESP8266 โดยมีระยะเวลาที่ใช้ขึ้นกับความเร็วอินเทอร์เน็ตและความเร็วของคอมพิวเตอร์ เมื่อติดตั้งเสร็จแล้ว จะมีตัวเลือกเลือกบอร์ด NodeMCU 1.0 (ESP-12E) ปรากฏขึ้น ดังแสดงในรูปที่ ก.5



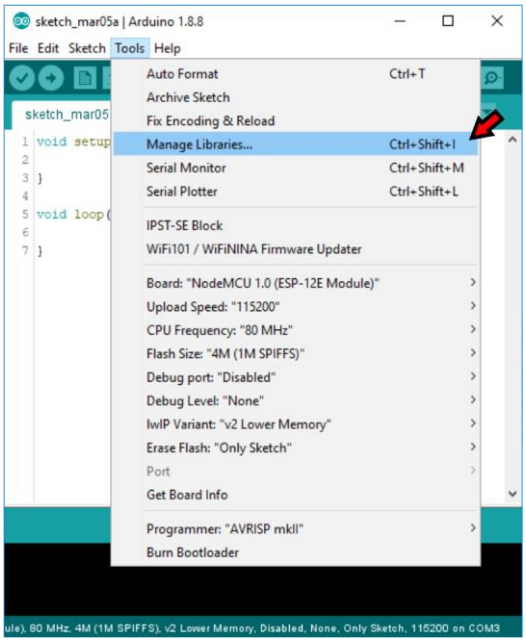
รูปที่ ก.5 การเลือก Board: NodeMCU 1.0 (ESP-12E Module) ในโปรแกรม Arduino IDE

ก.2 การเพิ่มไลบรารีในโปรแกรม Arduino IDE

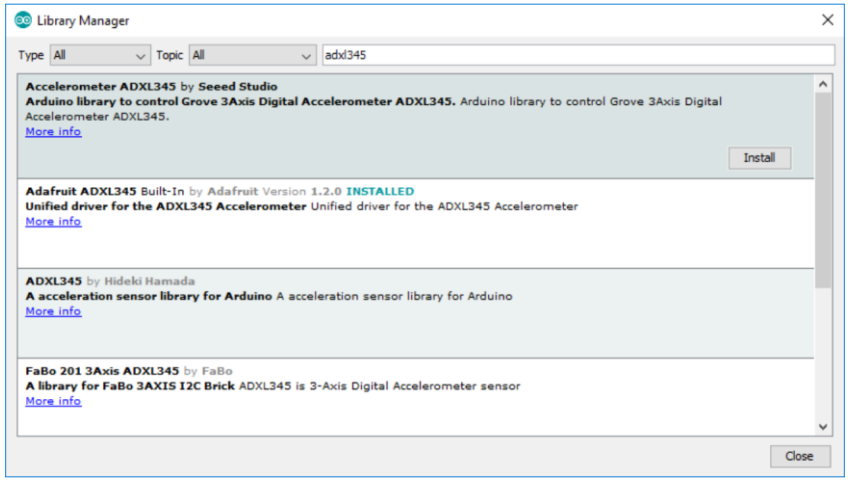
ในบทที่ 6 เราได้กล่าวถึง การเพิ่มไลบรารีลงในโปรแกรม Arduino IDE โดยไลบรารีเหล่านี้จำเป็นสำหรับการเรียกใช้งานฮาร์ดแวร์เฉพาะที่ทางผู้ผลิต/ผู้พัฒนาได้เตรียมไว้ ตัวอย่างเช่น OLED ที่ใช้ในบทที่ 6 อัลตราโซนิกส์โมดูลในบทที่ 7 เซอร์โวมอเตอร์ในบทที่ 8 โมดูลเซนเซอร์วัดความเร่งแบบสามแกน ADXL345 ในบทที่ 10 และ Blynk ในบทที่ 12 สำหรับการติดตั้งไลบรารีนี้ ผู้ใช้สามารถทำได้หลายวิธีซึ่งในภาคผนวกนี้ จะกล่าวถึงอีก 2 วิธี ที่มิได้นำมาใช้ในหนังสือเล่มนี้

การเพิ่มโดยตัวจัดการไลบรารี (library manager)

