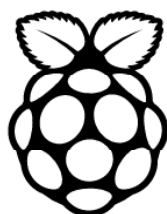


Raspberry Pi 3

บอร์ดคอมพิวเตอร์ขนาดเล็ก

ติดต่ออุปกรณ์ภายนอกผ่าน พอร์ตอินพุตเอาต์พุต GPIO



ผศ.โอภาส ศิริครรชิตถาวร

กฤษฎดา ใจเย็น

วรพจน์ กรแก้ววัฒนกุล

นวพร เหล่าวัฒนธรรม

ธีรวิธ จิตพรมมา



Raspberry Pi 3

บอร์ดคอมพิวเตอร์ขนาดเล็ก : ติดต่ออุปกรณ์ภายนอกผ่านพอร์ตอินพุตเอาต์พุต GPIO (เล่มที่ 1)

ผศ.โอภาส ศิริครรชิตถาวร, กฤษดา ใจเย็น, วรพจน์ กรแก้ววัฒนกุล,
นวพร เหล่าวัฒนธรรม, อีรุธ จิตพรมมา

สงวนลิขสิทธิ์ตาม พ.ร.บ. ลิขสิทธิ์ พ.ศ. 2537

ห้ามการลอกเลียนไม่ว่าส่วนหนึ่งส่วนใดของหนังสือเล่มนี้ นอกจากจะได้รับอนุญาต

ใครควรใช้หนังสือเล่มนี้

1. นักเรียน นิสิต นักศึกษา และบุคคลทั่วไปที่มีความสนใจใช้งานบอร์ดคอมพิวเตอร์ 32 บิตขนาดเล็กสมัยใหม่ที่ติดตั้งระบบปฏิบัติการที่เหมาะสมสำหรับการใช้งานเป็น Embedded Computer หรือ Embedded PC ที่มีประสิทธิภาพสูง
2. สถาบันการศึกษา โรงเรียน วิทยาลัย มหาวิทยาลัย ที่มีการเปิดการเรียนการสอนวิชาอิเล็กทรอนิกส์หรือภาควิชาวิศวกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์
3. คณาจารย์ที่มีความต้องการศึกษา และเตรียมการเรียนการสอนเกี่ยวกับระบบสมองกลฝังตัวทั้งในระดับเบื้องต้นจนถึงขั้นก้าวหน้าในระดับอาชีพศึกษา และปริญญาตรี

ดำเนินการจัดพิมพ์และจำหน่ายโดย

บริษัท อินโนเวทีฟ เอ็กเพอริเมนต์ จำกัด

108 ซ.สุขุมวิท 101/2 ถ.สุขุมวิท แขวงบางนา เขตบางนา กรุงเทพฯ 10260

โทรศัพท์ 0-2747-7001-4

โทรสาร 0-2747-7005

รายละเอียดที่ปรากฏในหนังสือเล่มนี้ได้ผ่านการตรวจทานอย่างละเอียดและถี่ถ้วน เพื่อให้มีความสมบูรณ์และถูกต้องมากที่สุดภายใต้เงื่อนไขและเวลาที่พึงมีก่อนการจัดพิมพ์เผยแพร่ ความเสียหายอันอาจเกิดจากการนำข้อมูลในหนังสือเล่มนี้ไปใช้ ทางบริษัท อินโนเวทีฟ เอ็กเพอริเมนต์ จำกัด มิได้มีภาระในการรับผิดชอบแต่ประการใด ความผิดพลาดคลาดเคลื่อนที่อาจมีและได้รับการจัดพิมพ์เผยแพร่ออกไปนั้น ทางบริษัทฯ จะพยายามชี้แจงและแก้ไขในการจัดพิมพ์ครั้งต่อไป

คำนำ

Raspberry Pi คือบอร์ดไมโครคอมพิวเตอร์แบบแผ่นเดียวที่โด่งดังที่สุดเท่าที่เคยมีมา มันคือบอร์ดคอมพิวเตอร์ที่ขี้อร่อยไม่ที่บรรจุความสามารถไว้อย่างมากมาย มีราคาถูก อุดมด้วยความสามารถขั้นเทพ ตัวบอร์ดมาพร้อมกับจุดเชื่อมต่ออุปกรณ์อินพุตเอาต์พุตหลากหลาย ทั้งผ่านพอร์ต USB, LAN, HDMI, ช่องสัญญาณภาพ และ GPIO สำหรับต่อกับวงจรหรืออุปกรณ์อิเล็กทรอนิกส์

Raspberry Pi เป็นบอร์ดคอมพิวเตอร์ที่ได้รับความนิยมสูงเป็นประวัติการณ์ของวงการระบบสมองกลฝังตัวโลก ด้วยยอดจำหน่ายมากกว่า 1 ล้านบอร์ดในเวลาไม่ถึงหนึ่งปีหลังจากที่ออกจำหน่ายครั้งแรกในวันที่ 29 กุมภาพันธ์ ค.ศ. 2012 และนับถึงวันที่จัดทำหนังสือเล่มนี้มียอดจำหน่ายไปมากกว่า 10 ล้านชิ้น (นับถึงเดือนกันยายน ค.ศ. 2016)

ในหนังสือเล่มนี้มีเนื้อหาต่อเนื่องจาก Raspberry Pi 3 : รู้จักและใช้งานเบื้องต้น โดยเน้นไปที่การใช้งานพอร์ตอินพุตเอาต์พุต GPIO ของ Raspberry Pi 3 ในการติดต่อกับอุปกรณ์ภายนอกทั้งแบบติดต่อโดยตรงและผ่านระบบบัสสื่อสารข้อมูล 2 สาย โดยมีอุปกรณ์ช่วยคือ บอร์ด LeafHAT อันเป็นบอร์ดอุปกรณ์อินพุตเอาต์พุตพร้อมใช้งานที่ติดตั้งเข้ากับคอนเน็กเตอร์ GPIO 40 ขาของ Raspberry Pi 3 โดยตรง มีขนาดตรงตามข้อกำหนด Hardware Attach on Top ของ Raspberry Pi Foundation

อนุกรมหนังสือเพื่อการเรียนรู้และใช้งาน Raspberry Pi ซึ่งจัดทำโดยบริษัท อินโนเวทีฟ เอ็กเพอริเมนต์ จำกัดหรือ INEX มีด้วยกัน 4 เล่ม คือ

Raspberry Pi 3 : รู้จักและใช้งานเบื้องต้น

Raspberry Pi 3 : ติดต่ออุปกรณ์ภายนอกผ่านพอร์ตอินพุตเอาต์พุต GPIO เล่มที่ 1 (เล่มนี้)

Raspberry Pi 3 : ติดต่ออุปกรณ์ภายนอกผ่านพอร์ตอินพุตเอาต์พุต GPIO เล่มที่ 2

Raspberry Pi 3 : ควบคุมอุปกรณ์ผ่านเครือข่ายอินเทอร์เน็ตเบื้องต้น

ทางผู้จัดทำหวังเป็นอย่างยิ่งว่า หนังสือในชุดนี้จะมีส่วนช่วยให้การใช้งานบอร์ดคอมพิวเตอร์ยอดนิยมนี้นี้เป็นไปอย่างมีประสิทธิภาพ ช่วยให้ผู้เริ่มต้นรู้จักและนำไปใช้งานได้อย่างที่ควรจะเป็น รวมถึงสามารถต่อยอดให้เกิดเป็นโครงการหรือสร้างงานที่ใช้ได้จริง ซึ่งจะส่งผลดีต่อการพัฒนาอุตสาหกรรมระบบสมองกลฝังตัวของประเทศในท้ายที่สุด

ชัยวัฒน์ ลิ้มพรจิตรวิไล

บรรณาธิการ

คำชี้แจงจากคณะผู้เขียน/เรียบเรียง

การนำเสนอข้อมูลเกี่ยวกับข้อมูลทางเทคนิคและเทคโนโลยีในหนังสือเล่มนี้เกิดจากความต้องการที่จะอธิบายกระบวนการและหลักการทำงาน ของอุปกรณ์ในภาพรวมด้วยถ้อยคำที่ง่ายเพื่อสร้างความเข้าใจแก่ผู้อ่าน ดังนั้นการแปลคำศัพท์ทางเทคนิคหลายๆ คำอาจไม่ตรงตามข้อบัญญัติของราชบัณฑิตยสถาน และมีหลายๆ คำที่ยังไม่มีการบัญญัติอย่างเป็นทางการ คณะผู้เขียนจึงขออนุญาตบัญญัติศัพท์ขึ้นมาใช้ในการอธิบาย โดยมีข้อจำกัดเพื่ออ้างอิงในหนังสือเล่มนี้เท่านั้น

ทั้งนี้สาเหตุหลักของข้อชี้แจงนี้มาจาก การรวบรวมข้อมูลของอุปกรณ์ในระบบสมองกลฝังตัวและเทคโนโลยีหุ่นยนต์สำหรับการศึกษาเพื่อนำมาเรียบเรียงเป็นภาษาไทยนั้นทำได้ไม่ย่งนัก ทางคณะผู้เขียนต้องทำการรวบรวมและทดลองเพื่อให้แน่ใจว่า ความเข้าใจในกระบวนการทำงานต่างๆ นั้นมีความคลาดเคลื่อนน้อยที่สุด

เมื่อต้องการเรียบเรียงออกมาเป็นภาษาไทย ศัพท์ทางเทคนิคหลายคำมีความหมายที่ทับซ้อนกันมาก การบัญญัติศัพท์จึงเกิดจากการปฏิบัติจริงร่วมกับความหมายทางภาษาศาสตร์ ดังนั้นหากมีความคลาดเคลื่อนหรือผิดพลาดเกิดขึ้น ทางคณะผู้เขียนขออภัยและหากได้รับคำอธิบายหรือชี้แนะจากท่าน ผู้รู้จะได้ทำการชี้แจงและปรับปรุงข้อผิดพลาดที่อาจมีเหล่านั้นโดยเร็วที่สุด

ทั้งนี้เพื่อให้การพัฒนาสื่อทางวิชาการ โดยเฉพาะอย่างยิ่งกับความรู้ของเทคโนโลยีสมัยใหม่สามารถดำเนินไปได้อย่างต่อเนื่อง ภายใต้การมีส่วนร่วมของผู้รู้ในทุกภาคส่วน

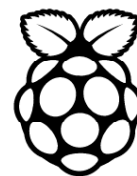
บริษัท อินโนเวทีฟ เอ็กเพอริเมนต์ จำกัด

สารบัญ

บทที่ 1	แนะนำ Raspberry Pi 3.....	7
บทที่ 2	พอร์ตอินพุตเอาต์พุตเนกประสงค์ หรือ GPIO ของ Raspberry Pi 3.....	19
บทที่ 3	Raspberry Pi 3 กับการแสดงผลด้วยจอแสดงผล OLED ผ่านบัส I ² C.....	47
บทที่ 4	Raspberry Pi 3 กับการขับโหลดกระแสไฟฟ้าสูง.....	61
บทที่ 5	Raspberry Pi 3 กับการติดต่อตัวตรวจจับสภาพแวดล้อม	87
	● ใช้งานโมดูลตรวจจับความเคลื่อนไหว ZX-PIR.....	87
	● ใช้งาน ZX-DHT11 บอร์ดวัดความชื้นสัมพัทธ์และอุณหภูมิ.....	93
	● BH1750 มินิบอร์ดตรวจจับและวัดความเข้มแสง.....	101
ภาคผนวก 1	Raspberry Pi 3 กับการพัฒนาโปรแกรมภาษา Python ด้วยซอฟต์แวร์ Geany.....	113

บทที่ 1

แนะนำ Raspberry Pi 3



Raspberry Pi คือบอร์ดคอมพิวเตอร์ขนาดเล็กที่มีราคาถูก ทว่ามีความสามารถเทียบเท่ากับคอมพิวเตอร์ขนาดย่อมๆ รองรับการใช้งานได้เหมือนคอมพิวเตอร์เครื่องหนึ่ง ต่อจอภาพและอุปกรณ์ USB เพื่อใช้งานได้ ไม่ว่าจะเป็นเมาส์ คีย์บอร์ด หรือ USB WiFi ทั้งยังมีจุดเด่นที่ต่างจากคอมพิวเตอร์ทั่วไปคือ มีพอร์ตอินพุตเอาต์พุตหรือ GPIO ให้ใช้งาน จึงทำให้น่า Raspberry Pi นำไปต่อกับอุปกรณ์ต่างๆ เพื่อประยุกต์การทำงานที่เกี่ยวกับอิเล็กทรอนิกส์ได้

จุดประสงค์ของบอร์ด Raspberry Pi คือ ใช้ในด้านการศึกษาและเรียนรู้สำหรับเยาวชน แต่ด้วยความสามารถ ขนาดและราคา ทำให้ Raspberry Pi กลายเป็นบอร์ดยอดนิยมสำหรับนักเรียน นักทดลองระบบสมองกลฝังตัวในวงกว้างเป็นอย่างมาก เพราะประยุกต์ใช้งานได้หลากหลาย นอกเหนือจากด้านการศึกษา ไม่ว่าจะเป็นระบบอัตโนมัติอย่าง Home Automation หรือระบบเซิร์ฟเวอร์หรือหน่วยบริการข้อมูลขนาดย่อมก็ทำได้ ในปัจจุบันมีผู้ผลิตสินค้าหลายรายนำ Raspberry Pi ไปใช้งานจริงในเชิงพาณิชย์



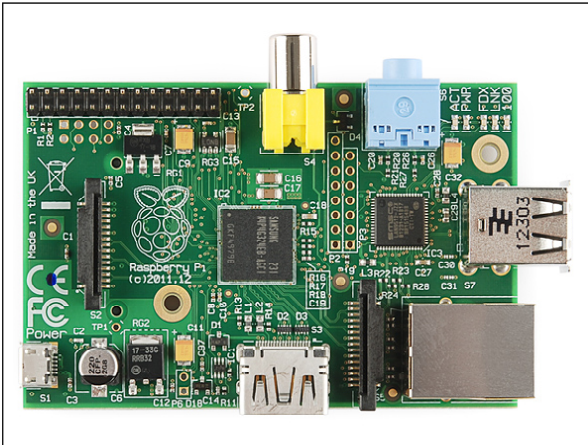
รูปที่ 1-1 Eben Upton ผู้ให้กำเนิดบอร์ด Raspberry Pi (ภาพจาก www.techcentral.co.za)

1.1 จุดกำเนิดของ Raspberry Pi 3

Raspberry Pi ถูกสร้างขึ้นภายใต้แนวคิดที่ไม่ซับซ้อนว่า “ต้องการให้เด็กๆ ฝึกเขียนโปรแกรมด้วยเครื่องมือที่ราคาถูก” แนวคิดนี้มาจากมูลนิธิ Raspberry Pi (www.raspberrypi.org) ที่ต้องการหาเครื่องมือที่ใช้ในการสอนเขียนโปรแกรมให้เด็กๆ ในแบบฟรีหรือถูกที่สุดเท่าที่จะเป็นไปได้ Raspberry Pi จึงถูกสร้างขึ้นโดย **Eben Upton** และคณะทำงานในมูลนิธิ Raspberry Pi แจกจ่ายแก่สาขาวิทยาการคอมพิวเตอร์ของมหาวิทยาลัยเคมบริดจ์ สหราชอาณาจักร เพื่อเรียนรู้และระดมสมอง โดยมีเป้าหมายไปสู่การพัฒนาสื่อเพื่อช่วยให้เด็กๆ สนใจในการเขียนโปรแกรม

Upton ได้ทำการนำเสนอ Raspberry Pi ต่อผู้สนใจ โดยคิดตั้งระบบปฏิบัติการ Linux สาธิตการเล่นไฟล์วิดีโอความละเอียดในระดับ HD ด้วยราคาเพียง US\$35 ทำเอาโลกทั้งใบแทบจะหยุดหมุน สปอตไลต์ของวงการ Embedded System และคอมพิวเตอร์ขนาดเล็กหันไปส่องและโฟกัสไปที่ Raspberry Pi แทบจะในทันที

8 ● Raspberry Pi 3 : ติดต่ออุปกรณ์ภายนอกผ่านพอร์ตอินพุตเอาต์พุต GPIO

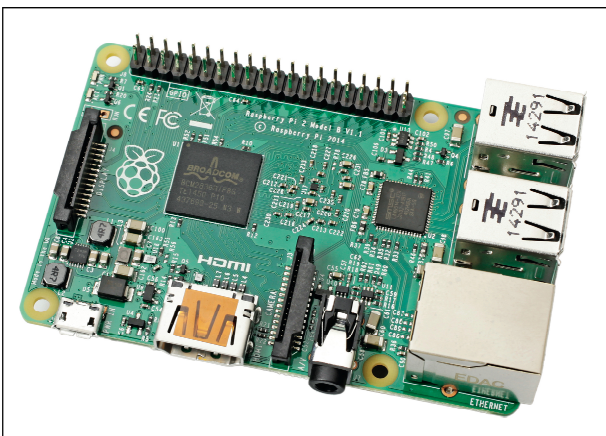


รูปที่ 1-2 บอร์ด Raspberry Pi Model B ที่ออกแบบมา
จำหน่ายเป็นรุ่นแรก

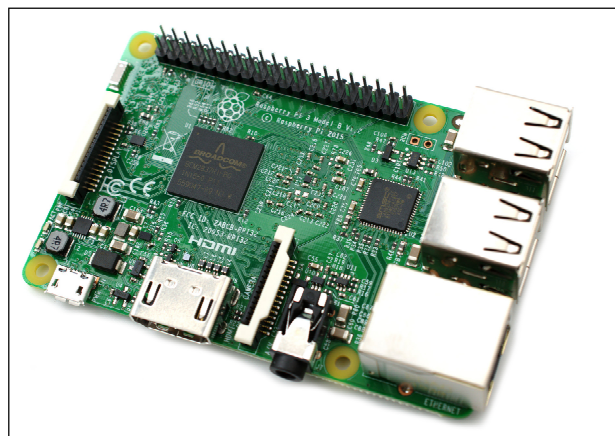
รูปที่ 1-3 บอร์ด Raspberry Pi Model B+ ที่มีการปรับ
อย่างมากด้านรูปลักษณ์และจุดต่ออุปกรณ์ภายนอก

ต่อมาโครงการ Raspberry Pi นี้ได้รับการขยายผลโดย BBC (British Broadcasting Corporation - www.bbc.com) อันเป็นเครือข่ายวิทยุและโทรทัศน์ที่ทรงอิทธิพลของสหราชอาณาจักรและของโลก ทั้งนี้ที่ทั่วโลกได้รับข่าวสารของ Raspberry Pi ยอดการสั่งซื้อก็หลั่งไหลมาอย่างมากมาย

วันที่ 29 กุมภาพันธ์ พ.ศ. 2559 คือวันครบรอบปีที่ 4 ของ Raspberry Pi ทาง Raspberry Pi Foundation ได้นำ Raspberry Pi 3 ออกสู่ตลาดในวันครบรอบวันคล้ายวันเกิด Raspberry Pi 3 มาพร้อมกับชิพชุดใหม่ 64 บิต และติดตั้งชิปวงจรสื่อสารไร้สายสมัยใหม่ที่รวม WiFi และ BLE (Bluetooth Low Energy) ไว้ในตัวถึงทีเดียวกัน เพื่อให้ Raspberry Pi 3 มีขีดความสามารถสูงขึ้น ใช้งานพอร์ต USB ในการต่อกับอุปกรณ์ได้มากขึ้น



รูปที่ 1-4 บอร์ด Raspberry Pi 2 ที่มีการปรับหน้าตา
เล็กน้อยเนื่องจากการเปลี่ยนชิพและเพิ่มความจุ
ของหน่วยความจำเป็น 1GB



รูปที่ 1-5 หน้าตาของบอร์ด Raspberry Pi 3 จะเห็นว่า
ในภาพรวมคล้ายกับบอร์ด Raspberry Pi 2 ค่อนข้าง
มาก (ภาพจาก <https://cdn.infoq.com>)

Raspberry Pi 3 ใช้ชิปประมวลผลหลักเป็น Broadcom BCM2837 ARM Cortex-A53 แบบ 4 แกน ความเร็ว 1.2GHz จึงเร็วกว่า Raspberry Pi รุ่นแรก 10 เท่า มีหน่วยความจำแรม 1GB และพอร์ตเชื่อมต่อต่างๆ อยู่ครบ ความโดดเด่นแตกต่างจึงไปอยู่ที่การบรรจุวงจร WiFi และ Bluetooth 4 (Bluetooth Low Energy - BLE) ไว้บนบอร์ดเลย ทำให้การเชื่อมต่อเครือข่ายแบบไร้สายง่ายและสะดวกขึ้นมาก แจ็ก AV รวมเอาต์พุตเสียงและสัญญาณภาพหรือวิดีโอไว้ด้วยกันเป็นแจ็กหูฟัง 3.5 มม.

1.2 คุณสมบัติทางเทคนิคที่สำคัญของ Raspberry Pi 3

- ชิปประมวลผลหลัก : Broadcom BCM2837 ARM Cortex-A53 แบบ 4 แกน ความเร็ว 1.2GHz
- หน่วยประมวลกราฟิกหรือ GPU : Broadcom VideoCore IV dual-core GPU หรือเทียบเท่ารองรับการแสดงผลผ่านจอ HDMI
- หน่วยความจำ SDRAM : 1GB LPDDR2
- ส่วนสื่อสารข้อมูล : พอร์ตอีเธอร์เน็ต 10/100, WiFi 2.4GHz 802.11n และ Bluetooth 4 (Bluetooth Low Energy - BLE)

● จุดต่อ :

USB 2.0 จำนวน 4 พอร์ต รองรับการขยายผ่าน USB ฮับ ใช้เชื่อมต่อกับอุปกรณ์ USB ได้หลายแบบ อาทิ เมาส์, คีย์บอร์ด, WiFi dongle, บลูทูธ dongle, ฮาร์ดดิสก์ภายนอก, USB แฟลชไดรฟ์ เป็นต้น

แจ็ก AV ที่รวมเอาต์พุตเสียงและสัญญาณภาพหรือวิดีโอไว้ด้วยกันเป็นแจ็กหูฟัง 3.5 มม.

คอนเน็กเตอร์หรือจุดต่อพอร์ตอินพุตเอาต์พุต 40 ขา (General Purpose Input/Output : GPIO) ที่มีขาต่อบัส SPI (Serial Peripheral Interface Bus), I²C, I²S, ขาสัญญาณรับส่งข้อมูลอนุกรมหรือ UART

ซ็อกเก็ตของ micro SD การ์ดสำหรับเสียบ micro SD การ์ดที่ติดตั้งระบบปฏิบัติการแล้ว

จุดต่อ CSI สำหรับต่อโมดูลกล้อง Raspberry Pi

จุดต่อ DSI สำหรับจอแสดงผล LCD

จุดต่อไฟเลี้ยง +5V ผ่านคอนเน็กเตอร์ microUSB

- ความต้องการไฟเลี้ยง : +5V 2.5A เป็นอย่างน้อย แนะนำใช้งานกับอะแดปเตอร์ไฟตรง +5V 2.5A ขึ้นไป หรือเครื่องจ่ายไฟสำรอง Power Bank +5V ที่มีความสามารถในการจ่ายกระแสไฟฟ้ามากกว่า 2A ขึ้นไป

- ขนาด : 85 x 56 มม.

วงจร WIFI และ BLE

เป็นชิป BCMBCM43438 บรรจุวงจรสื่อสารข้อมูล ไร้สาย WIFI และบลูทูธ 4.0 หรือ BLE (Bluetooth Low Energy) ติดตั้งอยู่ด้านหลังของแผ่นวงจรพิมพ์

จุดต่อพอร์ตอินเอาต์พุตดิจิทัลออกเนกประสงค์ 40 ขา

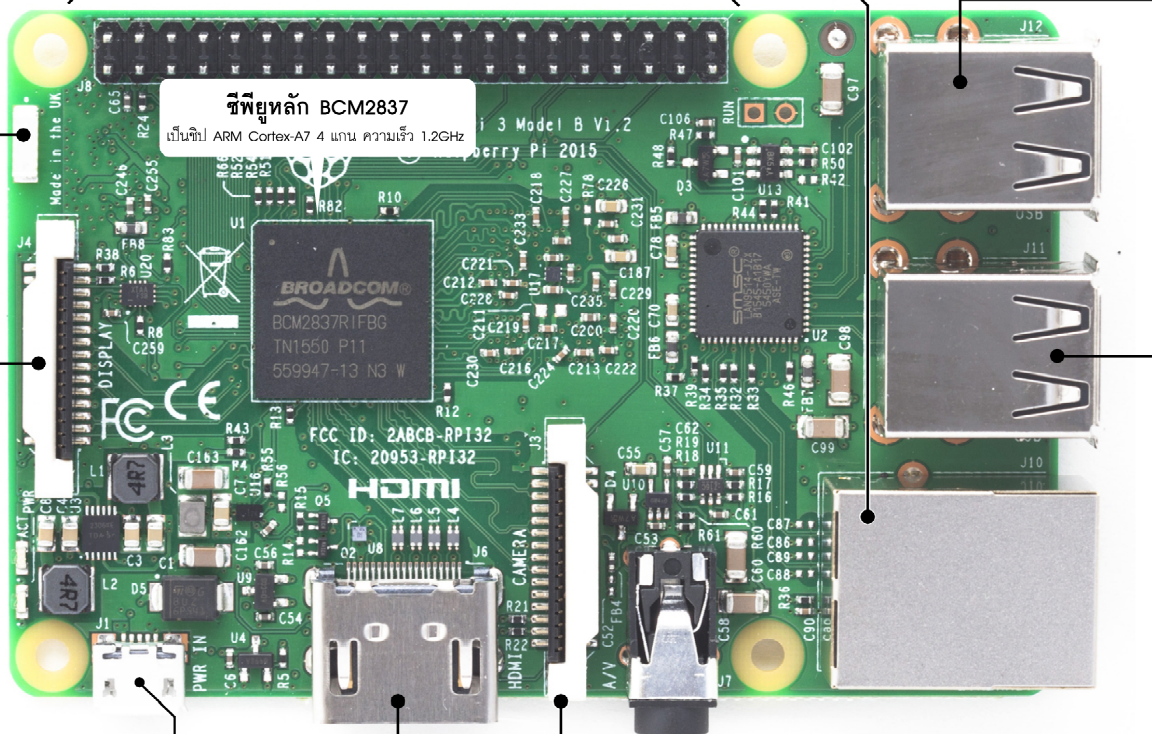
(GPIO - General Purpose Input Output) มีพอร์ตทั้งสิ้น 28 ขา
ใช้งานเป็นพอร์ตอินเอาต์พุตได้ 26 ขา
อีก 2 ขาใช้ต่อกับหน่วยความจำอีพียูรอมภายนอก

จุดต่ออีเธอร์เน็ต

ความเร็ว 10/100 เมกะบิตต่อวินาที สำหรับต่อกับระบบเครือข่ายหรือ LAN เพื่อเชื่อมต่ออินเทอร์เน็ต หรือสื่อสารข้อมูลระหว่างบอร์ดต่อบอร์ด

จุดต่อ USB2.0

มี 4 ช่อง สำหรับต่อกับอุปกรณ์ USB



จุดต่อจอภาพ DSI

ใช้เชื่อมต่อจอแสดงผล LCD ที่ใช้จุดต่อแบบ DSI (Display Serial Interface) ภายใต้มาตรฐาน MIPI (Mobile Industry Processor Interface)

จุดต่อ HDMI

- ใช้ต่อกับจอแสดงผลแบบ HDMI (High-Definition Multimedia Interface) ที่มีความคมชัดสูง มาพร้อมกับสัญญาณเสียงคุณภาพสูงระดับ Dolby Digital และ DTS
- รองรับตัวแปลงสัญญาณ HDMI เป็น DVI และ VGA เพื่อใช้งานกับจอมอนิเตอร์แบบ DVI และ VGA

จุดต่อไฟเลี้ยง +5V

ใช้คอนเนกเตอร์แบบ microUSB

ซ็อกเก็ต micro SD การ์ด

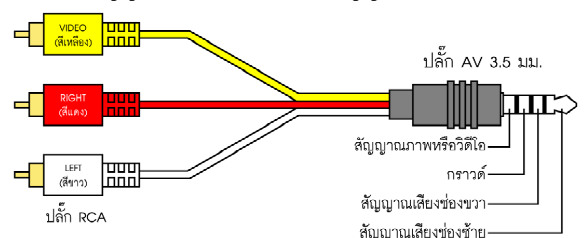
สำหรับติดตั้งแผ่น micro SD การ์ด ที่บรรจุระบบปฏิบัติการเพื่อควบคุมการทำงาน (ติดตั้งด้านหลังของแผ่นวงจรพิมพ์)

หน่วยความจำแรม 1GB

ทำงานร่วมกับชิป BCM2837 ช่วยให้ทำงานเร็วกว่า Raspberry Pi รุ่นแรก 10 เท่า

จุดต่อสัญญาณเสียงและวิดีโอ (AV)

สัญญาณเสียงสเตอริโอและภาพหรือวิดีโอรวมกันภายใต้แจ็ค AV ขนาด 3.5 มม. จัดหาเหมือนกับแจ็ค AV ของเครื่องเสียงและเครื่องเล่นสัญญาณภาพสมัยใหม่ หากต้องการนำสัญญาณไปใช้ ต้องใช้สาย AV แบบ 4 ขั้วต่อ ประกอบด้วย สัญญาณภาพ, กราวด์, สัญญาณเสียงช่องซ้าย และสัญญาณเสียงช่องขวา



จุดต่อโมดูลกล้อง

ใช้เชื่อมต่อโมดูลกล้องผ่านจุดต่อ CSI-2 (Camera Serial Interface Type 2)

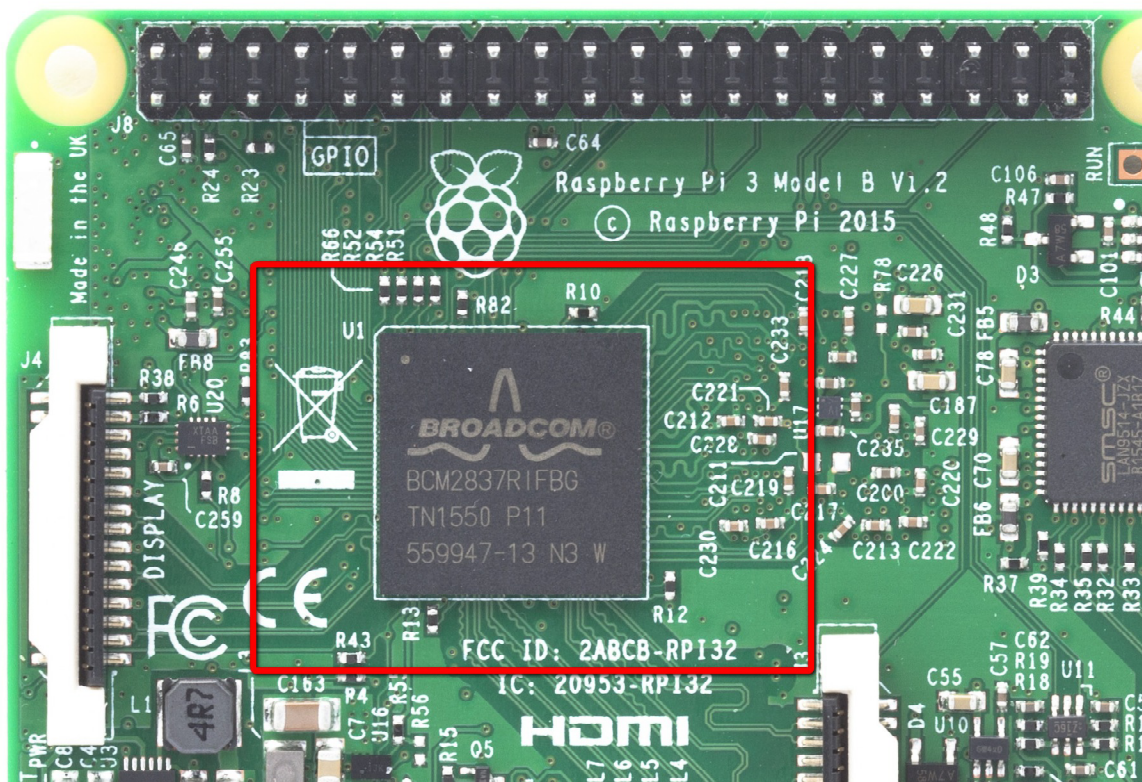
รูปที่ 1-6 ส่วนประกอบที่ควรทราบของบอร์ด Raspberry Pi 3

1.3 ส่วนประกอบที่สำคัญของ Raspberry Pi 3

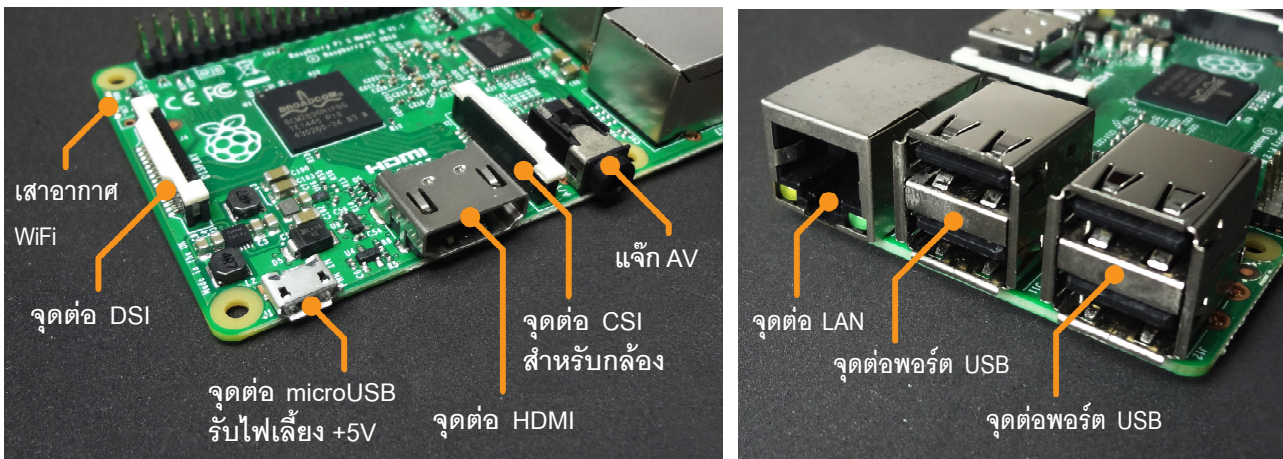
ในรูปที่ 1-6 แสดงส่วนประกอบที่สำคัญทั้งหมดของบอร์ด Raspberry Pi 3 ข้อมูลเบื้องต้นที่ควรทราบเกี่ยวกับบอร์ด Raspberry Pi 3 สรุปได้ดังนี้

1.3.1 สมองกลใหม่ 64 บิต

ชิพใหม่ของ Raspberry Pi 3 คือ Broadcom BCM2837 เป็นชิพที่ใช้ ARM Cortex-A53 แบบ 64 บิต ทำงานที่ความเร็ว 1.2GHz และมี Cache 32kB ที่ Level 1 เพื่อช่วยให้ชิพเข้าถึงคำสั่งที่ใช้บ่อยที่สุดได้เร็วขึ้น และ Cache 512kB ที่ Level 2 ช่วยให้ชิพทำงานได้เร็วขึ้นกว่าเดิม ต่างจาก Raspberry Pi 2 ที่ไม่มีหน่วยความจำแคช (Cache memory) เชื่อมต่อกับหน่วยความจำแรมที่มีขนาด 1GB เป็นแบบ LPDDR2 (Low Power DDR2) ที่กินพลังงานไฟฟ้าต่ำ และมีชิปประมวลผลกราฟิกเป็น VideoCore IV แบบ 2 แกนที่รองรับ OpenGL ES 2.0, OpenVG และรองรับ Decode H.264 ระดับ 1080p30



รูปที่ 1-7 ชิป Broadcom BCM2837 สมองกลตัวใหม่ของ Raspberry Pi 3



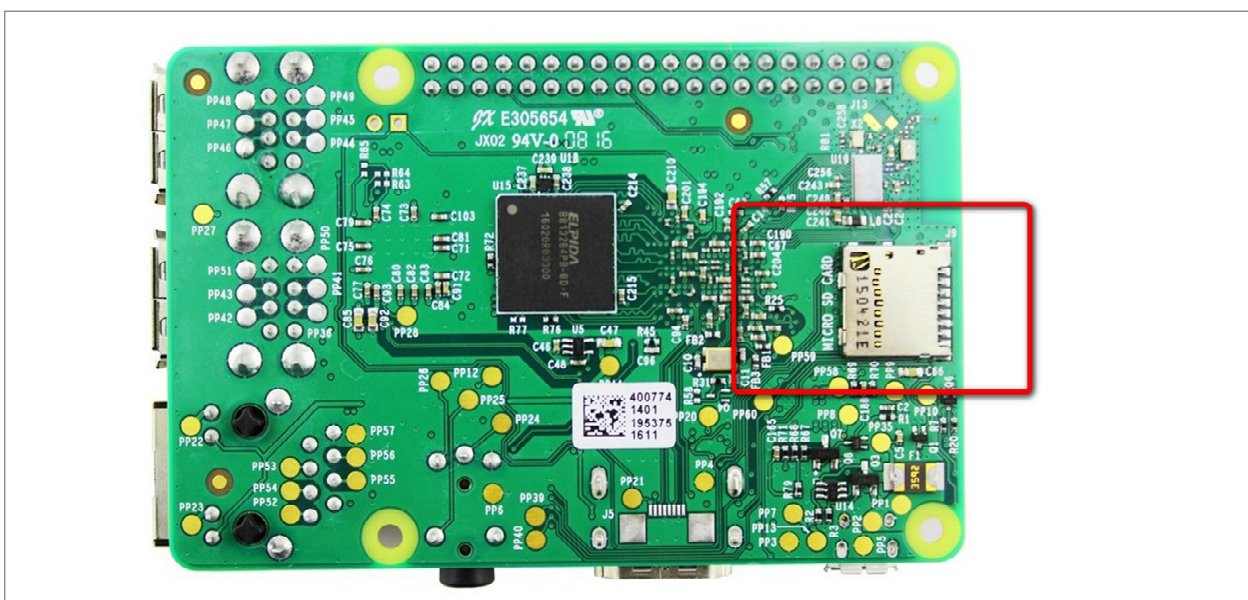
รูปที่ 1-8 แสดงส่วนเชื่อมต่ออุปกรณ์ภายนอกของบอร์ด Raspberry Pi 3

1.3.2 จุดต่ออุปกรณ์ภายนอก

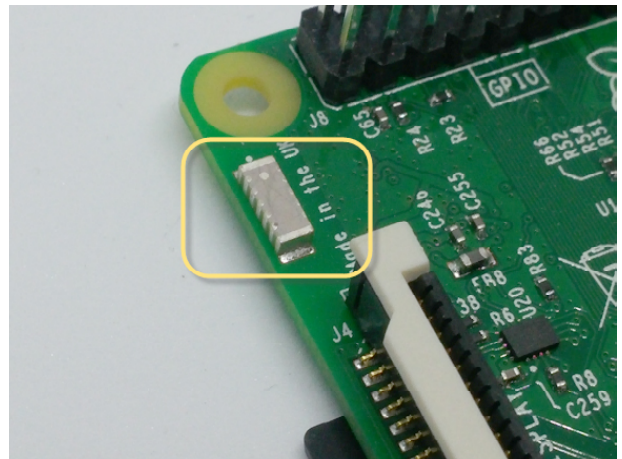
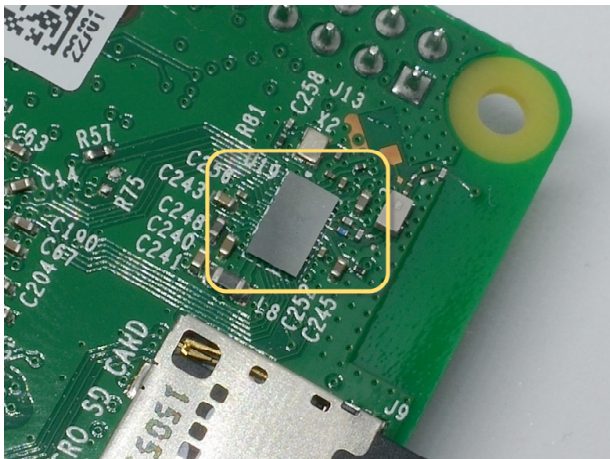
มีจุดต่อ HDMI สำหรับต่อกับจอแสดงผลความละเอียดสูง รองรับ HDMI 1.3 และ 1.4, มีแจ็ก AV 3.5 มม. สำหรับเอาต์พุตเสียงสเตอริโอและสัญญาณภาพแบบ Component Video, มีจุดต่อแหล่งจ่ายไฟ +5V ผ่านทางคอนเน็กเตอร์ microUSB, รองรับการเชื่อมต่อโมดูลกล้องผ่านคอนเน็กเตอร์ CSI (Camera Serial Interface), เชื่อมต่อกับจอแสดงผลผ่านคอนเน็กเตอร์ DSI (Display Serial Interface), รองรับการเชื่อมต่อเครือข่าย LAN ผ่านจุดต่ออีเทอร์เน็ต และจุดต่อพอร์ต USB 4 ช่อง เพื่อต่อกับอุปกรณ์ภายนอก

1.3.3 ส่วนติดต่อกับหน่วยความจำ

หน่วยความจำข้อมูลและระบบปฏิบัติการของบอร์ด Raspberry Pi 3 จะถูกเก็บไว้ในแผ่นหน่วยความจำ micro SD การ์ด โดยมีช่องเสียบ micro SD การ์ดอยู่ที่ใต้บอร์ด



รูปที่ 1-9 แสดงซ็อกเก็ตของ micro SD การ์ดของบอร์ด Raspberry Pi 3 ที่ติดตั้งอยู่ใต้บอร์ด



รูปที่ 1-10 ชิป BCM43438 ที่รวม WiFi และบลูทูธ 4.1 ไว้ในตัวติดตั้งอยู่ด้านล่างของบอร์ด ส่วนสายอากาศจะอยู่ด้านบน

1.3.4 จุดต่อพอร์ต GPIO

ที่ขาดไม่ได้เลยสำหรับบอร์ด Raspberry Pi 2 นั่นคือ พอร์ตอินพุตเอาต์พุตดิจิทัลต่อเนื่อง 40 หรือ GPIO 40 ขาเพื่อนำไปต่อเข้ากับวงจรอิเล็กทรอนิกส์ต่างๆ มีขาพอร์ตที่รองรับระดับสัญญาณลอจิก 3.3V ให้ใช้งานรวม 26 ขา รองรับ UART, I²C และ SPI แต่ไม่รองรับอินพุตเอาต์พุตอะนาล็อกโดยตรง

1.3.5 วงจรสื่อสารไร้สายผ่าน WiFi และ BLE

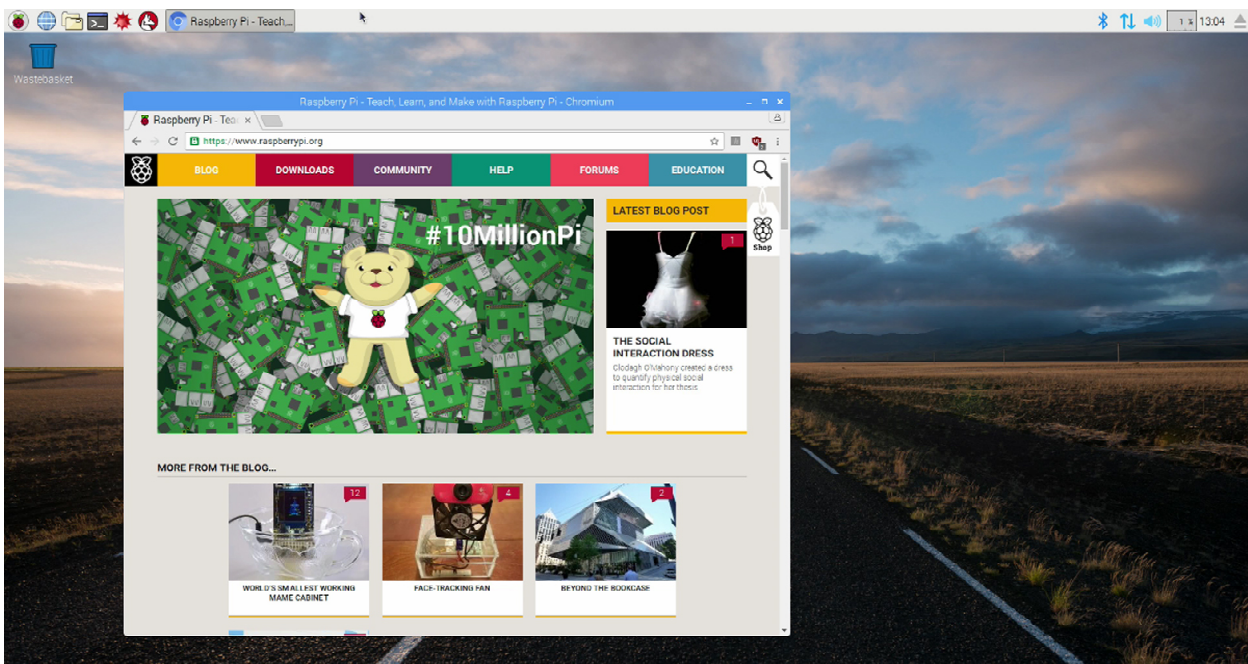
ความสามารถใหม่ที่สร้างความพอใจให้กับผู้ใช้งานอย่างมากคือ การผนวกชิป BCM43438 ซึ่งภายในมีวงจรสื่อสารข้อมูลไร้สายสมัยใหม่ทั้ง WiFi หรือ Wireless LAN และบลูทูธ 4.1 ซึ่งรองรับทั้งบลูทูธมาตรฐานและแบบพลังงานต่ำ (BLE) อยู่ในตัวเดียวกันทำให้บอร์ด Raspberry Pi 3 เชื่อมต่อเครือข่ายและอินเทอร์เน็ตในแบบไร้สายด้วย WiFi ได้เลย



รูปที่ 1-11 อะแดปเตอร์ +5V 2.5A ที่ทางผู้ผลิตคือ Raspberry Pi Foundation แนะนำให้ใช้เป็นแหล่งจ่ายไฟเลี้ยงสำหรับบอร์ด Raspberry Pi 3

1.3.6 ไฟเลี้ยงที่คู่ควร

เนื่องจาก Raspberry Pi 3 ใช้ชิพที่มีความเร็วมากขึ้น และมีการเพิ่ม WiFi กับบลูทูธ ส่งผลให้มีความต้องการใช้พลังงานมากขึ้น โดย Raspberry Pi 3 จะใช้กระแสไฟฟ้าเพิ่มเป็น 2.5A ที่ไฟเลี้ยง +5V จึงจะทำงานได้เต็มประสิทธิภาพ แต่ถ้าหากงานที่ใช้ไม่ได้เน้นเรื่องการสื่อสารข้อมูลไร้สาย อาจเลือกใช้อะแดปเตอร์ +5V 2A ได้



รูปที่ 1-12 หน้าต่างเดสก์ท็อปของระบบปฏิบัติการ Raspbian Jessie ที่จะพบเมื่อเริ่มใช้งาน Raspberry Pi 3

1.3.7 ระบบปฏิบัติการใหม่ Raspbian Jessie

ในวันที่ Raspberry Pi 3 ออกวางตลาด ทาง RaspberryPi Foundation ได้ทำการแนะนำระบบปฏิบัติการเวอร์ชันใหม่ให้ใช้งานได้ใช้งานไปพร้อมกัน นั่นคือ **Raspbian Jessie** โดยดาวน์โหลดได้ที่ <https://www.raspberrypi.org/downloads/raspbian/> โดยมีการปรับปรุงการเข้าถึงหน้าเดสก์ท็อปให้เร็วขึ้น มีหน้าต่างสำหรับตั้งค่าการทำงานต่างๆ ในแบบกราฟิกที่เป็นไดอะล็อกบ็อก ทำให้การตั้งค่าทำได้ง่าย สะดวก และดูร่วมสมัยมากขึ้น

1.4 การเตรียมใช้งาน Raspberry Pi

1.4.1 อุปกรณ์และเครื่องมือ

เตรียมอุปกรณ์ดังนี้

1. บอร์ด Raspberry Pi 3
2. แผ่นหน่วยความจำ microSD การ์ด สำหรับเก็บระบบปฏิบัติการและข้อมูล ควรมีความจุตั้งแต่ 8GB คลาส 4 ขึ้นไป ซึ่งก็คือ SDHC การ์ด แนะนำ Class 10 จะเป็นแบบ FAT32 หรือ NTFS ก็ได้
3. สาย microUSB ใช้สำหรับต่อกับแหล่งจ่ายไฟ +5V (แนะนำ 2.5A ขึ้นไป) ที่มีจุดต่อแบบ USB เช่นอะแดปเตอร์ของแท็บเล็ตแอนดรอยด์หรือวินโดวส์, เครื่องจ่ายไฟสำรองหรือเพาเวอร์แบงก์ (Power Bank) ขนาด 3,000mA ขึ้นไป ไม่ควรใช้แหล่งจ่ายไฟจากพอร์ต USB ของคอมพิวเตอร์ เนื่องจากมีความสามารถในการจ่ายกระแสไฟฟ้าไม่เพียงพอ



รูปที่ 1-13 อุปกรณ์ที่ต้องเตรียมสำหรับการเริ่มต้นใช้งานบอร์ดคอมพิวเตอร์ Raspberry Pi 3

4. อะแดปเตอร์ไฟตรง +5V 2.5A ขึ้นไป ที่มีจุดต่อแบบ USB หรือเครื่องจ่ายไฟสำรองหรือเพาเวอร์แบงก์ (Power Bank) ขนาด 3000mA ขึ้นไป หรืออะแดปเตอร์ +5V 2.5A ขึ้นไปที่มีหัวต่อแบบ microUSB หากใช้แบบนี้ สาย microUSB ในข้อ 3. ไม่ต้องใช้