ในโปรแกรม Arduino IDE ไปที่เมนู Tools แล้วเลือก Manager Libraries (หรือเลือกในเมนู Sketch\Include Libraries\Manage Libraries) ดังรูปที่ ก.6 แล้วทำการค้นหาและเลือกติดตั้งไลบรารีที่ต้องการ ดังรูปที่ ก.7



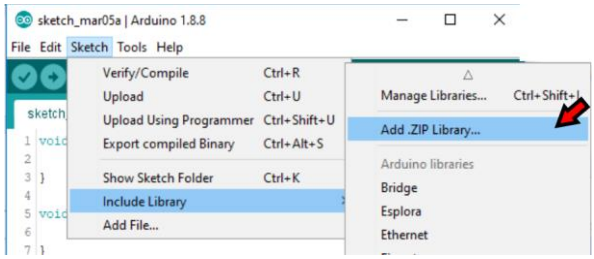
รูปที่ ก.6 การเลือก Manage Libraries ในเมนู Arduino IDE



รูปที่ ก.7 หน้าต่าง Library Manager

การเพิ่มจากไฟล์ ZIP

ในบางครั้ง ไลบรารีจากผู้พัฒนาที่สามารถดาวน์โหลดได้ อยู่ในรูปไฟล์ ZIP ซึ่งรวบรวมข้อมูลทั้งหมดไว้ในไฟล์เดียว โดยเราสามารถนำไลบรารีเข้ามาใน Arduino IDE ได้โดยการเลือกเมนู Sketch และ Include Library แล้วเลือก Add ZIP library ... จากนั้นจึงเลือกไฟล์ไลบรารี ดังแสดงในรูปที่ ก.8



รูปที่ ก.8 แสดงเมนู Add ZIP Library ...

หากการนำไลบรารีเข้ามาเป็นที่เรียบร้อยดี จะปรากฏตัวอย่างเพิ่มเติมจากไลบรารีที่นำเข้ามา โดยเราสามารถเปิดดูเพื่อศึกษาการใช้งานได้จากเมนู File ใน Examples