5. สาย HDMI สำหรับต่อกับจอแสดงผล สำหรับผู้ใช้งานที่มีจอแสดงผลแบบ DVi และ VGA จะต้องใช้ตัวแปลงหรือสายแปลงสัญญาณ HDMI เป็น DVi หรือVGA มาต่อแทน

6. คีย์บอร์ดและเมาส์แบบ USB สำหรับควบคุมการทำงานของบอร์ด Raspberry Pi 3

7. สาย LAN (มีหรือไม่มีก็ได้) สำหรับเชื่อมต่อกับระบบเครือข่ายและอินเทอร์เน็ต

8. สาย AV ที่มีหัวเสียบด้านหนึ่งเป็นปลั๊กหูฟัง 4 ขั้ว และปลายอีกด้านหนึ่งเป็นปลั๊ก RCA 3 หัว สำหรับสัญญาณเสียงช่องซ้าย, ขวา และสัญญาณวิดีโอ หรือสายหูฟังสเตอริโอ 3.5 มม. ในกรณีที่ไม่ต้องการใช้สัญญาณวิดีโอที่จุดนี้ (มีหรือไม่มีก็ได้)

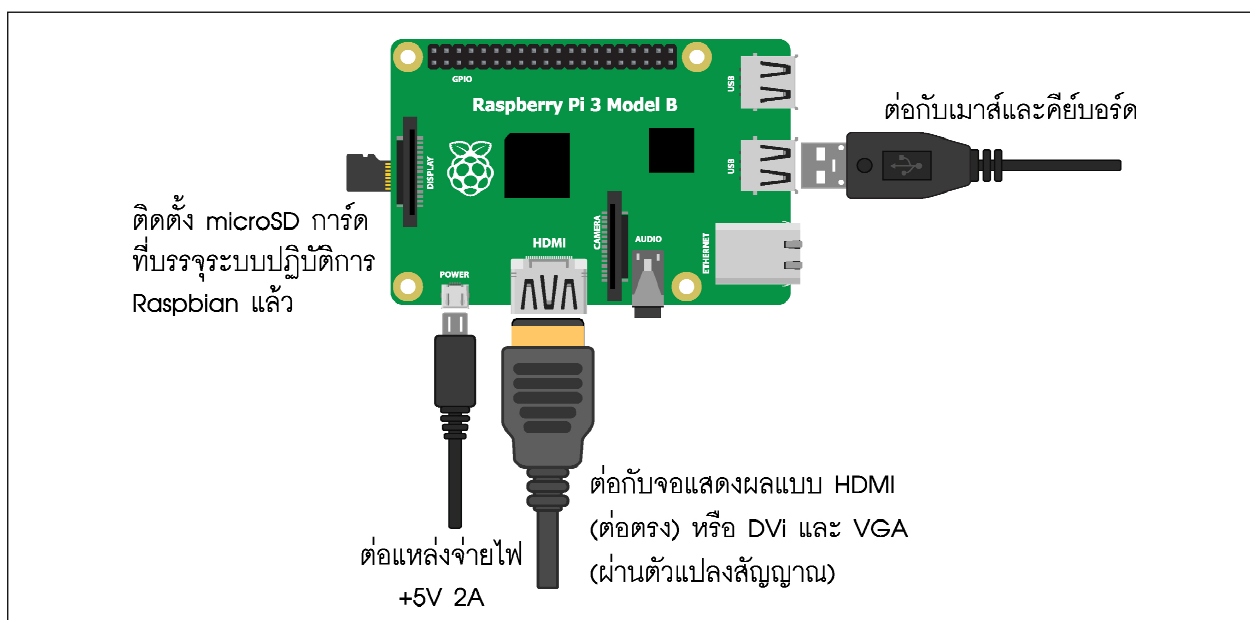
9. ลำโพงขยายเสียงที่มีจุดต่อสัญญาณอินพุตเป็นแจ็ก RCA หรือแจ็กหูฟังสเตอริโอ 3.5 มม. เพื่อต่อกับแจ็ก AV ของบอร์ด Raspberry Pi 3 (มีหรือไม่มีก็ได้)

รูปที่ 1-13 แสดงอุปกรณ์ทั้งหมดที่ควรมีสำหรับใช้งานบอร์ด Raspberry Pi 3

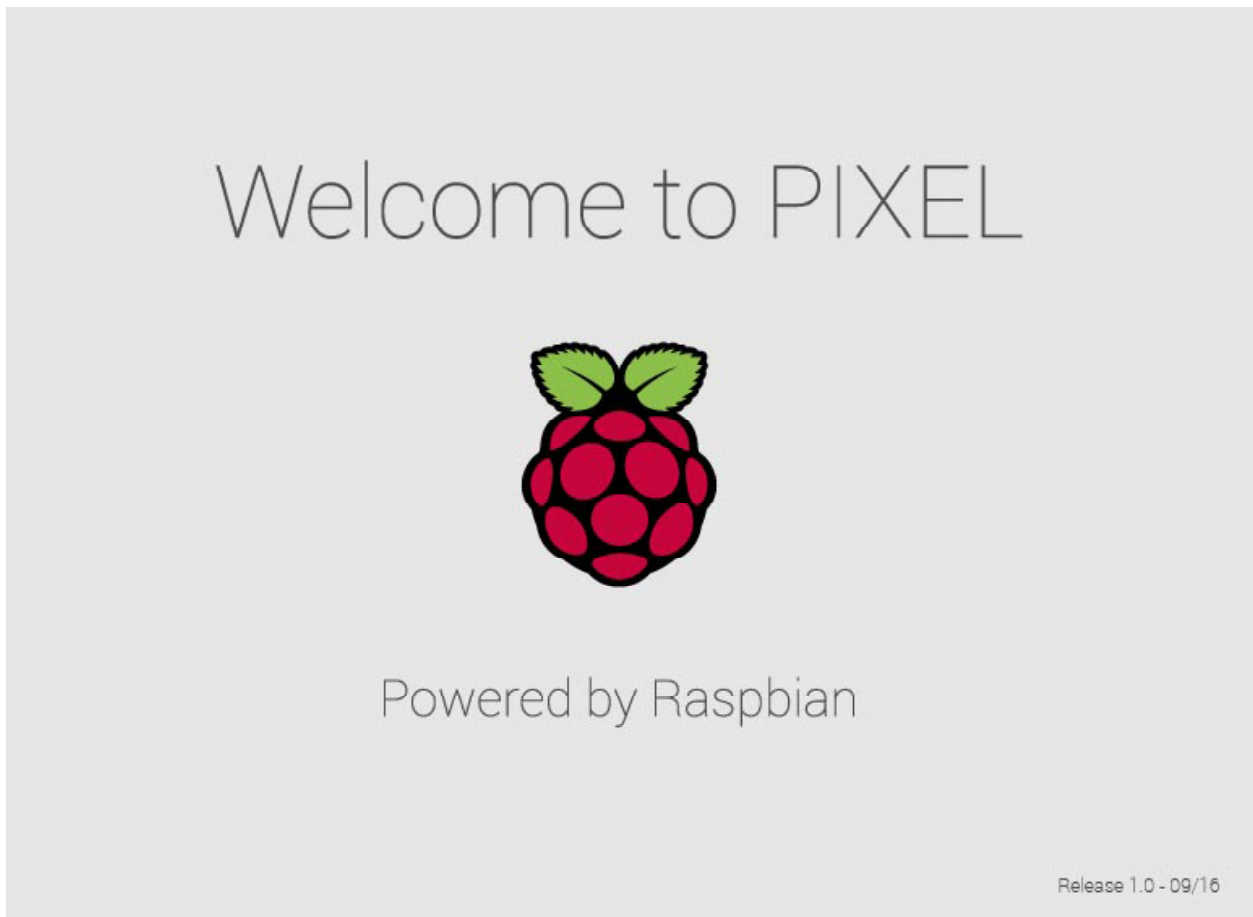
1.4.2 ทดสอบการทำงานเบื้องต้น

มีขั้นตอนโดยสรุปดังนี้

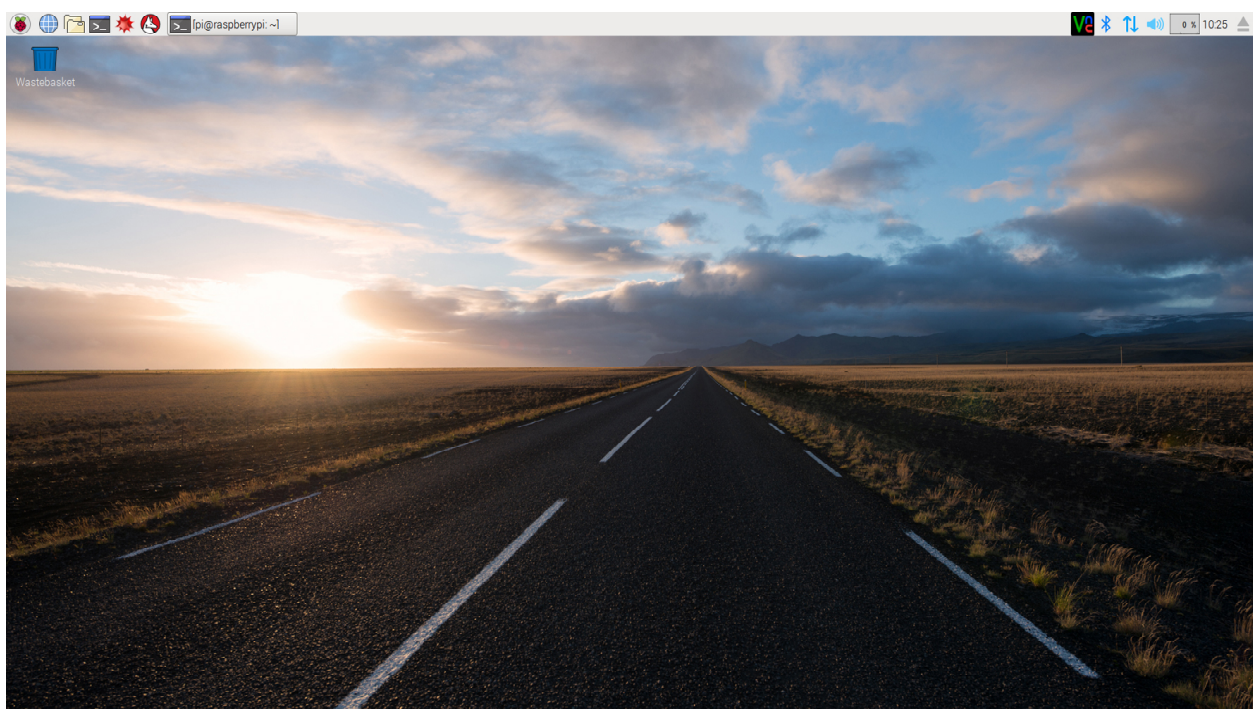
(1) ต่อสายคีย์บอร์ด, เมาส์, จอภาพ และแหล่งจ่ายไฟให้เรียบร้อย จากนั้นจ่ายไฟให้กับบอร์ด Raspberry Pi 3 ดังรูปที่ 1-14



รูปที่ 1-14 ต่ออุปกรณ์ต่าง ๆ ที่จำเป็นสำหรับเริ่มต้นใช้งานบอร์ด Raspberry Pi 3



รูปที่ 1-15 หน้าต่างแสดงการบูตระบบของ Raspberry Pi 3



รูปที่ 1-16 หน้าต่าง Desktop ของระบบปฏิบัติการ Raspbian Jessie with Pixel ในโหมดกราฟิก

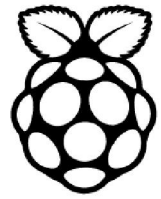
(2) จากนั้นบอร์ด Raspberry Pi 3 จะเริ่มทำงาน แสดงหน้าต่างต้อนรับเข้าสู่ระบบปฏิบัติการในรูปแบบใหม่ ดังรูปที่ 1-15 รอคูหนึ่งเมื่อบูตเรียบร้อยแล้วจะปรากฏหน้าต่างเดสก์ท็อปดังรูปที่ 1-16

ในบทนี้เป็นการแนะนำบอร์ด Raspberry Pi 3 ในขั้นต้น ข้อมูลเพิ่มเติมเกี่ยวกับการติดตั้งระบบปฏิบัติการ การตั้งค่าการทำงานและการเชื่อมต่ออุปกรณ์รอบข้าง ทั้งจอภาพ, เม้าส์ หรือคีย์บอร์ด ขอให้ดูเพิ่มเติมจากหนังสือ Raspberry Pi 3 บอร์ดคอมพิวเตอร์ขนาดเล็ก : รู้จักและใช้งานเบื้องต้น ซึ่งจัดทำโดยคณะผู้เขียนเดียวกัน โดยในเล่มบรรจุข้อมูลพื้นฐานและตัวอย่างการใช้งานเครื่องมือในการพัฒนาโปรแกรม สิ่งที่ต้องทราบ และแนวทางการเขียนโปรแกรมด้วยภาษา Python ก่อนจะเข้าไปสู่การทดลองเชื่อมต่อกับอุปกรณ์ภายนอกผ่านพอร์ต GPIO ในลำดับต่อไป



บทที่ 2

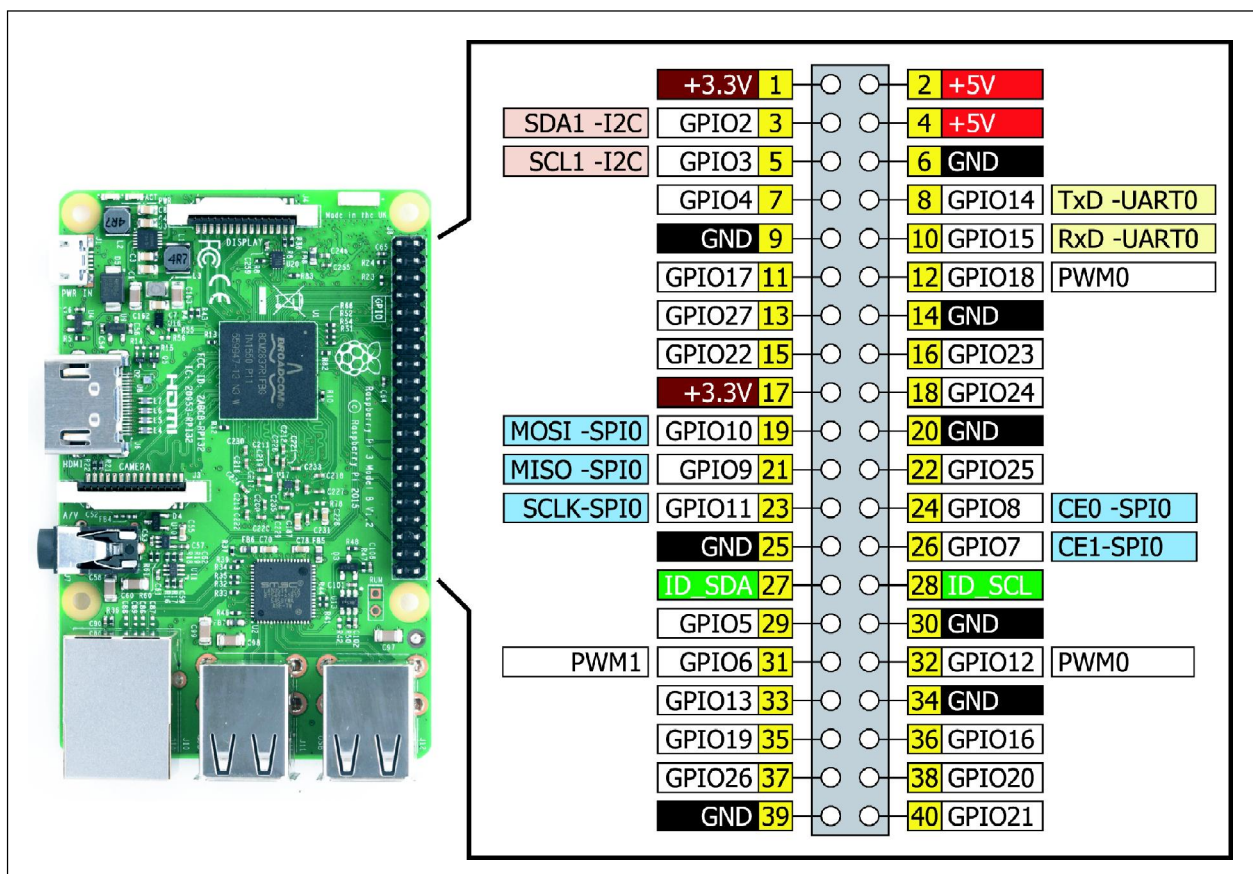
พอร์ตอินพุตเอาต์พุตอเนกประสงค์ หรือ GPIO ของ Raspberry Pi 3



จุดเด่นข้อหนึ่งที่สำคัญและทำให้บอร์ด Raspberry Pi แตกต่างจากบอร์ดคอมพิวเตอร์แผงเดี่ยวทั่วไปคือ Raspberry Pi มีจุดต่อพอร์ตอินพุตเอาต์พุตอเนกประสงค์หรือ GPIO (General Purpose Input Output) สำหรับนำไปเชื่อมต่อกับวงจรอิเล็กทรอนิกส์ภายนอกได้จึงทำให้นับว่าบอร์ด Raspberry Pi ไปสร้างสรรค์โครงงานและสิ่งประดิษฐ์ได้อย่างมากมาย และนั่นคือจุดขายที่สำคัญที่นำไปโปรแกรมเมอร์มาพบกับนักทดลองวงจรอิเล็กทรอนิกส์ ส่งผลให้เกิดกลุ่มของผู้ใช้งานที่กว้างขึ้นอย่างมาก

2.1 การจัดขา GPIO

บอร์ด Raspberry Pi 3 มีคอนเน็กเตอร์สำหรับขาพอร์ต GPIO 40 ขา แบ่งเป็น 2 แถว แถวละ 20 ขา โดยแต่ละขาจะมีหน้าที่ต่างกันดังแสดงในรูปที่ 2-1 จะเห็นได้ว่า มีขาพอร์ตอินพุตเอาต์พุตใช้งานรวม



รูปที่ 2-1 แสดงตำแหน่งของคอนเน็กเตอร์ GPIO 40 ขาบนบอร์ด Raspberry Pi 3

26 ขา มีระดับสัญญาณลอจิกเป็น 0 และ +3.3V จึงทำให้ Raspberry Pi 3 สามารถเชื่อมต่ออุปกรณ์อิเล็กทรอนิกส์ได้หลากหลาย ตั้งแต่ LED, สวิตช์ ไปจนถึงไอซีหน้าที่พิเศษหรือตัวตรวจจับต่างๆ ที่ใช้การเชื่อมต่อผ่านบัส UART, I²C และ SPI

ขาพอร์ตอินพุตเอาต์พุตของ Raspberry Pi 3 มีทั้งสิ้น 26 ขา มีการแบ่งปันหน้าที่การทำงานดังนี้

1. เป็นขาพอร์ตอินพุตเอาต์พุตดิจิทัล 26 ขา
2. ใช้งานเป็นขาเอาต์พุตสัญญาณ PWM ได้ 26 ขา (ด้วยการทำงานทางซอฟต์แวร์) โดยมีขาเอาต์พุต PWM แบบฮาร์ดแวร์ 4 ขาคือ GPIO12 กับ GPIO18 (PMW0) และ GPIO13 กับ GPIO19 (PWM1)
3. ใช้งานเป็นขาเชื่อมต่อบัส I²C 1 ช่อง (2 ขา : SDA และ SCL โดยแบ่งปันการทำงานกับขาพอร์ต GPIO2 และ 3)
4. ใช้งานเป็นขาเชื่อมต่อพอร์ตอนุกรมสำหรับสื่อสารข้อมูลอนุกรม UART 1 ช่อง (2 ขา : TxD และ RxD โดยแบ่งปันการทำงานกับขาพอร์ต GPIO14 และ 15)
5. ขาเชื่อมต่อบัส SPI 1 ช่อง (3 ขาหลัก : MOSI, MISO, SCK โดยแบ่งปันการทำงานกับขาพอร์ต GPIO10, 9 และ 11 ตามลำดับ พร้อมกับ 2 ขาเสริม : CE0 และ CE1 โดยแบ่งปันการทำงานกับขาพอร์ต GPIO8 และ 7 ตามลำดับ)

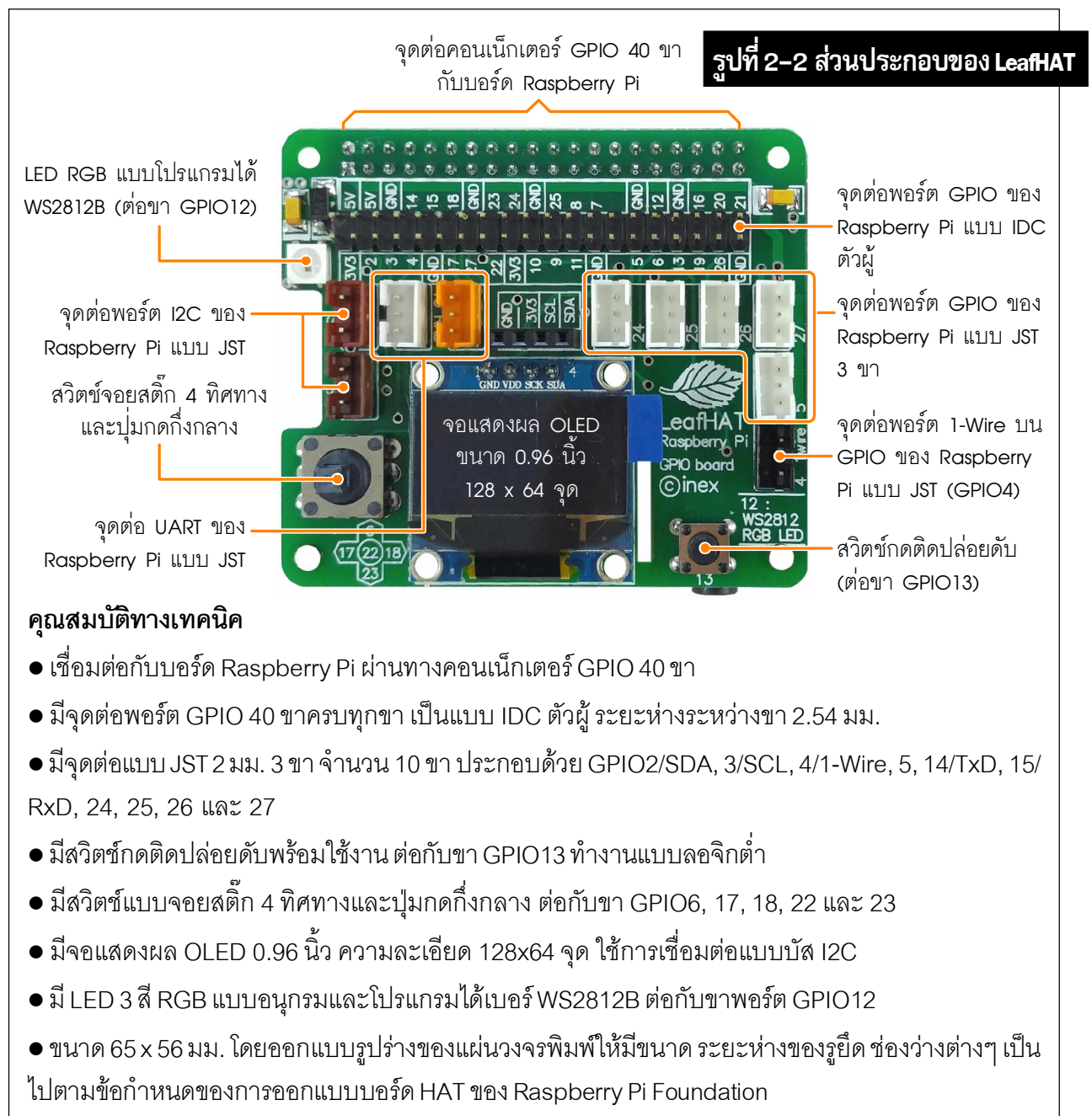
นอกจากนี้ยังมีขาพอร์ตสำหรับเชื่อมต่อหน่วยความจำอีอีพรอมอนุกรมภายนอก (Serial EEPROM) นั่นคือ ขา **ID_SCL** (ขาสัญญาณนาฬิกา) และ **ID_SDA** (ขาข้อมูล) เพื่อใช้กำหนดชื่อของฮาร์ดแวร์ที่นำมาต่อกับพอร์ต GPIO ของ Raspberry Pi 3 ภายใต้มาตรฐาน **HAT** (Hardware Attached on Top)