ภาคผนวก ข
การใช้งานระบบเก็บข้อมูลออนไลน์ด้วย ThingSpeak®

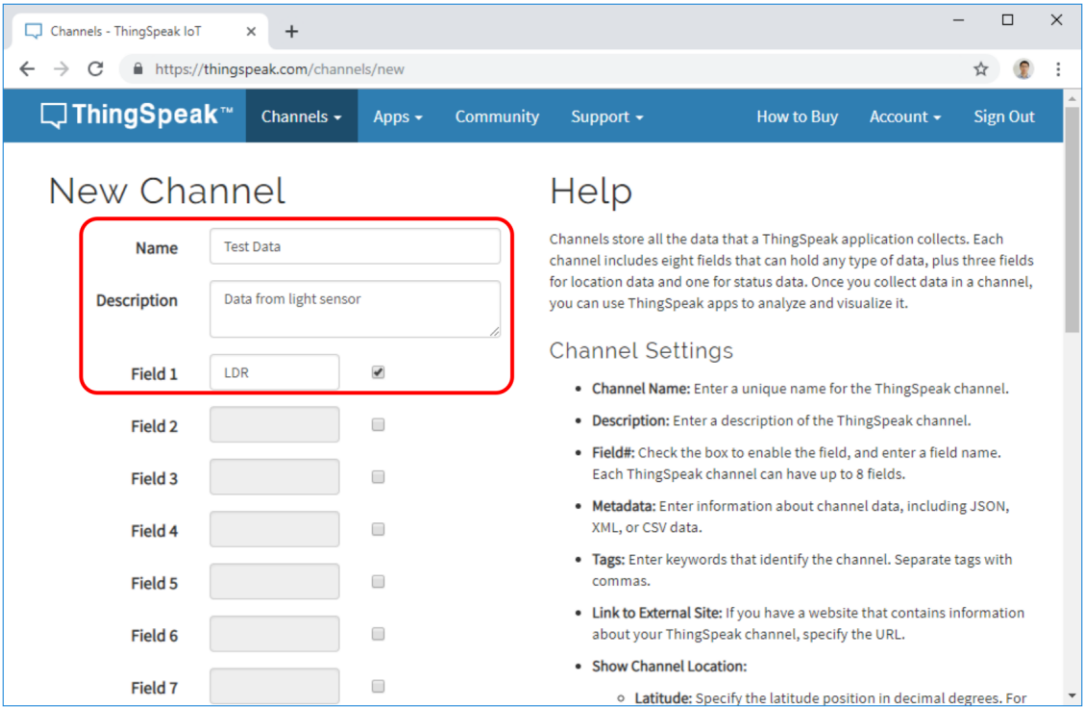
ข.1 ThingSpeak®

ThingSpeak® เป็นเว็บแอปพลิเคชันที่ให้บริการเก็บข้อมูลแบบต่อเนื่องตามเวลา หน้าเว็บ thingspeak.com แสดงดังรูปที่ ข.1 โดยการใช้เว็บแอปพลิเคชันนี้ทำให้เราสามารถเก็บข้อมูลที่อ่านได้จากเซนเซอร์ผ่านไมโครคอนโทรลเลอร์ได้อย่างต่อเนื่อง ซึ่งเหมาะสำหรับการจัดเก็บข้อมูลบางประเภท โดยเราจะต้องอาศัยการสื่อสารข้อมูลผ่านระบบอินเตอร์เน็ต ซึ่งเราหรือบุคคลอื่นที่เราอนุญาตสามารถเรียกดูและนำข้อมูลที่จัดเก็บออกมาได้ผ่านเว็บแอป-พลิเคชันนี้ โดย ThingSpeak มีให้บริการทั้งในรูปแบบมีค่าใช้จ่าย และไม่มีค่าใช้จ่าย ซึ่งสำหรับการบริการแบบไม่มีค่าใช้จ่ายนั้น ผู้ให้บริการได้จำกัดการบันทึกข้อมูลไว้ด้วยอัตราสูงสุด 1 ครั้ง ในช่วงเวลา 15 วินาที

รูปที่ ข.1 แสดงหน้าเว็บ ThingSpeak.com

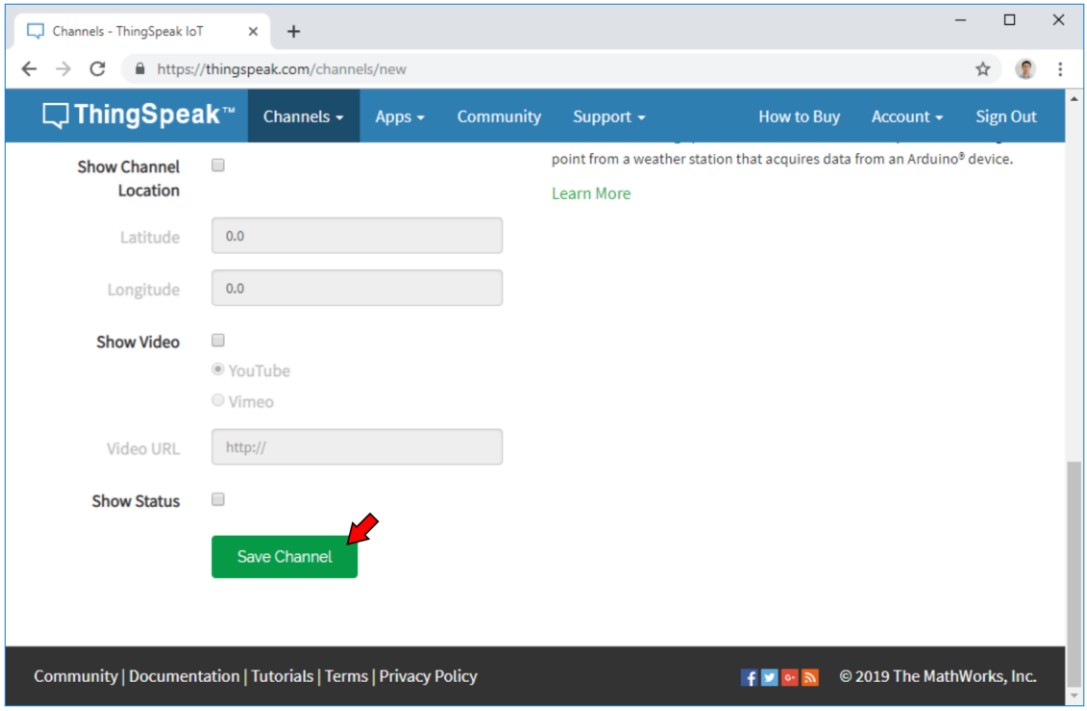
การใช้งาน ThingSpeak เบื้องต้น

ก่อนเริ่มใช้งาน ThingSpeak เราจะต้องสมัคร (Sign Up) เสียก่อน โดยการเข้าที่เว็บ https://thingspeak.com เพื่อทำการสมัครสมาชิก กรอกข้อมูลเพื่อลงทะเบียนเปิดใช้งาน เมื่อทำการสมัครสมาชิกเสร็จสิ้น ทำการ Sign In เพื่อทำการสร้าง Channel โดยให้เรากดไปที่ My Channel แล้วเลือก New Channel จะปรากฏหน้าต่าง ดังแสดงในรูปที่ ข.2



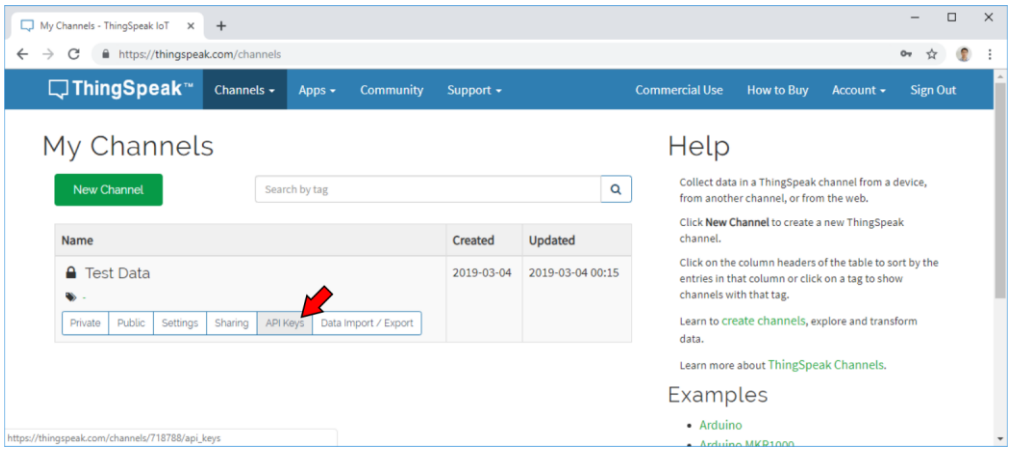
รูปที่ ข.2 หน้าต่างแสดงการกำหนด Channel ในการบันทึกข้อมูล

จากนั้นจึงทำการการกำหนดชื่อของ Channel กรอกคำอธิบาย กรอกชื่อชนิดของข้อมูลที่ต้องการบันทึก ตัวอย่างในรูปที่ ข.2 แสดงการบันทึกชื่อที่กำหนดในการอ่านค่าจากเซนเซอร์ LDR โดยกำหนดชื่อ “LDR” ในฟิลด์ของข้อมูล “Field 1” หลังจากนั้นจึงเลื่อนลงมาเพื่อบันทึก (Save Channel)