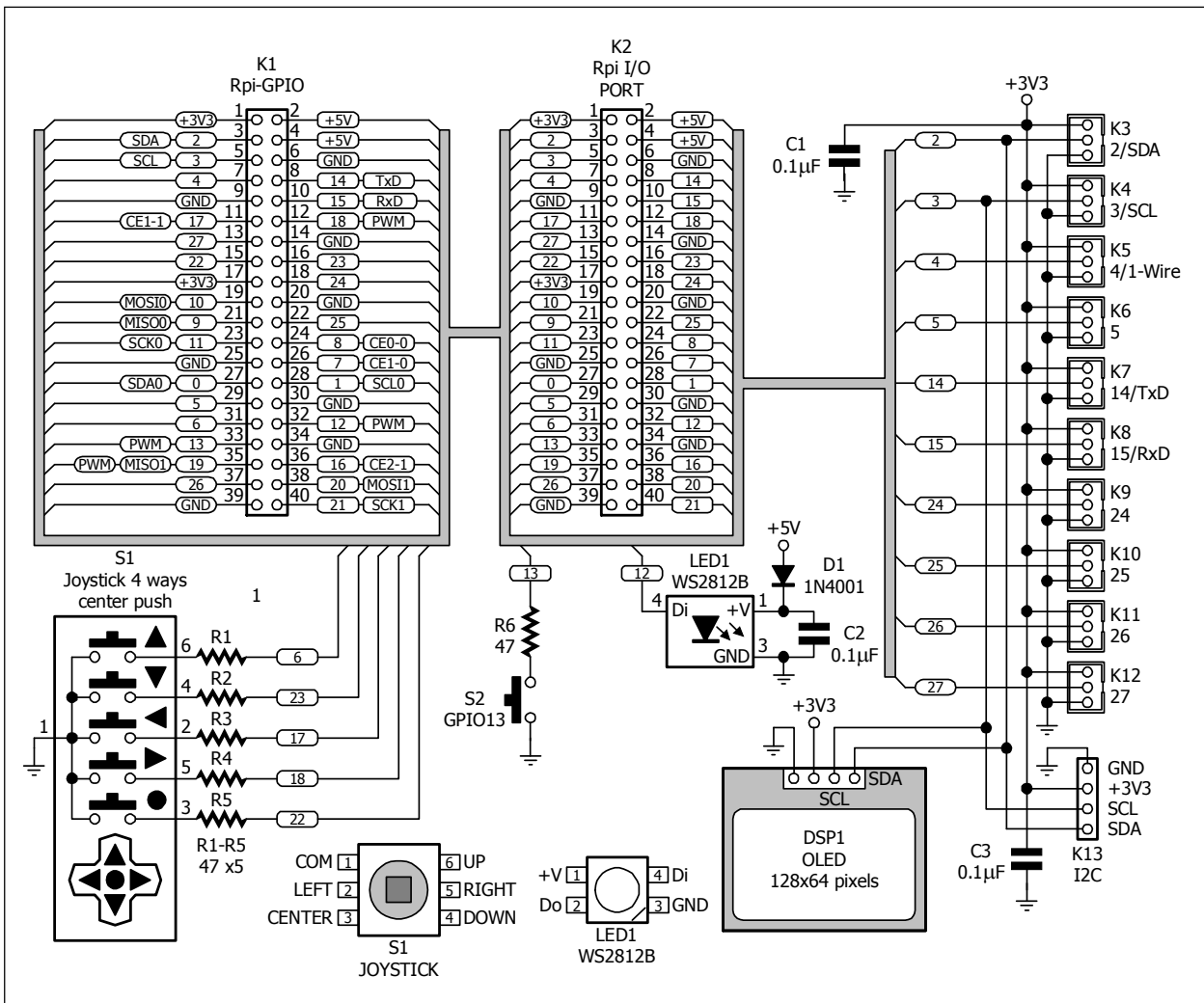
ขาพอร์ต GPIO ทั้งหมดรับและขับสัญญาณได้สูงสุด +3.3V ไม่สามารถรับแรงดัน +5V ได้เมื่อทำงานเป็นพอร์ตอินพุต ด้านความสามารถในการรับและจ่ายกระแสไฟฟ้าของพอร์ต GPIO แต่ละขาไม่เกิน 50mA แต่รวมทุกขาพอร์ตจะจ่ายกระแสไฟฟ้าได้ไม่เกิน 300mA เมื่อทำงานเป็นพอร์ตเอาต์พุต

ดังนั้นในการใช้งานขาพอร์ต GPIO ของ Raspberry Pi จึงต้องระมัดระวังอย่าป้อนแรงดัน +5V เข้าที่ขาพอร์ตโดยตรงอย่างเด็ดขาด เพราะจะทำให้วงจรของขาพอร์ต GPIO เสียหายได้ในทันที แม้ว่าที่จุดต่อหรือคอนเน็กเตอร์ GPIO ของ Raspberry Pi จะมีจุดต่อ +5V ไว้ให้ใช้งานก็ตาม ทั้งนี้การมีจุดต่อแรงดัน +5V (ซึ่งมาจากพอร์ต microUSB ของบอร์ด Raspberry Pi) ก็เพื่อประโยชน์ในการเป็นไฟเลี้ยงให้อุปกรณ์ต่อพ่วงที่นำมาต่อกับบอร์ด Raspberry Pi ซึ่งต้องการไฟเลี้ยง +5V จะได้ไม่ต้องจัดหาแหล่งจ่ายไฟเลี้ยงเพิ่มเติม ทำให้การใช้งานพอร์ต GPIO ของ Raspberry Pi กับอุปกรณ์ต่อพ่วงภายนอกทำได้สะดวกขึ้นอย่างมาก

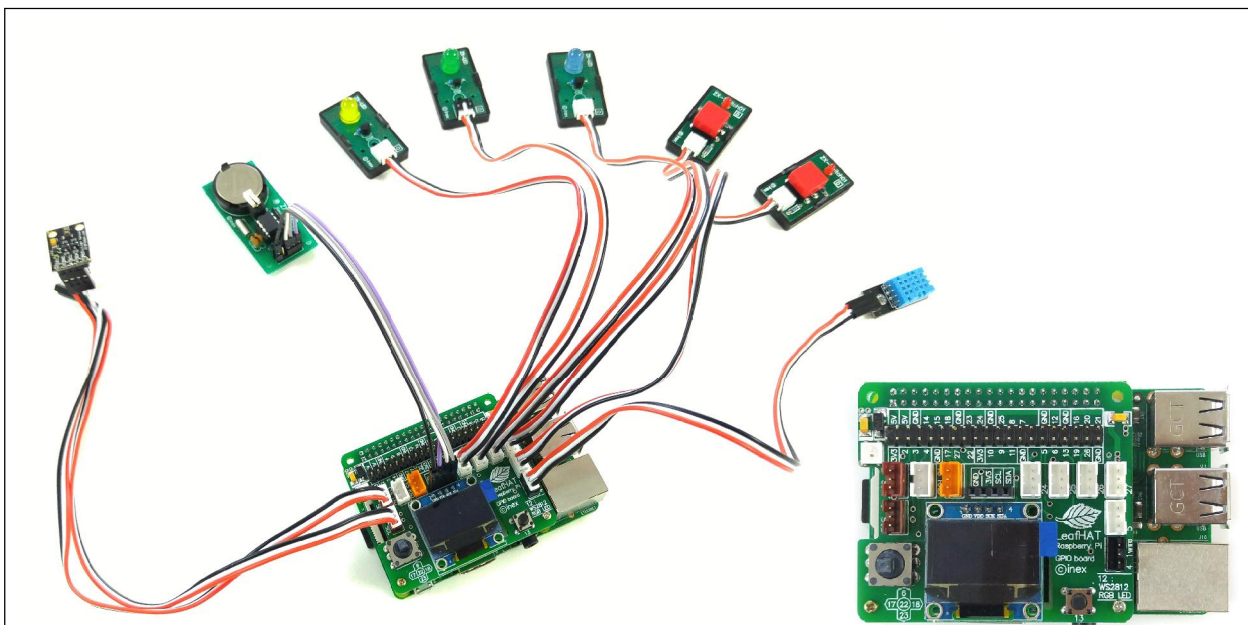
2.2 LeafHAT บอร์ดเชื่อมต่ออุปกรณ์อินพุตเอาต์พุตสำหรับพอร์ต GPIO ของ Raspberry Pi

เพื่อให้การทดลองเชื่อมต่ออุปกรณ์ภายนอกกับพอร์ต GPIO ของ Raspberry Pi ง่ายสำหรับนักทดลองมือใหม่ และสะดวกสำหรับนักทดลองที่มีประสบการณ์มาแล้ว จึงขอแนะนำบอร์ดเชื่อมต่ออุปกรณ์อินพุตเอาต์พุตที่ชื่อ **LeafHAT** มาใช้เป็นหลักในทุกตัวอย่างการทดลองของหนังสือเล่มนี้ โดย LeafHAT ได้รับการออกแบบภายใต้ข้อกำหนดของการพัฒนาบอร์ด HAT (Hardware Attach on Top) ของ Raspberry Pi Foundation สำหรับติดตั้งเข้ากับพอร์ต GPIO 40 ขาของบอร์ด Raspberry Pi





รูปที่ 2-3 วงจรสมบูรณของบอร์ด LeafHAT



รูปที่ 2-4 การต่อใช้งาน LeafHAT ร่วมกับ Raspberry Pi 3 และแผงวงจร อุปกรณ์อินพุตเอาต์พุตแบบต่าง ๆ

ในรูปที่ 2-2 แสดงส่วนประกอบและคุณสมบัติทางเทคนิคที่สำคัญของบอร์ด LeafHAT ส่วนวงจรสมบูรณ์แสดงในรูปที่ 2-3

การใช้งานร่วมกับ Raspberry Pi 3 ทำได้ง่ายมาก เพียงเสียบบอร์ด LeafHAT เข้ากับคอนเน็กเตอร์ GPIO 40 ขาของบอร์ด Raspberry Pi 3 ตามทิศทางให้ถูกต้อง ดังรูปที่ 2-4 บอร์ด LeafHAT ก็พร้อมใช้งานกับ Raspberry Pi 3 แล้ว

บอร์ด LeafHAT ยังใช้งานได้กับบอร์ด Raspberry Pi ทุกรุ่นที่มีคอนเน็กเตอร์ GPIO 40 ขา โดยไม่ต้องดัดแปลงแต่อย่างใด

2.3 การเขียนโปรแกรมเพื่อติดต่อกับพอร์ต GPIO บนบอร์ด Raspberry Pi

สำหรับการเขียนโปรแกรมเพื่อติดต่อควบคุม, อ่านค่า และส่งข้อมูลหรือสัญญาณไปยังขาพอร์ต GPIO บนบอร์ด Raspberry Pi 3 ทำได้หลายวิธี ไม่ว่าจะเป็นการใช้โปรแกรมภาษา Python, C/C++, Java หรือแม้แต่ Javascript ก็ตาม

สำหรับในหนังสือเล่มนี้ (และในอนุกรมเดียวกัน) แนะนำให้เขียนโปรแกรมด้วยภาษา Python เพื่อสร้างไฟล์สคริปต์ โดยมีไลบรารีที่ชื่อว่า **RPi.GPIO** เป็นหัวใจหลัก ซึ่งในระบบปฏิบัติการ Raspbian เวอร์ชันใหม่ๆ จะติดตั้งไลบรารี RPi.GPIO มาให้แล้ว จึงไม่จำเป็นต้องติดตั้งเพิ่ม

แต่ถ้ายังไม่ได้ติดตั้ง ให้เข้าสู่หน้าต่างเทอร์มินอล แล้วพิมพ์คำสั่งที่เชลพร้อมพ์ ดังนี้

```
sudo apt-get install python3-rpi.gpio
```

2.4 การกำหนดขา GPIO

บอร์ด Raspberry Pi 3 จะมีการเรียกใช้งานขา GPIO ในภาษา Python อยู่ 2 แบบคือ อิงตามเลขขาบนบอร์ด (Board) และอิงตามขาของตัวชิป Broadcom (BCM) การเลือกรูปแบบการจดจำใช้คำสั่ง

```
GPIO.setmode(GPIO.BOARD)
```

```
GPIO.setmode(GPIO.BCM)
```

ตัวอย่างในหนังสือเล่มนี้ใช้การอ้างอิงขาพอร์ตตามขาของตัวชิป *Broadcom (GPIO.BCM)*

การเรียนรู้เกี่ยวกับโครงสร้างภาษาไพธอน (Python) ตลอดจนคำสั่งและการเขียนโปรแกรมโดยใช้ซอฟต์แวร์ที่ชื่อ Geany ดูเพิ่มเติมได้จาก *บทที่ 5 และ 6 ของหนังสือ Raspberry Pi 3 บอร์ดคอมพิวเตอร์ : รู้จักและใช้งานเบื้องต้น* และ *หนังสือรายวิชาเพิ่มเติม เทคโนโลยีสารสนเทศและการสื่อสาร ภาษาไพธอน จัดพิมพ์โดย สถาบันส่งเสริมการสอนวิทยาศาสตร์และเทคโนโลยี (สสวท.)* ที่จัดมาพร้อมกันในชุด Raspberry Pi 3 Education Kit ที่จัดทำโดยบริษัท อินโนเวทีฟ เอกซ์เพอริเมนต์ จำกัด (INEX - www.inex.co.th)

2.5 ตัวอย่างการติดต่อพอร์ต GPIO ของบอร์ด Raspberry Pi 3 เพื่อใช้งานขาพอร์ตเอาต์พุตเบื้องต้น ด้วยภาษา Python

ตัวอย่างแรกของการใช้งาน GPIO ของบอร์ด Raspberry Pi เบื้องต้น เป็นการใช้งานขา GPIO5 เป็นขาพอร์ตเอาต์พุตเพื่อขับ LED ให้ติดดับสลับกัน หรือทำงานเป็นไฟกะพริบ 1 ดวง

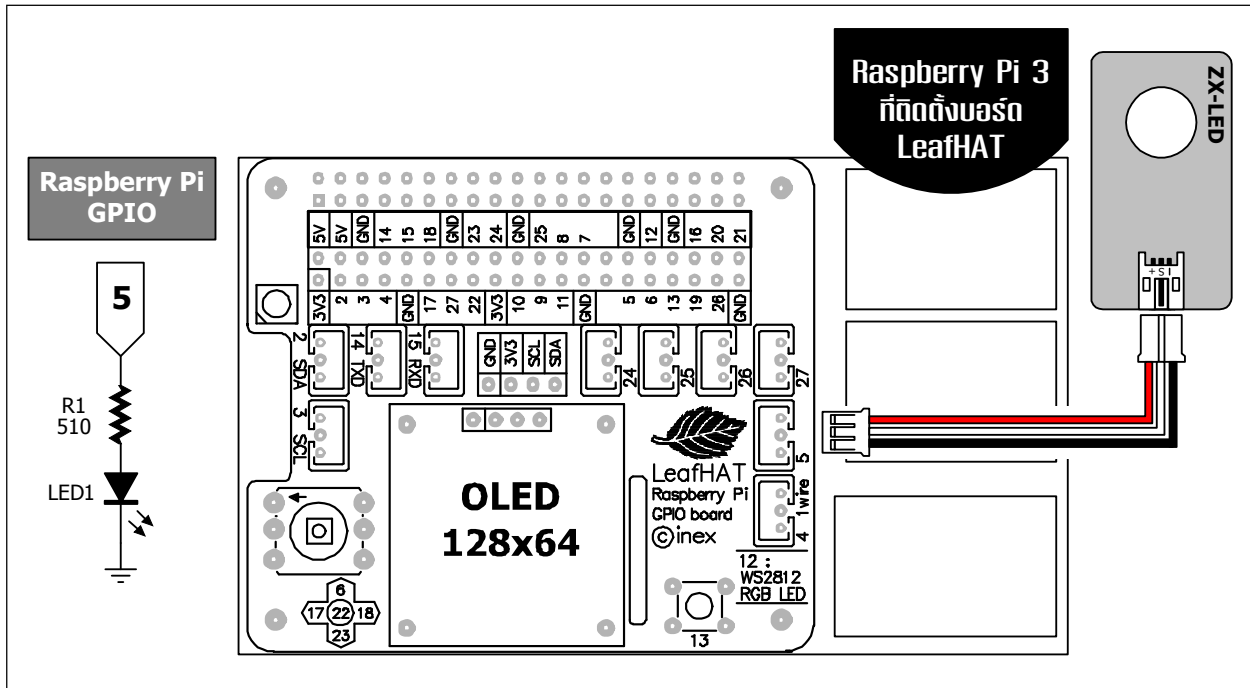
(2.5.1) ต่อยังจอร์เพื่อเชื่อมต่อบอร์ด Raspberry Pi 3 กับ LED ดังวงจรในรูปที่ 2-5 โดยใช้บอร์ด LeafHAT และ ZX-LED ในการทดลอง

(2.5.2) เปิดซอฟต์แวร์ Geany เพื่อสร้างไฟล์สคริปต์ โดยเขียนโปรแกรมที่ 2-1

(2.5.3) บันทึกไฟล์สคริปต์ โดยตั้งชื่อว่า **LED_Blinking.py**

(2.5.4) ทำการรันไฟล์สคริปต์ โดยไปที่เมนู **Bulid > Execute**

หน้าต่าง *Terminal* จะปรากฏขึ้นมาเพื่อแสดงสถานะลอจิกของขาพอร์ต และ LED ที่ต่อกับขาพอร์ต GPIO5 จะติดดับด้วยอัตรา 1 วินาที ถ้าต้องการหยุดการทำงานให้กด *Ctrl* และ *C* (กดคีย์ *Ctrl* ค้างไว้ แล้วกดคีย์ *C* ตาม)



รูปที่ 2-5 วงจรทดลองใช้งานขาพอร์ต GPIO ของบอร์ด Raspberry Pi 3 อย่างง่ายด้วยการสร้างวงจรไฟกะพริบ โดยใช้บอร์ด LeafHAT และ ZX-LED

```
import RPi.GPIO as GPIO
import time

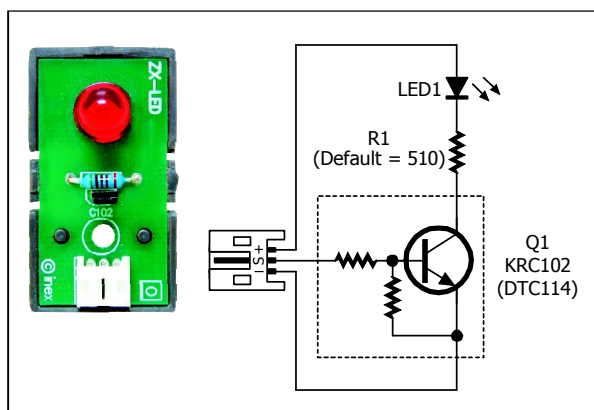
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.OUT)
while True:
    GPIO.output(5, 0)
    print('GPIO5=OFF')
    time.sleep(1)
    GPIO.output(5, 1)
    print('GPIO5=ON')
    time.sleep(1)
```

โปรแกรมที่ 2-1 โค้ดภาษา Python สำหรับใช้งานขาพอร์ต GPIO5 ของ Raspberry Pi 2 ขับ LED (ใช้ Python3 คอมไพเลอร์)

คุณสมบัติของอุปกรณ์อินพุตเอาต์พุตพื้นฐานที่ใช้ในการทดลอง

ZX-LED : มินิบอร์ด LED

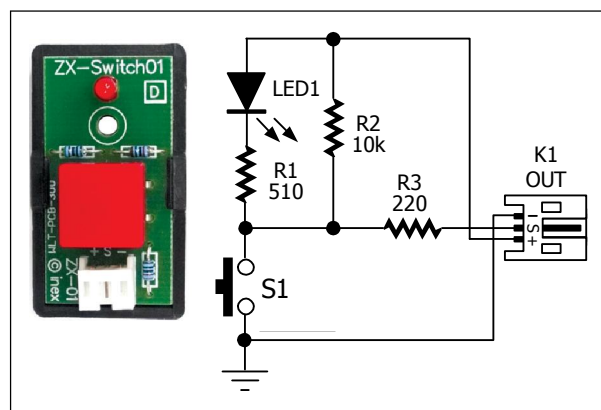
ใช้ LED ขนาด 8 มิลลิเมตร ขับให้สว่างโดยทรานซิสเตอร์ ต้องการลอจิก “1” ในการทำงาน มีรูปร่างและวงจรแสดงในรูปที่ 2A ใช้ไฟเลี้ยงได้ทั้ง +3.3V และ +5V



รูปที่ 2A รูปร่างและวงจรของมินิบอร์ดสำหรับแสดงผล ZX-LED

ZX-SWITCH01 : มินิบอร์ดสวิตช์ 1 ช่อง

มีวงจรแสดงในรูปที่ 2B ประกอบด้วยสวิตช์กดติดปล่อยดับพร้อม LED แสดงผล ให้เอาต์พุตคือ หากมีการกดสวิตช์ จะส่งลอจิก “0” (ระดับแรงดัน 0V) และไฟสีแดงติด



รูปที่ 2B รูปร่างและวงจรของ ZX-SWITCH01 มินิบอร์ดสวิตช์กดติดปล่อยดับพร้อม LED แสดงผล

2.6 ใช้งานพอร์ต GPIO ของ Raspberry Pi 3 เป็นพอร์ตอินพุตเอาต์พุตดิจิทัล

พอร์ตอินพุตดิจิทัลและเอาต์พุตดิจิทัลเป็นคุณสมบัติพื้นฐานของ GPIO ของ Raspberry Pi 3 เป็นการสั่งให้สถานะของขา GPIO เป็นลอจิกสูง (High) หรือลอจิกต่ำ (Low) ในการใช้งานพอร์ตอินพุตเอาต์พุตดิจิทัลทุกครั้งจะต้องกำหนดให้ขา GPIO นั้นๆ ทำงานแบบใด เพราะ GPIO ของ Raspberry Pi 3 ไม่สามารถทำงานเป็นพอร์ตอินพุตและเอาต์พุตดิจิทัลได้พร้อมกัน

2.6.1 การกำหนดทิศทางของขาพอร์ต

ตัวอย่างการกำหนดการทำงานให้กับขา GPIO

```
GPIO.setup(5, GPIO.OUT)
```

กำหนดให้ขา GPIO5 เป็นพอร์ตเอาต์พุตดิจิทัล

```
GPIO.setup(6, GPIO.IN)
```

กำหนดให้ขา GPIO6 เป็นพอร์ตอินพุตดิจิทัล

สำหรับขาพอร์ตอินพุตดิจิทัลยังกำหนดได้ด้วยว่าจะให้มีการต่อพูลอัพ (pull up - กำหนดให้เป็นลอจิกสูง) หรือพูลดาวน์ (pull down - กำหนดให้เป็นลอจิกต่ำ) ให้กับขาพอร์ตในขณะยังไม่มีกระแสสัญญาณเข้ามา โดยกำหนดได้ดังนี้

สำหรับพูลอัพ :

```
GPIO.setup(5, GPIO.IN, pull_up_down==GPIO.PUD_UP)
```

สำหรับพูลดาวน์ :

```
GPIO.setup(5, GPIO.IN, pull_up_down==GPIO.PUD_DOWN)
```

2.6.2 พอร์ตเอาต์พุตดิจิทัล

เมื่อต้องการให้ขา GPIO ขับสัญญาณตามที่กำหนด จะต้องกำหนดให้ทำงานเป็นพอร์ตเอาต์พุตดิจิทัล สัญญาณที่ขับได้ มี 2 สถานะคือ HIGH หรือลอจิกสูง หรือ “1” กับ LOW หรือลอจิกต่ำ หรือ “0”