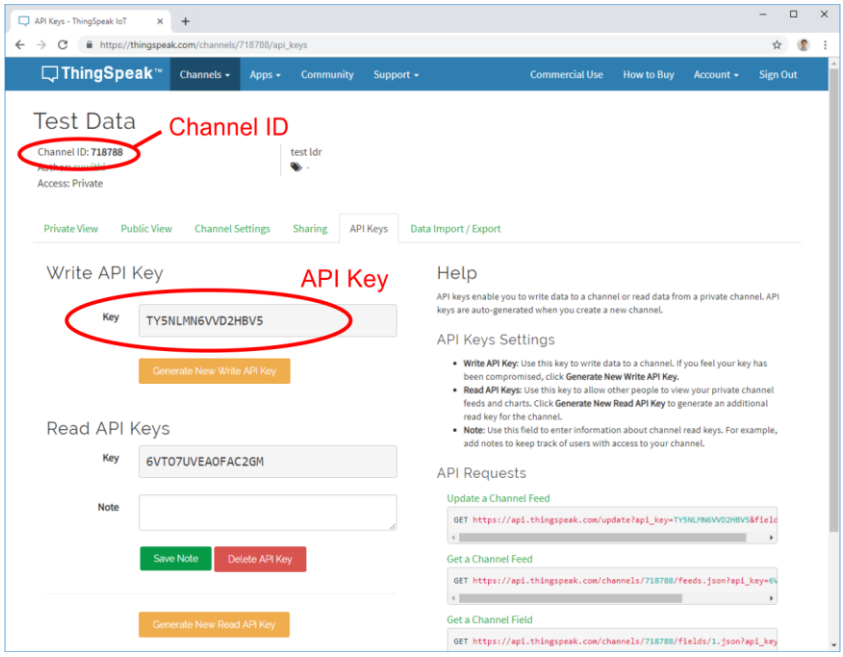


รูปที่ ข.3 หน้าต่างแสดงปุ่มบันทึก Channel ที่สร้าง

จากนั้นเราจะตรวจพบ Channel ที่เราสร้างขึ้นภายใต้เมนู My Channel โดยเมื่อเลือกเมนู จะพบปุ่ม API Key (รูปที่ ข.4) ซึ่งเราจะต้องทำการจดบันทึกค่า ID และ API Key เพื่อนำไปใส่ในโปรแกรมที่เขียนลงในไมโครคอนโทรลเลอร์ เพื่อระบุเส้นทางการส่งข้อมูล เมนูตัวอย่างค่า ID และ API Key แสดงดังรูปที่ ข.5



รูปที่ ข.4 หน้าต่างแสดง Channel ที่สร้างขึ้น



รูปที่ ข.5 หน้าต่างแสดง ID และ API Key

ข.2 การเชื่อมต่อกับ ThingSpeak®

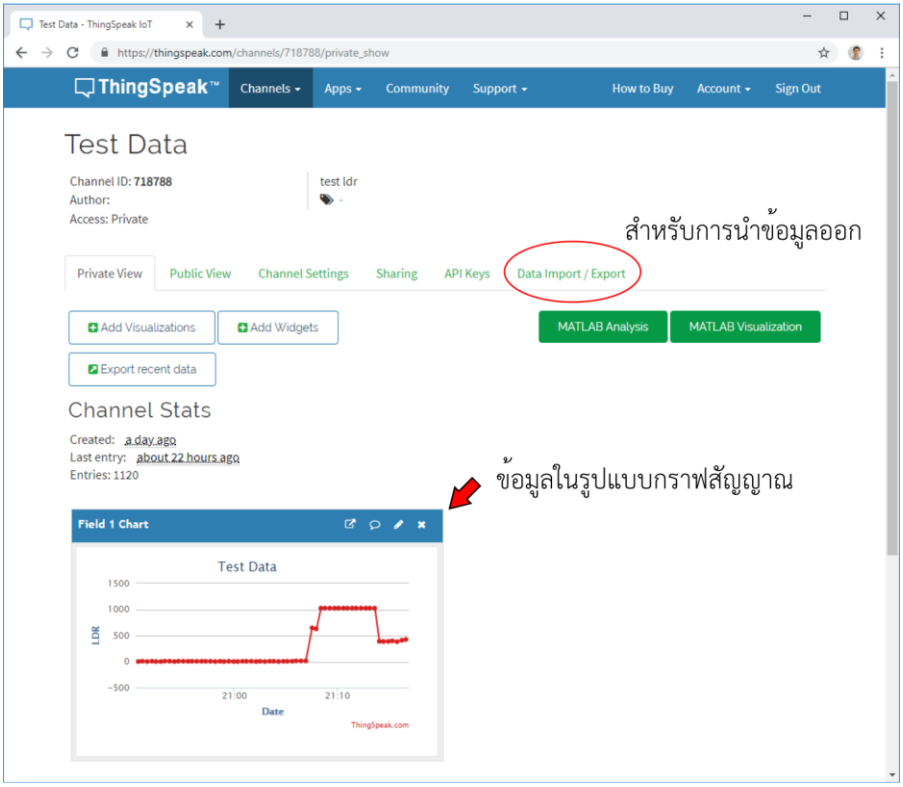
ThingSpeak® ได้จัดเตรียมไลบรารีและตัวอย่างไฟล์สำหรับการเชื่อมต่อไว้ โดยผู้ใช้สามารถดาวน์โหลดได้จาก https://github.com/mathworks/thingspeak-arduino และ https://github.com/nothans/thingspeak-esp-examples เมื่อผู้ใช้ได้ทำการติดตั้งไลบรารีแล้วก็จะสามารถเรียกใช้ไลบรารีใน ThingSpeak ได้โดยใช้คำสั่ง #include “ThingSpeak.h” สำหรับแนวการเขียนโค้ดเบื้องต้นในการส่งข้อมูลจากสัญญาณ A0 ไปเก็บยัง ThingSpeak ผู้ใช้สามารถศึกษาได้จากโค้ดตัวอย่าง (จากไฟล์ A0_to_ThingSpeak.ino ที่อยู่ใน thingspeak-esp-examples-master) โดยในตัวอย่างที่แสดงนี้จะมีคำอธิบายจากผู้พัฒนาอยู่ด้วย ดังแสดงข้างล่างนี้

```
/*
 ESP8266 --> ThingSpeak Channel

 This sketch sends the value of Analog Input (A0)
 to a ThingSpeak channel using the ThingSpeak API
*/

#include "ThingSpeak.h"
#include "secrets.h"

unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;

#include <ESP8266WiFi.h>

char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;

WiFiClient  client;

void setup() {
  Serial.begin(115200);
  delay(100);
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
}
```

```
 27
 28 void loop() {
 29   // Connect or reconnect to WiFi
 30   if (WiFi.status() != WL_CONNECTED) {
 31    Serial.print("Attempting to connect to SSID:");
 32     Serial.println(SECRET_SSID);
 33     while (WiFi.status() != WL_CONNECTED) {
 34       WiFi.begin(ssid, pass);
353       Serial.print(".");
 36       delay(5000);
 37     }
 38     Serial.println("\nConnected.");
 39   }
 40
 41   // Measure Analog Input (A0)
 42   int valueA0 = analogRead(A0);
 43
 44   // Write value to Field 1 of a ThingSpeak Channel
 45   int httpCode =
ThingSpeak.writeField(myChannelNumber, 1, valueA0,
myWriteAPIKey);
 46
 47   if (httpCode == 200) {
 48     Serial.println("Channel write successful.");
 49   }
 50   else {
 51     Serial.println("Problem writing to channel.
 52 HTTP error code " + String(httpCode));
 53   }
 54
 55   // Wait 20 seconds to update the channel again
 56   delay(20000);
 57 }
 58
```