รูปแบบคำสั่ง

```
GPIO.output(ขาพอร์ต, สถานะ)
```

ตัวอย่างที่ 2-1

```
GPIO.output(19, GPIO.HIGH)
```

กำหนดให้ขา GPIO19 ขับสัญญาณลอจิกสูง

```
GPIO.output(2, GPIO.LOW)
```

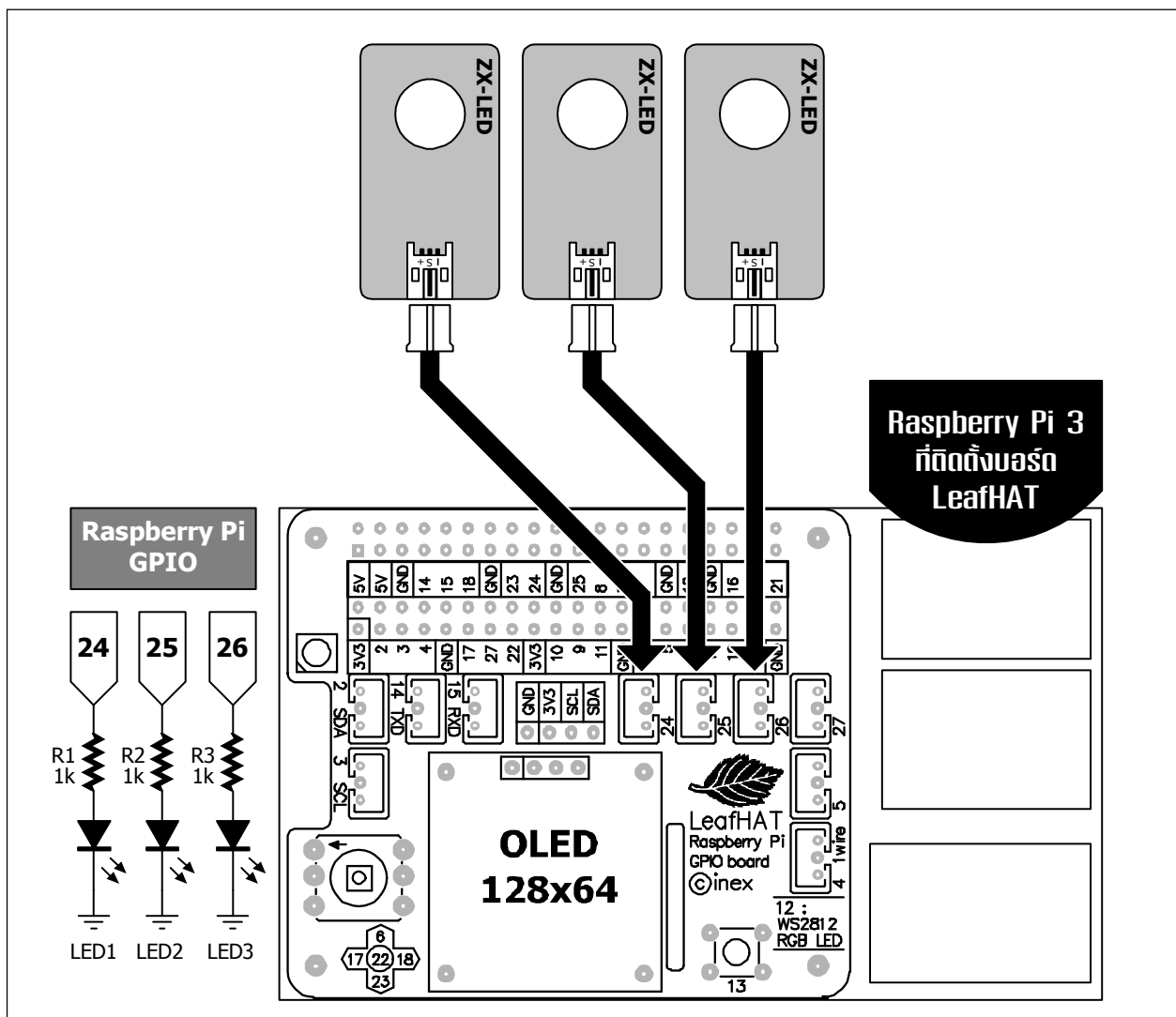
กำหนดให้ขา GPIO2 ขับสัญญาณลอจิกต่ำ

2.6.3 ทดลองสร้างไฟวิ่ง LED 3 ดวงอย่างง่าย

- (1) ต่อยังจตามรูปที่ 2-6
- (2) เปิดซอฟต์แวร์ Geany แล้วสร้างไฟล์สคริปต์ จากนั้นพิมพ์คำสั่งตามโปรแกรมที่ 2-2
- (3) ทำการบันทึกไฟล์ **LED3Scrolling.py**
- (4) รันไฟล์สคริปต์โดยไปที่เมนู **Bulid > Execute** หรือรันผ่านเทอร์มินอลด้วยคำสั่ง

```
sudo python3 LED6Scrolling.py
```

เมื่อโปรแกรมเริ่มทำงาน LED ทั้ง 3 ดวงจะติดดับในลักษณะไฟวิ่ง ถ้าต้องการหยุดทำงาน กด **Ctrl** และ **C** (กดคีย์ **Ctrl** ค้างไว้ แล้วกดคีย์ **C** ตาม)



รูปที่ 2-6 วงจรทดลองใช้งานขาพอร์ต GPIO ของบอร์ด Raspberry Pi 3 สร้างวงจรไฟวิ่ง 3 ดวง โดยใช้บอร์ด LeafHAT และ ZX-LED

```

import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

pin1 = 24
pin2 = 25
pin3 = 26

GPIO.setup(pin1, GPIO.OUT)
GPIO.setup(pin2, GPIO.OUT)
GPIO.setup(pin3, GPIO.OUT)

GPIO.output(pin1, False)
GPIO.output(pin2, False)
GPIO.output(pin3, False)

while(1):
    GPIO.output(pin1, GPIO.HIGH)
    time.sleep(0.3)
    GPIO.output(pin1, GPIO.LOW)
    GPIO.output(pin2, GPIO.HIGH)
    time.sleep(0.3)
    GPIO.output(pin2, GPIO.LOW)
    GPIO.output(pin3, GPIO.HIGH)
    time.sleep(0.3)
    GPIO.output(pin3, GPIO.LOW)

```

โปรแกรมที่ 2-2 โค้ดภาษา Python สำหรับใช้งานขาพอร์ต GPIO ของบอร์ด Raspberry Pi 3 เป็นขาพอร์ตเอาต์พุตดิจิทัลเพื่อขับ LED 3 ดวง

```

import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

pin_set = [24, 25, 26]

for pin in pin_set:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, False)

while(1):
    for pin in pin_set:
        GPIO.output(pin, GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(pin, GPIO.LOW)

```

โปรแกรมที่ 2-3 โค้ดภาษา Python สำหรับใช้งานขาพอร์ต GPIO ของบอร์ด Raspberry Pi 3 เป็นขาพอร์ตเอาต์พุตดิจิทัลเพื่อขับ LED 3 ดวง ที่ปรับปรุงจากโปรแกรมที่ 2-2 ให้กระชับมากขึ้น

2.6.4 สร้างไฟวิ่ง LED 3 ดวงแบบกระซิบ

- (1) ยังคงใช้วงจรในรูปที่ 2-6 ในการทดลอง
- (2) เปิดซอฟต์แวร์ Geany แล้วสร้างไฟล์สคริปต์ จากนั้นพิมพ์คำสั่งตามโปรแกรมที่ 2-3
- (3) ทำการบันทึกไฟล์ชื่อ **LED3Scrolling-Short.py**
- (4) รันไฟล์สคริปต์โดยไปที่เมนู **Bulid > Execute** หรือรันผ่านเทอร์มินอลด้วยคำสั่ง

```
sudo python3 LED3Scrolling-Short.py
```

เมื่อโปรแกรมเริ่มทำงาน LED ทั้ง 3 ดวงจะติดดับในลักษณะไฟวิ่งเหมือนกับโปรแกรม *LED3Scrolling* แต่ถ้าพิจารณาจากโค้ดแล้ว จะพบว่า โปรแกรม *LED3Scrolling-Short* จะสั้นและกระชับกว่า

2.7 พอร์ตอินพุตดิจิตอล

เมื่อต้องการสั่งงานให้ขา GPIO รับสัญญาณดิจิตอลจากภายนอก ต้องกำหนดให้ GPIO ทำงานเป็นพอร์ตอินพุตสัญญาณที่รับได้มี 2 สถานะคือ HIGH หรือลอจิกสูง หรือ “1” กับ LOW หรือลอจิกต่ำ หรือ “0” โดยพอร์ตอินพุตดิจิตอลมีวิธีรับสัญญาณได้ 2 แบบใหญ่ๆ คือ แบบซิงโครนัส (synchronous) และอะซิงโครนัส (asynchronous)

2.7.1 พอร์ตอินพุตดิจิตอลแบบซิงโครนัส

การใช้งานพอร์ตอินพุตดิจิตอลแบบซิงโครนัสมี 2 วิธีคือ

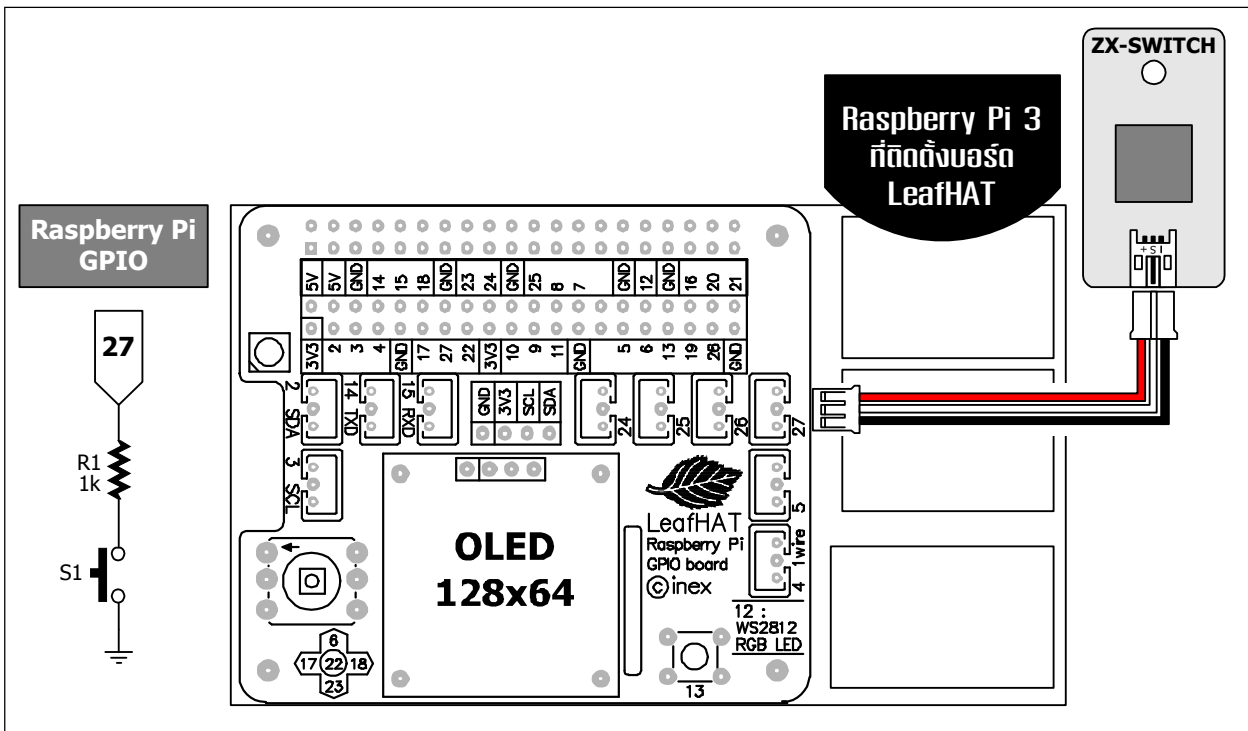
วิธีที่ 1 เป็นการอ่านค่าสถานะ ณ เวลานั้นของขา GPIO เมื่ออ่านค่าได้แล้ว จึงกระทำคำสั่งต่อไป

วิธีที่ 2 เป็นการหยุดรอจนกว่าสัญญาณจะตรงกับที่กำหนดไว้ จึงกระทำคำสั่งต่อไป

(ก) ใช้งานพอร์ตอินพุตดิจิตอลแบบซิงโครนัสด้วยวิธีที่ 1

ในวิธีนี้จะเป็นการอ่านค่าสถานะในปัจจุบันของขา GPIO เมื่ออ่านค่าได้แล้ว จึงกระทำคำสั่งต่อไป ต้องใช้คำสั่งต่อไปนี้ในการกำหนดลักษณะการทำงาน

```
state = GPIO.input(ขาพอร์ต)
```



รูปที่ 2-7 การทดลองใช้งานขาพอร์ต 27 ของ Raspberry Pi 3 เป็นพอร์ตอินพุต โดยใช้บอร์ด LeafHAT และ ZX-SWITCH01 ในการทดลอง

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

input_pin = 27

GPIO.setup(input_pin, GPIO.IN)

while(1):
    state = GPIO.input(input_pin)
    print (state)
    time.sleep(0.5)
```

โปรแกรมที่ 2-4 โค้ดภาษา Python ไฟล์ SynchronousInput1.py สำหรับอ่านค่าสถานะของขาพอร์ต 27 ของบอร์ด Raspberry Pi 3 เมื่อทำงานเป็นขาพอร์ตอินพุต

ตัวอย่างที่ 2-2

(1) ต่อดวงจรมตามรูปที่ 2-7

(2) เข้าสู่โปรแกรม Geany แล้วเขียนโปรแกรมที่ 2-4

(3) บันทึกไฟล์ SynchronousInput1.py ทำการคอมไพล์และเ็กซิกคิวต์ เพื่อทดสอบการทำงาน

โปรแกรมก็จะคอยอ่านค่าสถานะของขา GPIO27 ว่า มีสถานะเป็นอย่างไร ถ้ามีสถานะเป็น High จะแสดงค่าออกมาเป็นเลข 1 และถ้าเป็น Low ก็แสดงเป็น 0 โดยจะอ่านค่าทุกๆ 0.5 วินาที แล้วแสดงบนหน้าต่างเทอร์มินอล ทดลองง่ายๆ ด้วยการกดและปล่อยสวิตช์ เมื่อขาพอร์ตต่อไฟเลี้ยง +3.3V จะอ่านค่าสถานะเป็น “1” หากต่อลงกราวด์จะแสดงสถานะ “0”

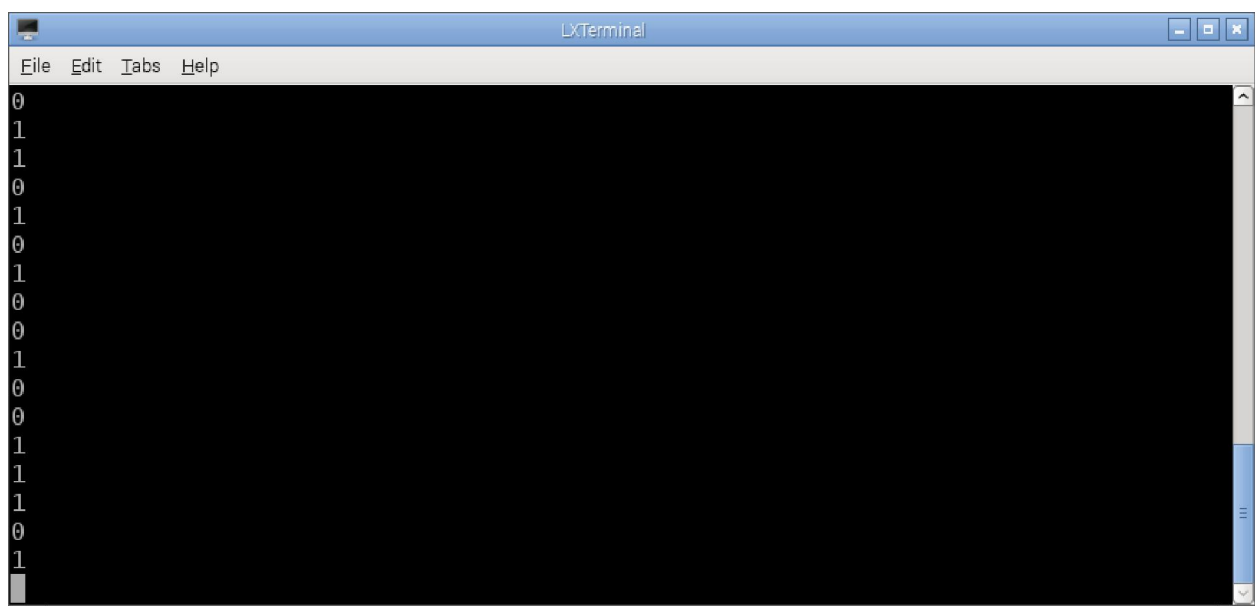
(4) การแสดงผลเป็นเลข 0 และ 1 อาจจะทำให้ดูเข้าใจยาก ดังนั้นจึงควรปรับเปลี่ยนการแสดงผลค่าให้เหมาะสมมากขึ้นด้วยการแสดงเป็นข้อความบอกว่า ปุ่มถูกกดหรือไม่ถูกกด

(5) เขียนโปรแกรมทดสอบเพิ่มเติมดังโปรแกรมที่ 2-5

(6) บันทึกไฟล์ชื่อ SynchronousInput2.py ทำการคอมไพล์และเ็กซิกคิวต์ เพื่อทดสอบการทำงาน

(7) ทดลองกดสวิตช์แล้วปล่อยสลับกัน สังเกตผลการทำงานที่หน้าต่างเทอร์มินอล

เมื่อกดปุ่มหรือสวิตช์ไม่ถูกกดจะแสดงข้อความว่า *Button OK* และเมื่อกดปุ่มหรือสวิตช์ถูกกดจะแสดงข้อความว่า *Button Pressed* ที่จอแสดงผล ดังรูปที่ 2-9



รูปที่ 2-8 ผลการทำงานของโปรแกรมที่ 2-4 เป็นการแสดงสถานะลอจิกของขา 27 ของพอร์ต GPIO ของบอร์ด Raspberry Pi 3 ที่ใช้งานเป็นขาพอร์ตอินพุต

```
import RPi.GPIO as GPIO
import time

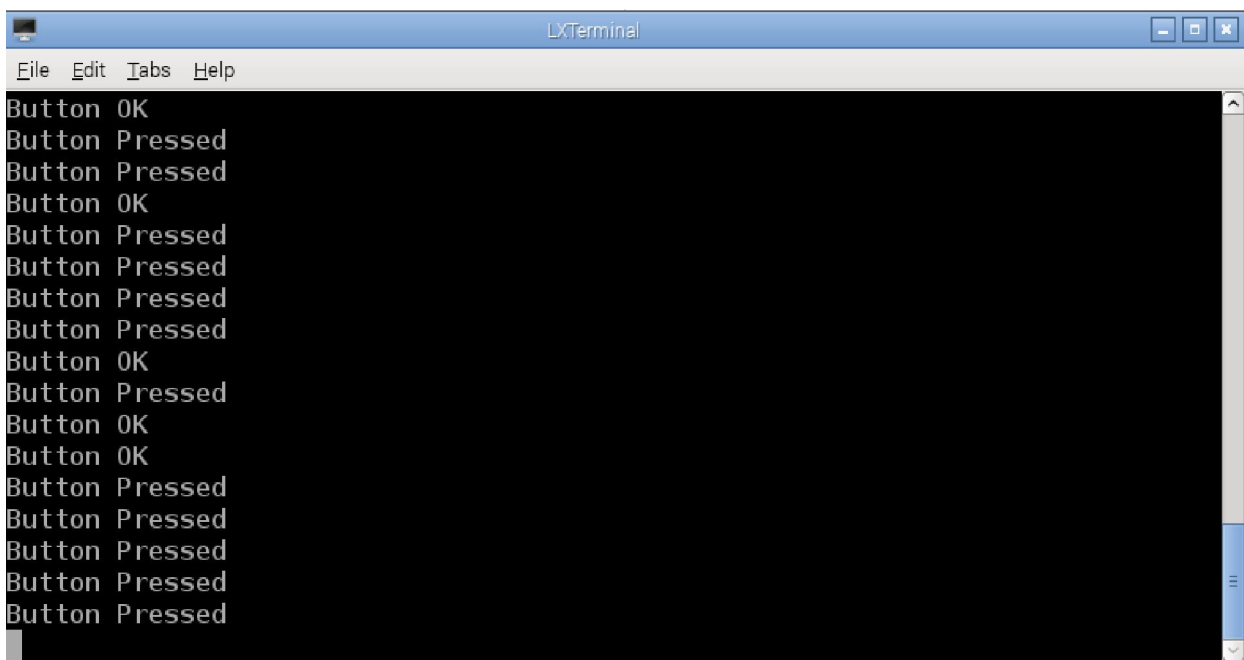
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

input_pin = 27

GPIO.setup(input_pin, GPIO.IN)

while(1):
    state = GPIO.input(input_pin)
    if(state == GPIO.HIGH):
        print ("Button Pressed")
    elif(state == GPIO.LOW):
        print ("Button OK")
    time.sleep(0.5)
```

โปรแกรมที่ 2-5 โค้ดภาษา Python ไฟล์ SynchronousInput2.py สำหรับใช้งานพอร์ต GPIO ของบอร์ด Raspberry Pi 3 เป็นอินพุตทำงานแบบซิงโครนัสในลักษณะที่เมื่ออ่านค่าได้แล้ว ให้กระทำคำสั่งต่อไป



รูปที่ 2-9 ผลการทำงานของโปรแกรมที่ 2-5 เป็นการแสดงสถานะลอจิกของขา GPIO27 ของพอร์ต GPIO ของบอร์ด Raspberry Pi 3 ที่ใช้งานเป็นขาพอร์ตอินพุตเมื่อมีการกดและปล่อยสวิตช์

(ข) ใช้งานพอร์ตอินพุตดิจิทัลแบบซิงโครนัสด้วยวิธีที่ 2

เป็นการรอกกว่าค่าสัญญาณจะตรงกับที่กำหนดไว้ จึงกระทำคำสั่งต่อไป ให้ใช้คำสั่งต่อไปนี้

state = GPIO.wait_for_edge (ขาพอร์ต, ขอบขาของสัญญาณที่ต้องการอ่านค่า)
เช่น

```
state = GPIO.wait_for_edge(20, GPIO.FALLING)
```

สำหรับตรวจจับขอบขาลงของสัญญาณ

```
state = GPIO.wait_for_edge(20, GPIO.RISING)
```

สำหรับตรวจจับขอบขาขึ้นของสัญญาณ

ตัวอย่างที่ 2-3

(1) ใช้วงจรจากรูปที่ 2-7 ในการทดลอง เข้าสู่โปรแกรม Geany แล้วเขียนโปรแกรมที่ 2-6

(2) บันทึกไฟล์ SynchronousInputEdge.py ทำการคอมไพล์และรันไฟล์สคริปต์ โดยไปที่เมนู

Bulid > Execute หรือรันผ่านเทอร์มินอลด้วยคำสั่ง **sudo python3 SynchronousInputEdge.py**

โปรแกรมนี้เป็นการรอสัญญาณขอบขาขึ้นที่ขา GPIO27 ของ Raspberry Pi 3 โดยจะหยุดรอที่คำสั่ง GPIO.wait_for_edge เมื่อกดสวิตช์ สถานะที่ขาพอร์ตจะเป็น “1” หรือเกิดสัญญาณขอบขาขึ้น ทำให้เงื่อนไขเป็นจริง แสดงสถานะทางจอแสดงผล แล้วรอ 0.2 วินาทีก่อนวนกลับไปอ่านค่าใหม่

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

input_pin = 27
GPIO.setup(input_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
while(1):
    GPIO.wait_for_edge(input_pin, GPIO.RISING)
    print ("Button Pressed")
    time.sleep(0.2)
```

คำอธิบายโปรแกรมเพิ่มเติม

ที่บรรทัด GPIO.setup(input_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) เป็นการกำหนดให้ขาพอร์ตที่เลือกในที่นี้คือขา GPIO27 จากคำสั่ง input_pin = 27 เป็นอินพุต และมีการกำหนดสถานะลอจิกตั้งต้นของขาพอร์ตเป็นแบบพูลดาวน์หรือเป็น “0”

คำสั่งตรวจสอบสถานะของขาพอร์ตคือ GPIO.wait_for_edge(input_pin, GPIO.RISING) หากพบสัญญาณขอบขาขึ้น (Rising edge) ก็จะทำงานในคำสั่งถัดไป ซึ่งก็คือ คำสั่ง print ("Button Pressed") เพื่อแสดงสถานะของการกดสวิตช์ไปยังจอแสดงผล

โปรแกรมที่ 2-6 โค้ดภาษา Python ไฟล์ SynchronousInputEdge.py สำหรับใช้งาน GPIO ของ Raspberry Pi 3 เป็นอินพุตทำงานแบบซิงโครนัสในลักษณะรอกกว่าสถานะของขาพอร์ตตรงกับที่กำหนดไว้จึงกระทำคำสั่งต่อไป

2.7.2 พอร์ตอินพุตดิจิทัลแบบอะซิงโครนัส

การใช้งานพอร์ตอินพุตดิจิทัลแบบนี้เป็นการใช้อีเวนต์ (Event) หรือเหตุการณ์เข้ามาช่วย โดยไม่จำเป็นต้องคอยเขียนคำสั่งวนตรวจสอบค่าตลอดเวลาเพียงกำหนดอีเวนต์หรือเหตุการณ์ที่จะเกิดขึ้น และเมื่อเกิดอีเวนต์ตามที่กำหนดไว้ โปรแกรมจะทำงานตามชุดคำสั่งในส่วนนั้นให้ทันที

ตัวอย่างที่ 2-4

- (1) ยังคงใช้วงจรในรูปที่ 2-7 ในการทดลอง เข้าสู่โปรแกรม Geany แล้วเขียนโปรแกรมที่ 2-7
- (2) บันทึกไฟล์ `AsynchronousInput.py` ทำการคอมไพล์และเอ็กซิคิวต์ เพื่อทดสอบการทำงาน

ผลการทำงานจะเหมือนกับโปรแกรมที่ 2-6 หากแต่ลักษณะของโปรแกรมแตกต่างกัน โดยในโปรแกรมที่ 2-7 นี้กำหนดให้ขาพอร์ตอินพุตทำงานในแบบอะซิงโครนัส

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

input_pin = 27

def button_pressed(channel):
    print ("Button Pressed")

GPIO.setup(input_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.add_event_detect(input_pin, GPIO.RISING, callback=button_pressed,
bouncetime=300)
while(1):
    pass
```

คำอธิบายโปรแกรมเพิ่มเติม

ในโปรแกรมนี้นี้สร้างฟังก์ชันที่ชื่อว่า `button_pressed` เตรียมไว้ จากนั้นกำหนดขา GPIO27 ให้เป็นพอร์ตอินพุตดิจิทัล และกำหนดให้มีการต่อพูลดาวน์ไว้ จากนั้นกำหนดเงื่อนไขให้กับขา GPIO27 ว่า ถ้ามีสัญญาณที่เป็นขอบขาขึ้น (Rising) ก็จะเรียกให้ฟังก์ชัน `button_pressed` ทำงาน และกำหนดระยะเวลาสำหรับดีเบายซ์ (debounce) หรือการแก้สัญญาณรบกวนจากการกดสวิตช์ไว้ที่ 300 มิลลิวินาที

เมื่อกดสวิตช์ จะทำให้มีสัญญาณขอบขาขึ้นที่ขา GPIO27 ฟังก์ชัน `button_pressed` ถูกเรียกให้ทำงาน จะแสดงข้อความบนหน้าจอว่า สวิตช์ที่ต่อกับขา GPIO27 ถูกกด ให้สังเกตที่ตัวแปร `channel` ที่อยู่ในฟังก์ชัน `button_pressed` ตัวแปรนี้ก็คือ ขา GPIO ตัวที่ใช้งานนั่นเอง ในกรณีนี้คือขา GPIO27

โปรแกรมที่ 2-7 โค้ดภาษา Python ไฟล์ `AsynchronousInput.py` สำหรับใช้งาน GPIO ของบอร์ด Raspberry Pi 3 เป็นอินพุตทำงานแบบอะซิงโครนัส เมื่อมีเหตุการณ์หรืออีเวนต์ (Event) ตามที่กำหนดไว้เกิดขึ้น โปรแกรมจะทำงานตามชุดคำสั่งที่กำหนดทันที

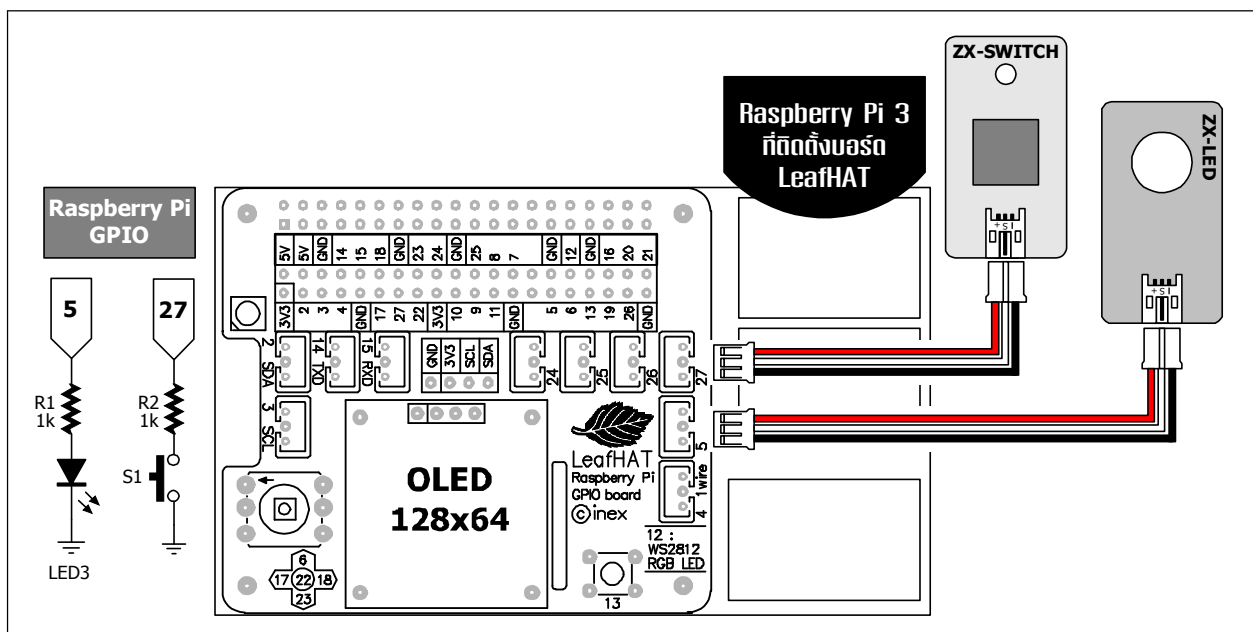
2.8 ตัวอย่างการใช้งานพอร์ตอินพุตเอาต์พุต

2.8.1 สวิตช์ควบคุม LED

ในตัวอย่างนี้เป็นการกำหนดให้ Raspberry Pi 3 อ่านค่าจากขา GPIO27 ที่ต่อกับสวิตช์เพื่อนำค่ามาขับ LED ที่ต่อกับขา GPIO5

ตัวอย่างที่ 2-5

- (1) ต่อดังตามรูปที่ 2-10
- (2) สร้างไฟล์สคริปต์ด้วย Geany หรือ nano เทกซ์เอดิเตอร์



รูปที่ 2-10 วงจรทดสอบการทำงานของพอร์ต GPIO เบื้องต้นในลักษณะทำงานร่วมกันทั้งอินพุตและเอาต์พุต

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.OUT)
GPIO.setup(27, GPIO.IN)

while (1):
    state=GPIO.input(27)
    GPIO.output(5, state)
    print("SW=", state)
```

โปรแกรมที่ 2-8 โค้ดภาษา Python ไฟล์ SwitchLED.py สำหรับใช้งาน GPIO ของบอร์ด Raspberry Pi 3 ทั้งอินพุตและเอาต์พุตให้ทำงานร่วมกัน โดยอ่านค่าสถานะของสวิตช์จากพอร์ตอินพุตมาควบคุมการทำงานของ LED ที่ต่อกับขาพอร์ตเอาต์พุต

(3) พิมพ์คำสั่งตามโปรแกรมที่ 2-8

(4) ทำการบันทึกไฟล์ชื่อ SwitchLED.py

(5) สั่งให้ไฟล์สคริปต์ทำงาน โดยไปที่เมนู Bulid > Execute หรือรันไฟล์สคริปต์ด้วยเทอร์มินอลโดยใช้คำสั่ง **sudo python3 SwitchLED.py**

(6) สังเกตการทำงานของ LED และหน้าจอแสดงผลเมื่อไม่ได้กดสวิตช์ และหลังจากกดสวิตช์

เมื่อยังไม่ได้กดสวิตช์ ที่ขา GPIO27 จะมีค่าเป็น 1 และส่งผลให้ ขา GPIO5 ที่กำหนดไว้เป็น เอาต์พุต มีสถานะเป็น 1 ทำให้ LED ที่ต่อกับขา GPIO5 สว่าง และเมื่อกดสวิตช์ สถานะที่ขา GPIO27 จะมีค่าเป็น 0 ส่งผลให้ LED ที่ต่อกับขา GPIO5 ดับ และที่จอแสดงผลจะแสดงข้อความดังนี้

SW=1 หมายความว่าไม่ได้กดสวิตช์

SW=0 หมายความว่า กดสวิตช์

2.8.2 สวิตช์ควบคุมไฟวง LED

ในตัวอย่างนี้เป็นการกำหนดให้ Raspberry Pi 3 อ่านค่าจากขา 27 ที่ต่อกับสวิตช์เพื่อนำสถานะ มาควบคุมการทำงานของไฟวง LED ที่ต่อกับขา 24, 25 และ 26

(1) ต่อวงจรตามรูปที่ 2-11 จากนั้นสร้างไฟล์สคริปต์ด้วย Geany หรือ Nano เท็กซ์เอดิเตอร์

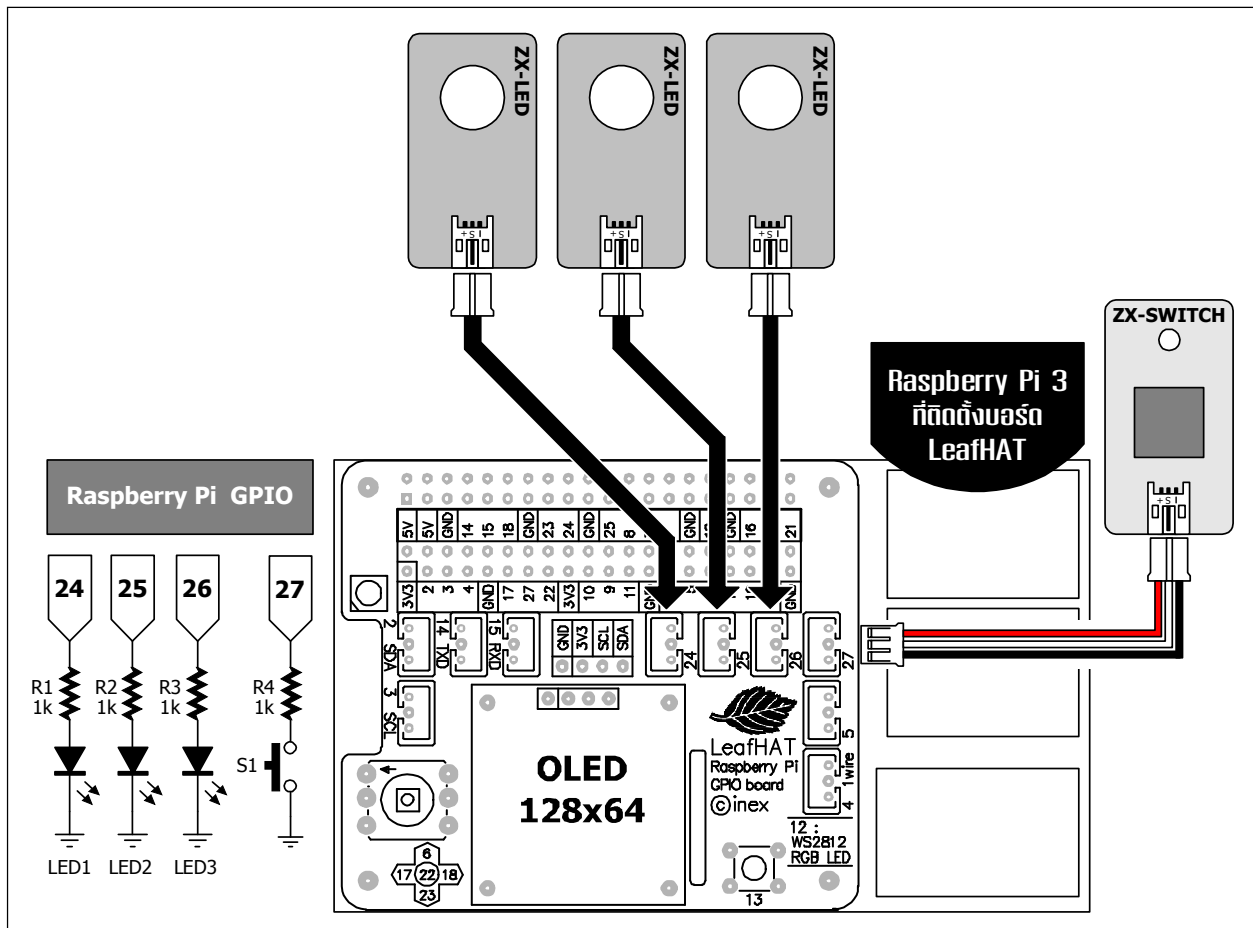
(2) พิมพ์คำสั่งตามโปรแกรมที่ 2-9

(3) ทำการบันทึกไฟล์ชื่อ SwitchLED3Running.py

(4) สั่งให้ไฟล์สคริปต์ทำงาน โดยไปที่เมนู **Bulid > Execute** ในหน้าต่างของ Geany หรือรันไฟล์สคริปต์ด้วยเทอร์มินอล โดยใช้คำสั่ง **sudo python3 SwitchLED3Running.py**

(5) สังเกตการทำงานของ LED และจอแสดงผลเมื่อไม่ได้กดสวิตช์ และหลังจากกดสวิตช์ค้างไว้

เมื่อกดสวิตช์ วงจรไฟวง LED ยังไม่ทำงาน ทันทีที่ปล่อยสวิตช์ วงจรไฟวง LED จะเริ่มทำงาน โดย LED ที่ต่อกับขา GPIO24 จะติดเป็นดวงแรกนาน 0.5 วินาทีแล้วดับลง LED ที่ต่อกับขา GPIO25 จะติดขึ้นมาแทน ตามด้วย LED ที่ต่อกับขา GPIO 26 แล้วหยุดทำงาน เพื่อรอให้กดและปล่อยสวิตช์ที่ต่อกับขา GPIO27 อีกครั้ง



รูปที่ 2-11 วงจรสวิตช์ควบคุมไฟวิ่ง 3 ดวง เป็นวงจรทดสอบการทำงานของพอร์ต GPIO เบื้องต้นในลักษณะทำงานร่วมกันทั้งอินพุตและเอาต์พุต

```

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
my_pin=[24,25,26]
count_pin=0;
count_state=1;
input_pin=27
GPIO.setup(input_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
for pin in my_pin:
    GPIO.setup(pin,GPIO.OUT)
    count_pin+=1
while (1):
    GPIO.wait_for_edge(input_pin, GPIO.RISING)
    count_state+=1
    state=count_state%2
    print("State=",state)
    if state==0:
        for i in range(count_pin):
            GPIO.output(my_pin[i],1)
            time.sleep(0.5)
            GPIO.output(my_pin[i],0)

    elif state==1:
        for i in range(count_pin):
            i=count_pin-1-i
            GPIO.output(my_pin[i],1)
            time.sleep(0.5)
            GPIO.output(my_pin[i],0)

```

คำอธิบายโปรแกรมเพิ่มเติม

ในโปรแกรมนี้ ใช้สัญญาณขอบขาขึ้นในขณะที่ปล่อยสวิตช์เป็นตัวกำหนดการทำงานของ LED ด้วยคำสั่ง `GPIO.wait_for_edge(input_pin, GPIO.RISING)` ซึ่งจะรอจนกว่าจะมีสัญญาณขอบขาขึ้นปรากฏขึ้นมา จากนั้นเพิ่ม `count_state` 1 ค่า แล้วหารด้วย 2 โดยใช้คำสั่ง `mod (%)` เพื่อหาเศษ ผลลัพธ์ที่ได้คือ 0 และ 1

นั่นหมายความว่า ถ้า `count_state` เป็นจำนวนคู่ ผลลัพธ์จะเป็น “0” และถ้า `count_state` เป็นจำนวนคี่ ผลลัพธ์จะเป็น “1” จากนั้นนำผลลัพธ์มาเป็นเงื่อนไขในการตรวจสอบต่อไป

โปรแกรมที่ 2-9 โค้ดภาษา Python ไฟล์ SwitchLED3Running.py สำหรับใช้งานพอร์ต GPIO ของบอร์ด Raspberry Pi 3 ทั้งอินพุตและเอาต์พุตให้ทำงานร่วมกัน โดยอ่านค่าสถานะของสวิตช์จากพอร์ตอินพุต มาควบคุมการทำงานของไฟวิ่ง LED 3 ดวงที่ต่อกับขาพอร์ตเอาต์พุต 3 ขา

2.9 การสร้างสัญญาณ PWM สำหรับ GPIO ของ Raspberry Pi 3

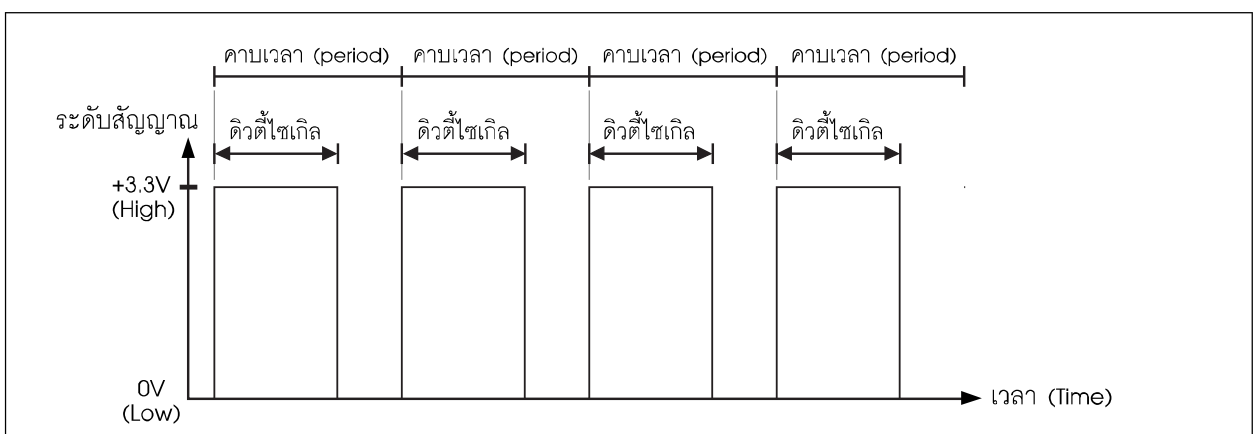
สัญญาณ PWM (Pulse-width Modulation) หรือสัญญาณมอดูเลชันทางความกว้างของพัลส์ เป็นสัญญาณดิจิทัลอีกแบบหนึ่งที่เป็นที่นิยมใช้ในการควบคุมความเร็วมอเตอร์ และควบคุมความสว่างของ LED โดยใช้ช่วงเวลาที่สัญญาณ PWM เป็นลอจิกสูงในการควบคุม โดยแรงดันที่ใช้ในการควบคุมความเร็วของมอเตอร์หรือความสว่างของ LED จะขึ้นกับความกว้างของสัญญาณพัลส์บวกของสัญญาณ PWM ดังนั้นหากพัลส์บวกมีความกว้างมาก แรงดันไฟตรงที่ได้จะสูง

2.9.1 แนะนำสัญญาณ PWM

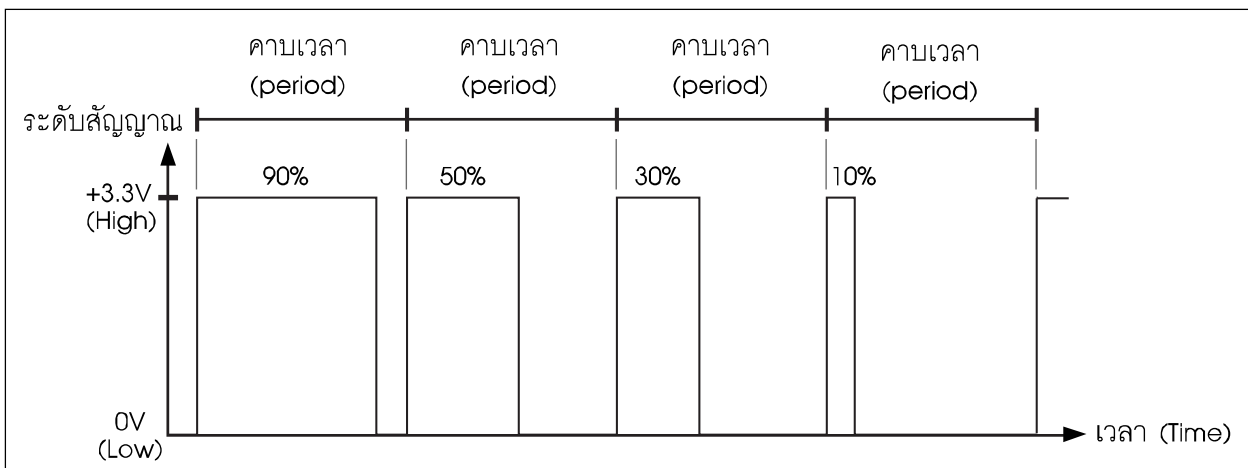
ในรูปที่ 2-12 แสดงตัวอย่างสัญญาณ PWM ซึ่งมีลักษณะเป็นสัญญาณพัลส์ที่มีความกว้างค่าหนึ่งที่สามารถกำหนดได้ โดยมีค่าของคาบเวลาที่คงที่ เนื่องจากสัญญาณ PWM มีทั้งช่วงเวลาที่ เป็นลอจิกสูงและต่ำสลับกัน มากน้อยขึ้นกับการกำหนดความกว้างของพัลส์ ซึ่งเรียกสัดส่วนของสัญญาณพัลส์บวก (ช่วงที่เป็นลอจิกสูง) กับคาบเวลาทั้งหมดของสัญญาณพัลส์ว่า **ดิวตี้ไซเคิล (duty cycle)** โดยคิดค่าดิวตี้ไซเคิลเป็นเปอร์เซ็นต์ของค่าความกว้างพัลส์ ทั้งหมด

เมื่อสัญญาณมีลักษณะเป็นพัลส์ดังรูปที่ 2-12 ก็จะทำให้แรงดันไฟตรงของสัญญาณ PWM นี้มี ถูกเฉลี่ยตามอัตราส่วนระหว่างความกว้างพัลส์กับคาบเวลาของสัญญาณหรือดิวตี้ไซเคิล หากค่าของ ดิวตี้ไซเคิลมาก แรงดันไฟตรงเฉลี่ยของสัญญาณ PWM จะสูง ในทางตรงข้าม หากค่าดิวตี้ไซเคิลต่ำ แรงดันเฉลี่ยก็จะต่ำ

ในรูปที่ 2-13 แสดงให้เห็นถึงสัญญาณ PWM ที่มีค่าดิวตี้ไซเคิลต่างๆ กัน จะเห็นว่า คาบเวลา ของสัญญาณจะคงที่ แต่ความกว้างของสัญญาณพัลส์บวกแตกต่างกัน ตัวอย่างจากรูปที่ 2-13 พัลส์ ลูกซ้ายสุดมีค่าดิวตี้ไซเคิล 90% หมายถึง ความกว้างของพัลส์ช่วงบวกมีความกว้างเป็น 90% ของความกว้างทั้งหมด ดังนั้นแรงดันเฉลี่ยที่ได้เท่ากับ $(90 \times 3.3) / 100 = 2.97V$ (ที่ไฟเลี้ยงวงจร +3.3V)



รูปที่ 2-12 ตัวอย่างของสัญญาณ PWM



รูปที่ 2-13 แสดงตัวอย่างสัญญาณ PWM ที่มีค่าดีวตี้ไซเคิลต่าง ๆ กัน

และถ้าหากสัญญาณ PWM มีดีวตี้ไซเคิล 50% แรงดันเฉลี่ยจะเท่ากับ 1.65V ลดลงเหลือ 0.99V ที่ดีวตี้ไซเคิล 30% และ 0.33V ที่ดีวตี้ไซเคิล 10%

ผู้พัฒนาสามารถปรับค่าดีวตี้ไซเคิล, คาบเวลา และความถี่ของสัญญาณ PWM ได้ตามจุดประสงค์ในการใช้งาน โดยมีการคำนวณเบื้องต้นดังนี้

ตัวอย่างที่ 2-6

สัญญาณ PWM มีคาบเวลา 20 มิลลิวินาที มีความกว้างพัลส์ 5 มิลลิวินาที จะมีดีวตี้ไซเคิลเท่ากับ $5 / 20 \times 100\% = 25\%$

ตัวอย่างที่ 2-7

สัญญาณ PWM ความถี่ 4kHz และมีดีวตี้ไซเคิล 18% คำนวณหาคาบเวลาและความกว้างของพัลส์ได้ดังนี้

$$\text{คาบเวลา} = 1/4000\text{Hz} = 0.25 \text{ มิลลิวินาที หรือ } 250 \text{ ไมโครวินาที}$$

$$\text{ความกว้างพัลส์} = 250 \times 18\% / 100\% = 45 \text{ ไมโครวินาที}$$

2.9.2 Raspberry Pi 3 กับการสร้างสัญญาณ PWM

ในการสร้างสัญญาณ PWM บน Raspberry Pi 3 นั้น ทำได้ง่ายๆ ด้วยคำสั่ง PWM ที่มีอยู่แล้วใน RPi.GPIO สิ่งที่ต้องทำคือ กำหนดขาที่ต้องการขับสัญญาณ PWM, ความถี่ และค่าดิวตี้ไซเคิล

ในชิพ Broadcom BCM2837 ของ Raspberry Pi 3 มีฮาร์ดแวร์สร้างสัญญาณ PWM อยู่แล้ว โดยกำหนดให้ขา GPIO12 กับ 18 เป็นขาเอาต์พุต PWM0 และขา GPIO13 กับ GPIO19 เป็นเอาต์พุต PWM1 หากมีการเลือกใช้งาน แต่ในไลบรารี RPi.GPIO ไม่ได้เรียกใช้ โดยเลือกให้กำเนิดสัญญาณ PWM ด้วยกระบวนการทางซอฟต์แวร์แทน หรือเรียกว่า **PWM ทางซอฟต์แวร์ (Software PWM)** ซึ่งมีข้อดีคือทำให้ขาพอร์ต GPIO ทุกขาสร้างสัญญาณ PWM ได้

การกำหนดให้ขาพอร์ต GPIO ของ Raspberry Pi 3 เป็นขาเอาต์พุต PWM จะต้องเรียกใช้โมดูล RPi.GPIO และกำหนดให้ขาพอร์ตที่ต้องการใช้งานเป็นเอาต์พุตก่อน ด้วยคำสั่ง

```
GPIO.setup(pin, GPIO.OUT)
```

ส่วนคำสั่งสำหรับการสร้างสัญญาณ PWM ประกอบด้วย

2.9.2.1 คำสั่งเลือกขาพอร์ตและความถี่

รูปแบบคำสั่ง

```
p = GPIO.PWM(pin, frequency)
```

พารามิเตอร์

p คือ ตัวแปรใดๆ

pin คือ ขาพอร์ต GPIO

frequency คือ ความถี่ของสัญญาณ PWM ในหน่วย Hz

ตัวอย่างที่ 2-8

```
p = GPIO.PWM(25, 50) # ใช้ขาพอร์ต GPIO25 ขับสัญญาณ PWM ด้วยความถี่ 50Hz
```