โดยไฟล์ Secrets.h ที่เรียกในตอนแรก จะใช้เก็บ SSID และรหัสผ่านของระบบเครือข่ายที่ NodeMCU เชื่อมต่ออยู่ และเก็บ ID และ API Key ของผู้ใช้ ThingSpeak (ดูรูปที่ ข.5)

เมื่อทำการอัปโหลดโปรแกรมแล้วปล่อยให้ NodeMCU ทำงานไปสักพักหนึ่ง (เนื่องจากข้อมูลจะเก็บทุก ๆ 20 วินาที ตามโค้ดที่แสดงข้างบน) เราก็จะสามารถเรียกดูข้อมูลที่เก็บในเว็ป ThingSpeak ได้ ตัวอย่างข้อมูลแสดงดังรูปที่ ข.6 โดยผู้ใช้สามารถนำข้อมูลที่บันทึกไว้ออกมาหรือนำข้อมูลไปวิเคราะห์ร่วมกับข้อมูลอื่น ๆ ได้ในลำดับต่อไป



รูปที่ ข.6 หน้าต่างแสดงข้อมูลที่บันทึกไว้

บรรณานุกรม

ประเภทหนังสือ

- ผู้ช่วยศาสตราจารย์ดอนสัน ปงผาบ, ภาษาซีและ Arduino, พิมพ์ครั้งที่ 1 สำนักพิมพ์สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2560

- ประภาส พุ่มพวง, การเขียนและการประยุกต์ใช้งานโปรแกรม Arduino, ซีเอ็ดยูเคชั่น, 2561

- ผศ. ดร. เดชฤทธิ์ มณีธรรม, คัมภีร์และการใช้งานไมโครคอนโทรลเลอร์ Arduino, ซีเอ็ดยูเคชั่น, 2560

- ดร. กอบเกียรติ สระอุบล, พัฒนา IoT บนแพลตฟอร์ม Arduino และ Raspberry Pi, สำนักพิมพ์ อินเตอร์มีเดีย, 2561

- สนธยา นงนุช, การใช้งาน ESP32 เบื้องต้น, ร้านไอโอเอ็กซ์ฮือบ, 2560

- ผศ. สุรพนธ์ ตุ้มนาค, เรียนรู้และพัฒนาอุปกรณ์ Internet of Things (IOT) อย่างง่ายกับ Blynk, อินโนเวตีฟ เอ็กเพอริเมนต์ (www.inex.co.th)