2.9.2.2 คำสั่งเริ่มต้นการสร้างสัญญาณ PWM ด้วยการกำหนดค่าดิวตี้ไซเคิล

รูปแบบคำสั่ง

```
p.start(dutycycle)
```

พารามิเตอร์

p คือ ตัวแปรใดๆ

dutycycle คือ ค่าดิวตี้ไซเคิล มีค่า 0.0 ถึง 100.0

ตัวอย่างที่ 2-9

```
p.start = (0) # ไม่มีสัญญาณออกจากขาพอร์ต เนื่องจากค่าดิวตี้ไซเคิล = 0%
```

```
p.start = (50) # กำหนดสัญญาณ PWM ที่มีค่าดิวตี้ไซเคิล 50%
```


2.9.2.3 คำสั่งเปลี่ยนค่าความถี่

รูปแบบคำสั่ง

p.ChangeFrequency (frequency)

พารามิเตอร์

p คือ ตัวแปรใดๆ

frequency คือ ความถี่ของสัญญาณ PWM ในหน่วย Hz

ตัวอย่างที่ 2-10

```
p.ChangeFrequency(100)
```

เปลี่ยนความถี่ของสัญญาณ PWM เป็น 100Hz

2.9.2.4 คำสั่งเปลี่ยนค่าดิวตี้ไซเคิล

รูปแบบคำสั่ง

p.ChangeDutyCycle (duty_cycle)

พารามิเตอร์

p คือ ตัวแปรใดๆ

duty_cycle คือ ค่าดิวตี้ไซเคิล มีค่า 0.0 ถึง 100.0

ตัวอย่างที่ 2-11

```
p.ChangeDutyCycle(90)
```

เปลี่ยนค่าดิวตี้ไซเคิลเป็น 90%

2.9.2.5 คำสั่งหยุดการสร้างสัญญาณ PWM

รูปแบบคำสั่ง

p.stop()

พารามิเตอร์

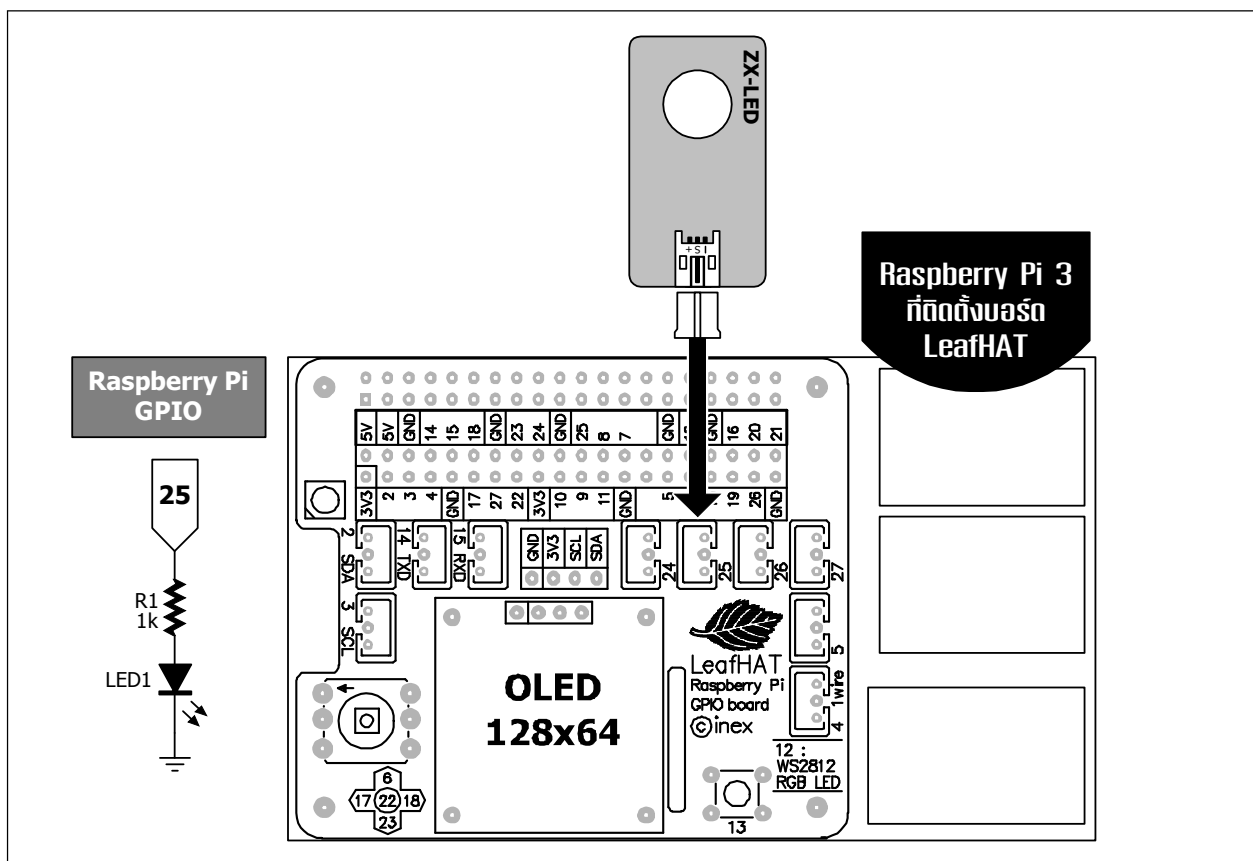
p คือ ตัวแปรใดๆ

2.9.3 ตัวอย่างการสร้างสัญญาณ PWM ของ Raspberry Pi 3

2.9.3.1 ขับ LED กระพริบด้วยสัญญาณ PWM

- (1) ต่อยังจตามรูปที่ 2-14 โดยต่อแผงวงจร ZX-LED เข้าที่ขา GPIO25 ผ่านบอร์ด LeafHAT
- (2) สร้างไฟล์สคริปต์ด้วย Geanny หรือ nano เท็กซ์เอดิเตอร์ พิมพ์คำสั่งตามโปรแกรมที่ 2-10
- (3) บันทึกไฟล์ชื่อ PWMblink.py แล้วทำการคอมไพล์และเอ็กซิกิวต์เพื่อทดสอบการทำงาน

โปรแกรมนี้จะสร้างสัญญาณ PWM ความถี่ 1Hz มีค่าคิวตี้ไซเคิล 50% ส่งออกไปยังขา GPIO25 โดยสัญญาณ PWM ที่ขับออกมานี้จะทำให้ LED ที่ต่อกับขา GPIO25 กระพริบทุกๆ 0.5 วินาที



รูปที่ 2-14 วงจรทดลองสร้างสัญญาณ PWM ของบอร์ด Raspberry Pi 3 ผ่านทางขา GPIO25

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(25,GPIO.OUT)

blink = GPIO.PWM(25, 1)    # Generate PWM at GPIO25 with 1Hz frequency
print ('Start Blinking')
blink.start(50)            # Start PWM by 50% duty cycle
while True:
    pass
```

โปรแกรมที่ 2-10 โค้ดภาษา Python สำหรับใช้งานพอร์ต GPIO ของบอร์ด Raspberry Pi 3 สร้างสัญญาณ PWM

2.9.3.2 ปรับความสว่างของ LED ด้วยสัญญาณ PWM

(1) ยังคงใช้วงจรตามรูปที่ 2-14 ในการทดลอง

(2) เข้าสู่โปรแกรม Geany หรือ nano เท็กซ์เอดิเตอร์ แล้วเขียนโปรแกรมที่ 2-11

(3) บันทึกไฟล์ชื่อ **PWMLED.py** สั่งให้ไฟล์สคริปต์ทำงาน โดยไปที่เมนู **Bulid > Execute** หรือรันไฟล์สคริปต์ด้วยการเปิดหน้าต่าง Terminal แล้วใช้คำสั่ง **sudo python3 PWMLED.py**

โปรแกรมนี้จะส่งสัญญาณ *PWM* ออกไปทางขา *GPIO25* เพื่อขับให้ *LED* สว่างเพิ่มขึ้นเรื่อยๆ จนสว่างที่สุดที่ค่าคิวดิโไซเกิด 100% และเมื่อสว่างเต็มที่แล้ว สัญญาณ *PWM* จะลดค่าคิวดิโไซเกิดลง ทำให้ *LED* ค่อยหรี่ๆ ลงจนดับ แล้ววนทำงานในลักษณะนี้ตลอดเวลา หากต้องการให้หยุดทำงาน ให้กด *Ctrl* และ *C* (กดคีย์ *Ctrl* ค้างไว้ แล้วกดคีย์ *C* ตาม)

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)

blink = GPIO.PWM(25, 1000)

multiply = 1
duty = 0

while True:
    blink.start(duty)
    time.sleep(0.01)

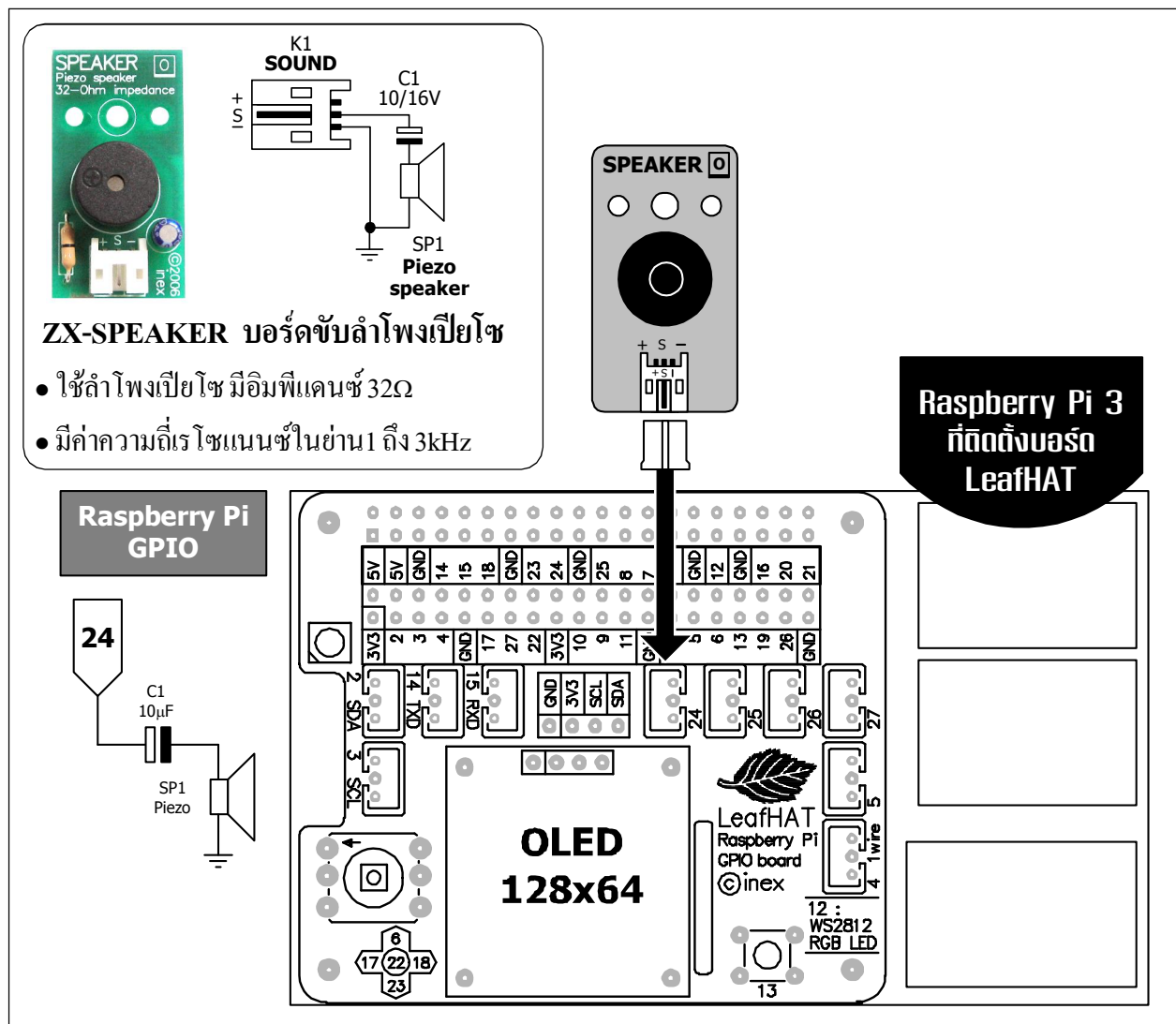
    if duty >= 100:
        multiply = -1
    if duty <= 0:
        multiply = 1
    duty += multiply
    print (duty)
```

โปรแกรมที่ 2-11 โค้ดภาษา Python ไฟล์ **PWMLED.py** สำหรับใช้งาน **GPIO** ของบอร์ด **Raspberry Pi 3** เพื่อสร้างสัญญาณ **PWM** สำหรับขับและควบคุมความสว่างของ **LED**

2.9.3.3 สร้างเสียงเตือนด้วย PWM

- (1) ต่อดังตามรูปที่ 2-15
- (2) เข้าสู่โปรแกรม Geany หรือ nano เท็กซ์เอดิเตอร์ แล้วเขียนโปรแกรมที่ 2-12
- (3) บันทึกไฟล์ชื่อ PWMpiezo.py
- (4) สั่งให้ไฟล์สคริปต์ทำงาน โดยไปที่เมนู **Bulid > Execute** หรือรันไฟล์สคริปต์ด้วยการเปิดหน้าต่าง Terminal แล้วใช้คำสั่ง `sudo python3 PWMpiezo.py`

จะได้ยินเสียงดังที่ลำโพงเปียโซซึ่งต่อกับขา GPIO24 ทุกๆ 0.7 วินาที และดังอยู่นาน 0.3 วินาที โดยสัญญาณ PWM ที่สร้างขึ้นนี้มีค่าที่ 500Hz ส่วนระยะเวลาของการกำเนิดเสียง ใช้พารามิเตอร์ `microsecond` ที่มีค่า 0 ถึง 999,999 ในการกำหนดเวลาของการกำหนดเสียง



รูปที่ 2-15 วงจรทดลองสร้างสัญญาณ PWM จากพอร์ต GPIO ของ Raspberry Pi 3 เพื่อขับออกลำโพงเปียโซ

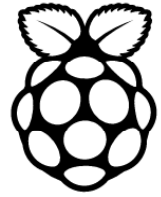
```
import RPi.GPIO as GPIO
import time
from datetime import datetime
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.OUT)
blink = GPIO.PWM(24, 500)
blink.start(0)
while True:
    da=datetime.now()
    microsec=da.microsecond
    if microsec > 700000:
        blink.ChangeDutyCycle(50)
    else:
        blink.ChangeDutyCycle(0)
```

โปรแกรมที่ 2-12 โค้ดภาษา Python ไฟล์ PWMpiezo.py สำหรับใช้งานพอร์ต GPIO ของบอร์ด Raspberry Pi 3 เพื่อสร้างสัญญาณ PWM สำหรับขับเสียงออกทางลำโพงเปียโซ

โดยค่าเวลานี้มาจากคำสั่ง `da=datetime.now()` เมื่อค่าเวลาของพารามิเตอร์ `microsecond` น้อยกว่า 700,000 จะทำให้คำสั่ง `blink.ChangeDutyCycle(0)` ได้ค่าของคิวตี้ไซเคิลเท่ากับ 0 จึงไม่มีการขับสัญญาณเสียงเกิดขึ้น เมื่อ ค่าเวลาของพารามิเตอร์ `microsecond` มากกว่า 700,000 คำสั่ง `blink.ChangeDutyCycle(50)` ทำงานเปลี่ยนค่าของคิวตี้ไซเคิลเท่ากับ 50% จึงเกิดการขับสัญญาณเสียงขึ้น และมีเสียงออกลำโพงเปียโซอย่างต่อเนื่องจนกระทั่งค่าเวลาของพารามิเตอร์ `microsecond` กลับเป็น 0 ดังนั้นจึงมีเสียงสัญญาณดังขึ้นนานประมาณ 300,000 ไมโครวินาทีหรือ 0.3 วินาที และเงียบ 0.7 วินาที



บทที่ 3

Raspberry Pi 3 กับการแสดงผลด้วย
จอแสดงผล OLED ผ่านบัส I²C

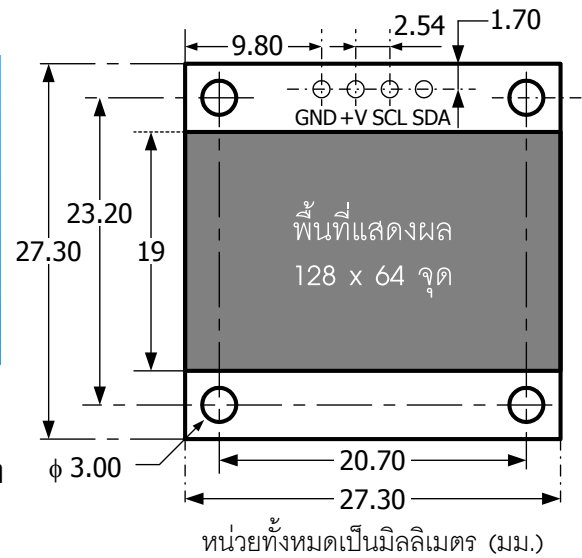
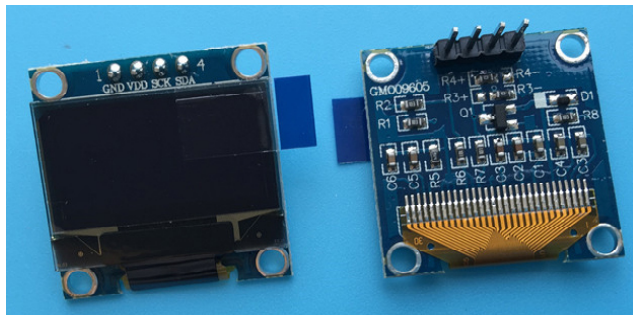
อุปกรณ์แสดงผลแบบหนึ่งที่ระบบสมองกลฝังตัวสมัยใหม่ซึ่งใช้ซีพียูความเร็วสูง และมีหน่วยความจำมากพอนิยมใช้งาน เพื่อรองรับการทำงานด้านการแสดงผลทั้งข้อความ สัญลักษณ์ และภาพกราฟิก นั่นคือ **OLED** หรือ **Organic Light Emitting Diodes** เป็นจอแสดงผลที่มีลักษณะคล้ายแผ่นฟิล์ม เปล่งแสงได้เมื่อได้รับแรงดันไฟฟ้าด้วยกระบวนการอิเล็กทรอนิกส์ (Electroluminescence) ไม่ต้องพึ่งพาแสงไฟส่องหลังหรือแบ็กไลต์ ไม่มีการแสดงผลในบริเวณที่เป็นพื้นสีดำ ทำให้บริเวณที่ไม่มีการแสดงผลมีสีดำสนิท ช่วยให้เห็นส่วนที่แสดงผลได้อย่างชัดเจน อีกคุณสมบัติที่โดดเด่นของจอแสดงผล OLED คือ ใช้พลังงานไฟฟ้าน้อยมาก

3.1 จอแสดงผล OLED 128 x 64

สำหรับจอแสดงผล OLED ที่นำมาแนะนำให้ใช้งานกันในวันนี้คือ จอแสดงผลที่ใช้ชิปควบคุมเบอร์ SSD1306 ของ Solomon Systech Limited มีคุณสมบัติทางเทคนิคที่สำคัญดังนี้

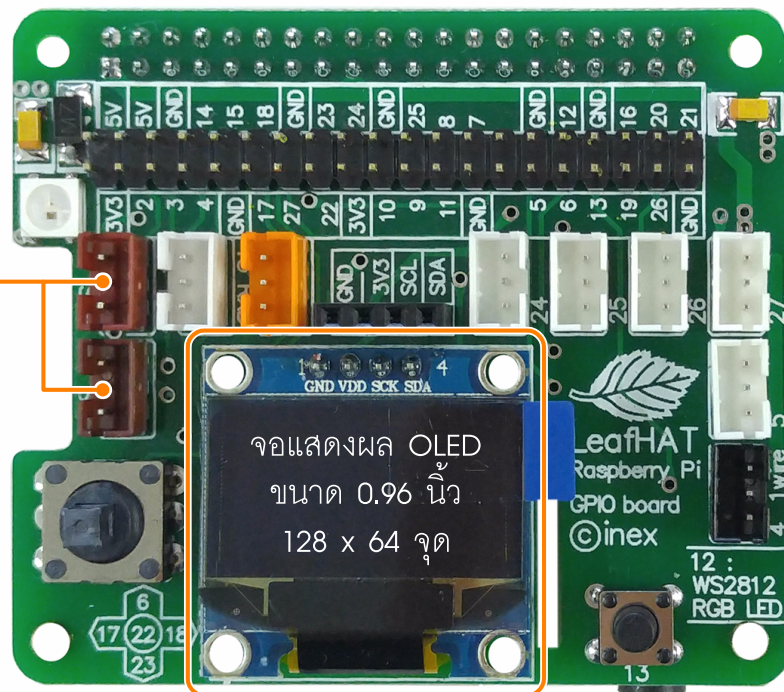
- ขนาดของจอแสดงผล 0.96 นิ้ว (วัดตามแนวเส้นทะแยงมุม)
- ใช้ชิปควบคุมเบอร์ SSD1306 หรือเทียบเท่า
- ความละเอียด 128 x 64 จุด
- ให้แสงสีขาวหรือฟ้า
- ติดต่อผ่านระบบบัส I²C
- ไฟเลี้ยง +3.3V 20mA บนบอร์ดมีวงจรทวิแรงดัน +4.5V เพื่อขับ LED

ในรูปที่ 3-1 แสดงหน้าตาของจอแสดงผล OLED 128x64 และการจัดขาสัญญาณ สำหรับการติดต่อกับ Raspberry Pi 3 ในหนังสือเล่มนี้จะเป็นการทำงานผ่านบอร์ด LeafHAT ซึ่งได้ติดตั้งจอแสดงผล OLED นี้ไว้แล้ว และทำการเชื่อมต่อกับขาบัส I²C ของพอร์ต GPIO ของบอร์ด Raspberry Pi 3 ไว้เรียบร้อยแล้ว สำหรับรูปที่ 3-2 แสดงตำแหน่งของจอแสดงผล OLED ที่ติดตั้งบนบอร์ดอุปกรณ์อินพุตเอาต์พุต LeafHAT



รูปที่ 3-1 จอแสดงผล OLED 128x64 และการจัดขา
สำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์

จุดต่อพอร์ต I²C ของ
Raspberry Pi แบบ JST
ต่อพ่วงกับขาบััส I²C
ของจอแสดงผล OLED
เพื่อเชื่อมต่อกับอุปกรณ์
ระบบบััส I²C อื่นๆ
เพิ่มเติม



รูปที่ 3-2 แสดงตำแหน่งการติดตั้งจอแสดงผล OLED 128x64 บนบอร์ด LeafHAT เพื่อเชื่อมต่อกับ
Raspberry Pi ผ่านทางจุดต่อพอร์ต GPIO 40 ขา

3.2 ความรู้เบื้องต้นของระบบบัส I²C

I²C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I²C ได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลักคือ ต้องการให้ไอซีหรือโมดูลสามารถติดต่อทำงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือ สายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I²C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานกันไป ส่วนการกำหนดแอดเดรสสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลร่วมกับการกำหนดสถานะลอจิกที่ขาแอดเดรส

สายข้อมูลบนบัส I²C มีชื่อเรียกว่า สายข้อมูลอนุกรม หรือ *SDA (Serial Data line)* ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรม หรือ *SCL (Serial Clock line)*

3.2.1 คุณสมบัติโดยทั่วไปของบัส I²C

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทาน पुलล์อัพกับแรงดันไฟเลี้ยง (+5V หรือ +3.3V) ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน อัตราการถ่ายเทข้อมูลบนบัส I²C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (standard mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (fast mode) อุปกรณ์ที่ต่ออยู่บนบัส I²C จะต้องมีความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400pF การเข้าถึงอุปกรณ์บนบัส I²C ใช้ข้อมูล 2 แบบคือ แบบ 7 บิต (7-bit addressing) หรือ 10 บิต (10-bit addressing)

3.2.2 หลักการของบัส I²C

บัส I²C ประกอบด้วยสายสัญญาณ 2 เส้น ดังที่ได้กล่าวมาแล้วคือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่า โพรโทคอล (protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้จะขออธิบายลักษณะ หน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I²C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่จะอธิบายการทำงานของบัส I²C ต่อไป