ประเภทแหล่งข้อมูลออนไลน์ (เรียงตามลำดับที่ปรากฏในเล่ม)

https://www.ecnmag.com/news/2019/01/top-10-robotic-innovations-2018

https://airdronecraze.com/drone-trends

https://iot-analytics.com

https://github.com/nodemcu/nodemcu-devkit

https://www.arduino.cc/en/main/software

https://inex.co.th/shop/software-download

https://github.com/ThingPulse/esp8266-oled-ssd1306

https://github.com/hemalchevli/ultrasonic-library

https://github.com/adafruit/Adafruit_Sensor

https://github.com/adafruit/Adafruit_ADXL345

https://learn.adafruit.com/adxl345-digital-accelerometer/programming

https://github.com/blynkkk/blynk-library

https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/

https://www.arduino.cc/en/guide/libraries

https://thingspeak.com

https://github.com/mathworks/thingspeak-arduino

https://github.com/nothans/thingspeak-esp-examples

รายชื่อคณะทำงาน

1. ฝ่ายวิชาการและการฝึกอบรม ประกอบด้วย

1. รศ.ดร. สุวิทย์	กิระวิทยา	คณะวิศวกรรมศาสตร์ ม.นเรศวร
2. รศ.ดร. พนัส	นัถฤทธิ์	คณะวิศวกรรมศาสตร์ ม.นเรศวร
3. อ. ธานี	โกสุม	คณะวิศวกรรมศาสตร์ ม.นเรศวร
4. ผศ.ดร. เกรียงศักดิ์	เตมีย์	คณะวิทยาศาสตร์ ม.นเรศวร
5. ดร. ประชา	คำภักดี	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
6. ดร. เกรียงศักดิ์	ตรีประพิณ	คณะวิทยาศาสตร์ ม.อุบลราชธานี
7. ผศ.ดร. อธิพงศ์	สุริยา	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
8. ดร. ธีรวุฒิ	ไชยธรรม	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
9. ดร. ธรรมรส	รักธรรม	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
10. อ. ผดุง	กิจแสวง	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
11. อ. สมนึก	เวียนวัฒนชัย	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
12. นายวุฒิไกร	สร้อยสิงห์	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
13. นายปวรุตม์	กองสมบัติสุข	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี

2. ฝ่ายประกวดแข่งขันและดำเนินงานโครงการ ประกอบด้วย

1. ผศ.ดร. วรการ	วงศ์สายเชื้อ	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
2. ผศ.ดร. ประสิทธิ์	นครราช	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
3. ผศ.ดร. ถนัดกิจ	ชารีรัตน์	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
4. ผศ.ดร. นันทวัฒน์	วีระยุทธ	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี
5. อ. บงกช	จันทมาส	คณะวิศวกรรมศาสตร์ ม.อุบลราชธานี

5. นส. นุชนารถ	แก้วแดง	ผู้ประสานงานศูนย์อบรมที่ 2 ม.นเรศวร
6. นายเฉลิมชัย	ไชยกาล	ผู้ประสานงานศูนย์อบรมที่ 1 ม.อุบลราชธานี
7. นางคอยจิตร์	ศาลางาม	ผู้ช่วยประสานงานศูนย์อบรมที่ 1 ม.อุบลราชธานี

3. ฝ่ายสนับสนุนงบประมาณและสถานที่ ประกอบด้วย

1. มร. ฮิเดยูกิ	ทาเคดะ	กรรมการผู้จัดการ บริษัท ไทยบริดจสโตน จำกัด
2. มร. ฮิโรยูกิ	ไซโตะ	ผู้อำนวยการสายงานการตลาดและกลยุทธ์
3. น.ส. ภัทรวัลลิ์	ขันทอง	ผู้จัดการแผนกสื่อสารองค์กรและกิจกรรมเพื่อสังคม บริษัท ไทยบริดจสโตน จำกัด
4. น.ส. สุกุลยา	ปั้นจาด	ผู้ช่วยผู้จัดการแผนกสื่อสารองค์กรและกิจกรรมเพื่อสังคม บริษัท ไทยบริดจสโตน จำกัด


BRIDGESTONE