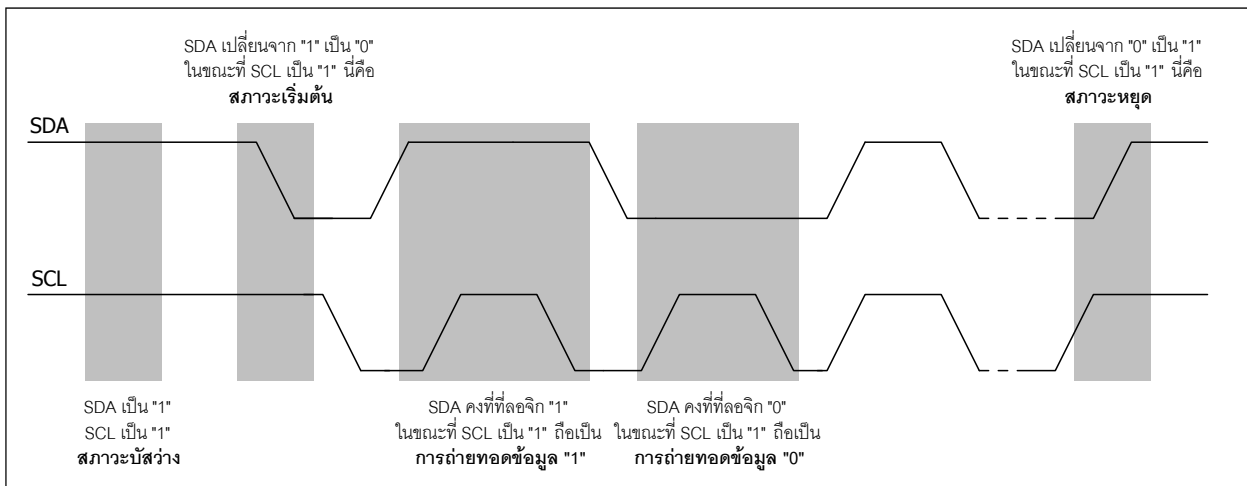
อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่า ตัวส่ง (transmitter)

อุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่า ตัวรับ (receiver)

ในอุปกรณ์บนบัส I²C สามารถเป็นได้ทั้งตัวรับและตัวส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับเพียงอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I²C ที่ทำหน้าที่เป็นตัวส่งเพียงอย่างเดียว

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการทำงานหรือการติดต่อบนบัส I²C เรียกว่า มาสเตอร์ (master)

อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I²C เรียกว่า สเลฟ (slave)



รูปที่ 3-3 แสดงไทม์ไลน์แสดงสถานะต่าง ๆ บนระบบบัส I²C

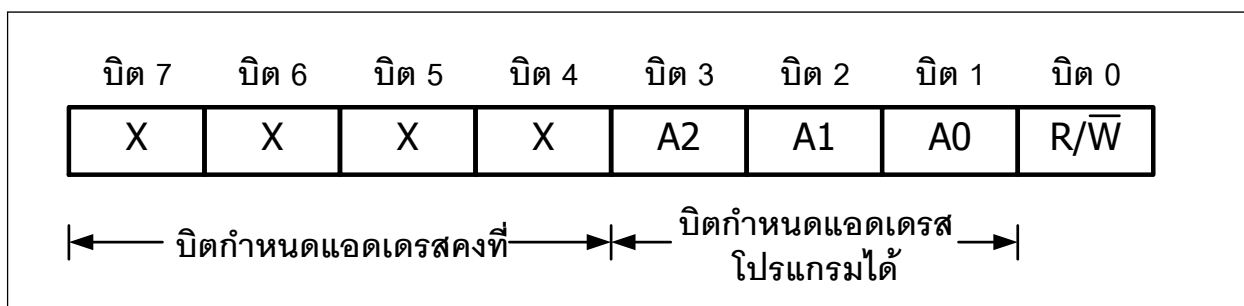
ข้อกำหนดประการสำคัญของการติดต่อบนบัส I²C คือ

- (1) การถ่ายทอข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น
- (2) ในระหว่างการถ่ายทอข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เปลี่ยนแปลงเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

3.2.3 สถานะที่เกิดขึ้นบนบัส I²C

มีด้วยกัน 5 สถานะ ดังนี้ (ดูรูปที่ 3-3 ประกอบ)

- (1) **บัสว่าง (Bus not busy)** สถานะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายทอข้อมูลสามารถเริ่มต้นขึ้นได้
- (2) **เริ่มต้นการถ่ายทอข้อมูล (start data transfer)** เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสถานะนี้ว่า สถานะเริ่มต้น (START)
- (3) **ข้อมูลดำรงอยู่บนบัส (data valid)** สถานะนี้เกิดขึ้นถัดจากสถานะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำกรถ่ายทอ เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่าเป็น "0" หรือ "1" ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอข้อมูลจะแปลความหมายเป็นสถานะหยุดหรือสถานะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ถ่ายทอนั้นเกิดความผิดพลาดขึ้น



รูปที่ 3-4 รูปแบบข้อมูลในการอ้างแอดเดรสของบัส I²C

(4) **รับรู้ข้อมูล (acknowledge)** เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิตเรียกว่า บิตรับรู้ (acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำหนดบิตรับรู้ที่มีสถานะลอจิกต่ำเพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว

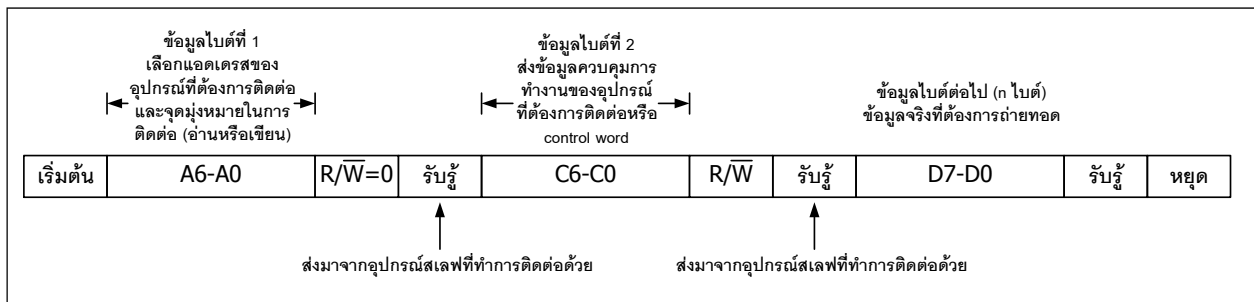
(5) **หยุดการถ่ายทอดข้อมูล (stop data transfer)** เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด (STOP)

3.2.4 การทำงานบนบัส I²C

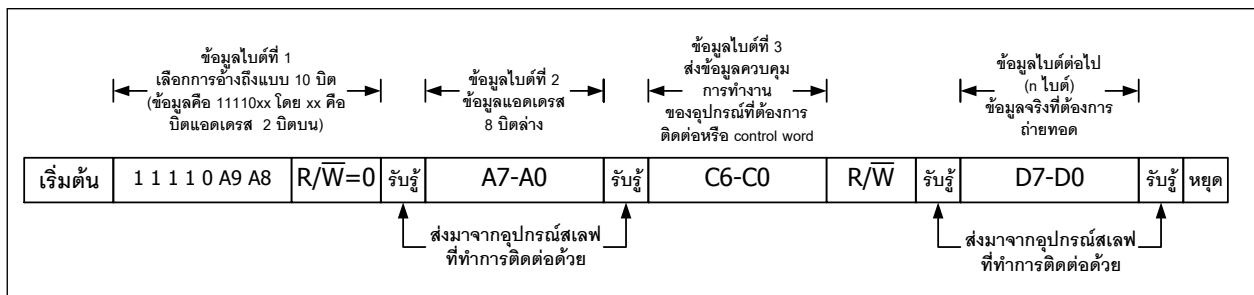
ก่อนที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆ ที่ต่ออยู่บนบัส ต้องมีการอ้างถึงอุปกรณ์เสียก่อน โดยการอ้างถึงอุปกรณ์บนบัส I²C นั้นจะใช้การอ้างถึงแบบ 7 บิตหรือ 10 บิต ในกรณีที่มิอุปกรณ์ต่ออยู่บนบัสไม่มาก ใช้การอ้างถึงแบบ 7 บิตก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้ว ก็จะเริ่มต้นการถ่ายทอดข้อมูลกันไป

3.2.4.1 การอ้างถึงแบบ 7 บิต (7-bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อดังแสดงในรูปที่ 3-4 โดยแบ่งเป็น บิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0 ถึง A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I²C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB เป็น “0” หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น “1” จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ



รูปที่ 3-5 รูปแบบข้อมูลในการอ้างถึงแบบ 7 บิต



รูปที่ 3-6 รูปแบบข้อมูลในการอ้างถึงแบบ 10 บิต

ข้อมูลในไบต์ต่อมาก็คือ ข้อมูลควบคุม (control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่าง ไอซีขยายพอร์ตมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต บิตใดเป็นเอาต์พุต ในขณะที่ไอซีแปลงสัญญาณอะนาลอกเป็นดิจิทัล/ดิจิทัลเป็นอะนาลอก (ADC/DAC) ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมาก็คือ ข้อมูลที่ทำการถ่ายทอดจริง (data)

หลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง ในรูปที่ 3-5 แสดงรูปแบบข้อมูลอนุกรมที่เกิดขึ้นในการติดต่อบนบัส I²C ของการอ้างถึงแบบ 7 บิต

3.2.4.2 การอ้างถึงแบบ 10 บิต

จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย จากรูปที่ 3-6 ประกอบ โดยในไบต์แรกหลังจากเกิดสถานะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่า ต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นก็จะจะเป็นข้อมูลจริงที่ใช้ในการติดต่อ

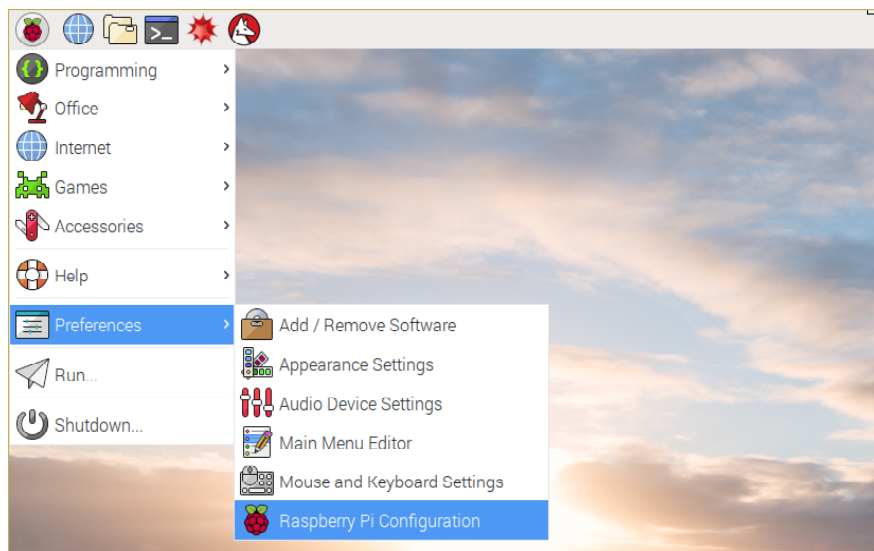
เช่นเดียวกับการอ้างถึงแบบ 7 บิต หลังจากถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสถานะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

3.3 การเปิดการติดต่อกับอุปกรณ์บััส I²C แก่บอร์ด Raspberry Pi

ในระบบปฏิบัติการ Raspbian Jessie with Pixel รุ่นใหม่ ได้เตรียมไลบรารีพื้นฐานสำหรับใช้ในการติดต่ออุปกรณ์ภายนอกไว้อย่างครอบคลุม ผู้ใช้งานสามารถเลือกเปิดใช้งานความสามารถในการติดต่ออุปกรณ์ภายนอกทั้งผ่านบััส I²C, UART หรือ บััส 1 สาย (1-Wire) ที่หน้าต่าง Raspberry Pi Configuration สำหรับจอแสดงผล OLED ใช้การติดต่อแบบบััส I²C เมื่อต้องการใช้งานกับ Raspberry Pi 3 จะมีขั้นตอนต่างๆ ในการเปิดการติดต่อดังนี้

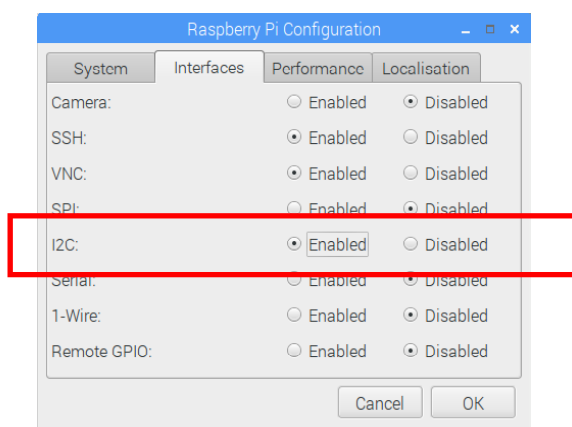
(3.3.1) เปิดความสามารถในการติดต่อบััส I²C ของ Raspberry Pi ด้วยการเข้าไปที่ **Preferences**

> **Raspberry Pi Configuration**



(3.3.2) เลือกเปิดแท็บ **Interfaces** แล้วคลิกทำเครื่องหมายเพื่อเลือกที่บรรทัดของ I²C โดย

- ถ้าต้องการเปิดใช้งานให้เลือก **Enabled**
- หากต้องการปิดการใช้งาน ให้เลือก **Disabled**



จากนั้นก็กดปุ่ม **OK** เพื่อบันทึกการตั้งค่า

3.4 ติดตั้งไลบรารีของจอแสดงผล OLED เพื่อใช้งานกับ Raspberry Pi

มีขั้นตอนโดยสรุปดังนี้

(3.4.1) เชื่อมต่อบอร์ด Raspberry Pi 3 เข้ากับเครือข่ายอินเทอร์เน็ต

(3.4.2) เปิดหน้าต่าง Terminal

(3.4.3) พิมพ์คำสั่งต่อไปนี้

```
git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git
OLED_python3
```

```
cd OLED_python3
```

```
sudo python3 setup.py install
```

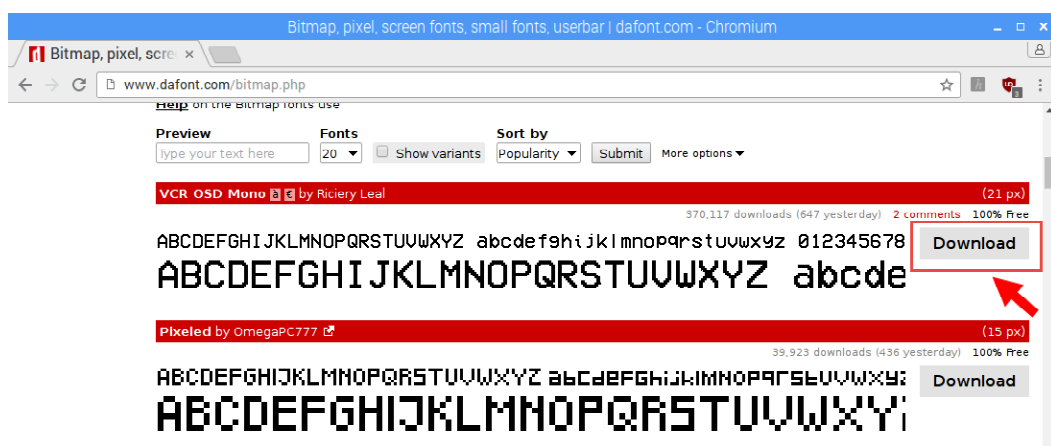
จากนั้นรออนการติดตั้งไลบรารีของจอแสดงผล OLED ลงในระบบปฏิบัติการ Raspbian Jessie with Pixel เสร็จสิ้น

ขอขอบคุณ *Adafruits Industries (www.adafruits.com)* ที่ได้จัดทำไลบรารีสำหรับติดต่อกับจอแสดงผลตัวนี้ที่รองรับโปรแกรมที่เขียนขึ้นด้วยภาษา Python 3.0

3.5 การเพิ่มฟอนต์ของตัวอักษรที่ใช้กับจอแสดงผล OLED

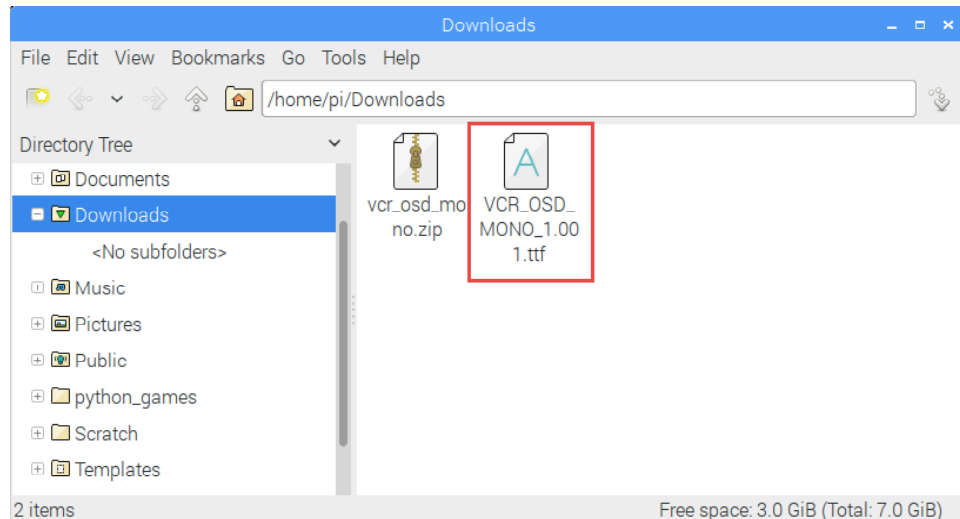
เนื่องจากจอแสดงผล OLED เป็นแบบกราฟิก จึงทำให้สามารถแสดงผลตัวอักษรและภาพกราฟิกได้อย่างหลากหลาย โดยต้องมีการเพิ่มฟอนต์ (font) หรือแบบตัวอักษรเข้าไปในระบบเสียก่อน ขั้นตอนต่างๆ มีดังนี้

(3.5.1) เปิดเว็บเบราว์เซอร์ แล้วเข้าไปที่ <http://www.dafont.com/bitmap.php> เพื่อเลือกฟอนต์จากตัวอย่างเลือกเป็น **CR OSD Mono** จากนั้นคลิกที่ปุ่ม **Download**



(3.5.2) เมื่อดาวน์โหลดเสร็จแล้ว จะได้เป็นไฟล์ .zip

(3.5.3) ทำการแตกไฟล์ จะได้เป็นไฟล์ .ttf จากให้คัดลอกไฟล์ .ttf ไปเก็บไว้ที่โฟลเดอร์เดียวกับโปรแกรมที่ใช้งานจอแสดงผล OLED ที่สร้างไว้



3.6 ตรวจสอบการเชื่อมต่อจอแสดงผล OLED กับ Raspberry Pi 3

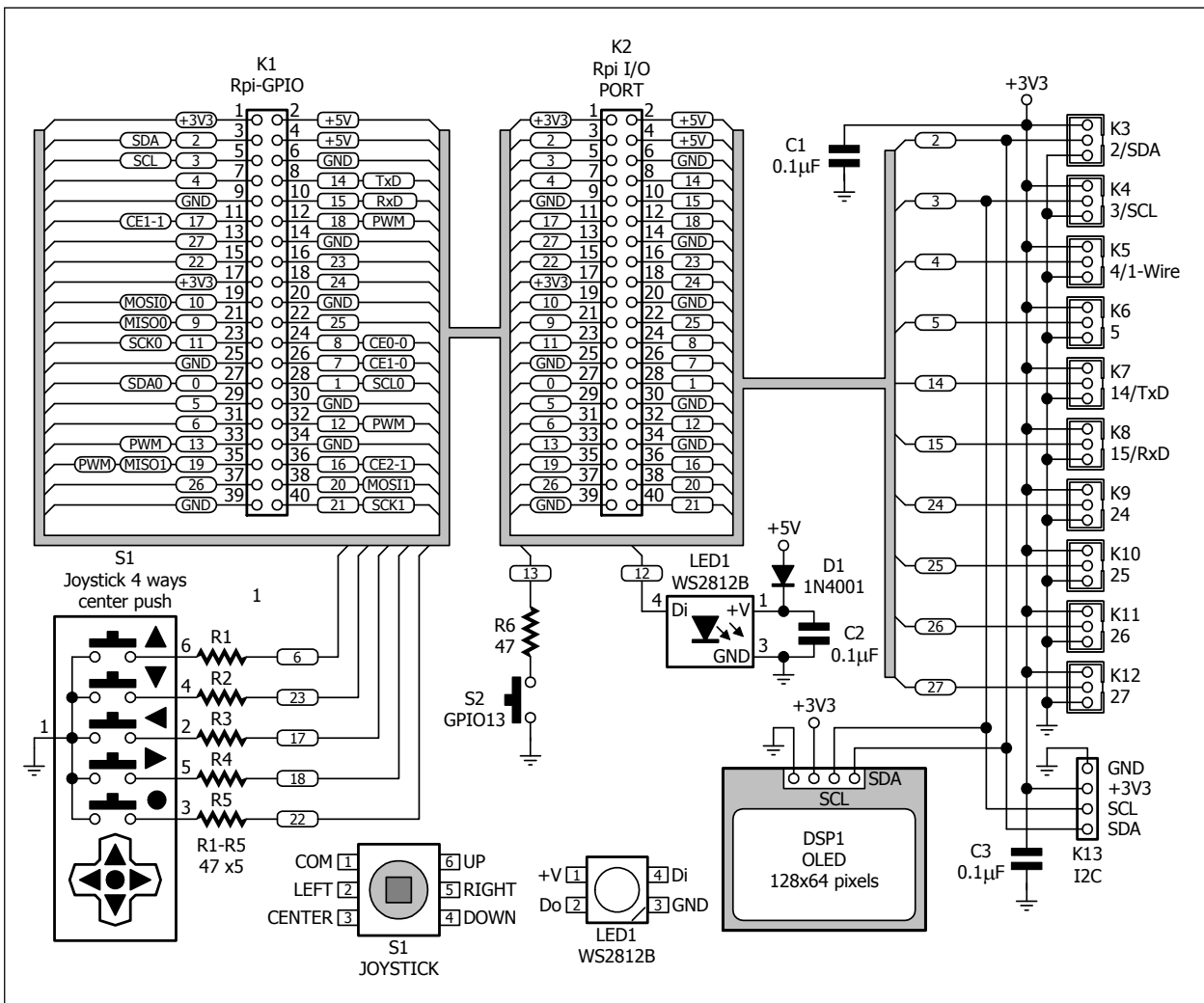
หลังจากติดตั้งไลบรารีและฟอนต์ให้กับจอแสดงผล OLED แล้ว ลำดับต่อไปเป็นการตรวจสอบการเชื่อมต่อระหว่างจอแสดงผล OLED กับบอร์ด Raspberry Pi 3

เปิดหน้าต่าง Terminal พิมพ์คำสั่งต่อไปนี้ เพื่อตรวจสอบการเชื่อมต่อ

```
sudo i2cdetect -y 1
```

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  -- 3c  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

หน้าต่างแสดงการเชื่อมต่ออุปกรณ์เข้ากับระบบบัส I²C ปรากฏขึ้นมา จากรูปจะพบว่า มีอุปกรณ์บัส I²C ต่ออยู่ 1 ตัว มีค่าแอดเดรสเท่ากับ 0x3C



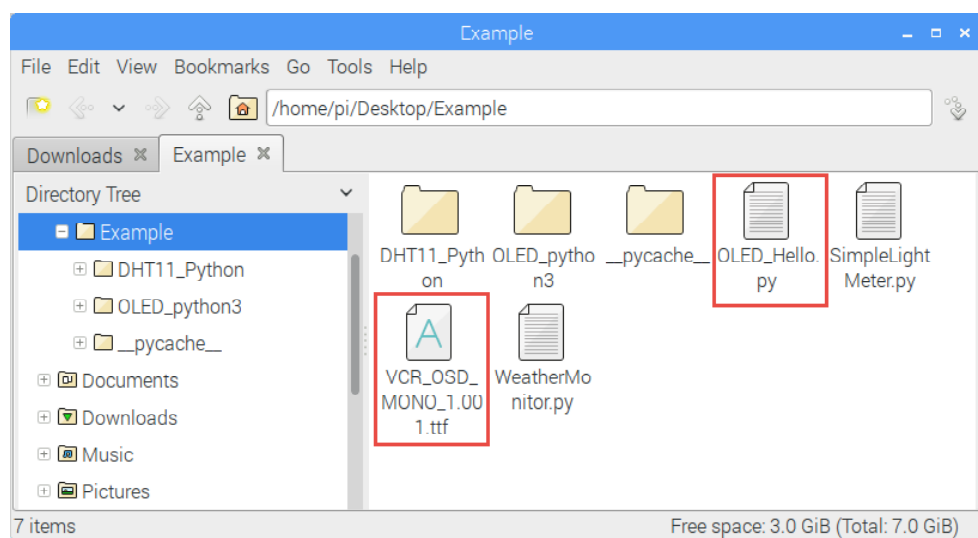
รูปที่ 2-3 วงจรของบอร์ด LeafHAT จะเห็นว่า มีการเชื่อมต่อจอแสดงผล OLED 128 x 64 พิกเซล (DSP1) เข้ากับจุดต่อที่ทำหน้าที่เป็นขาสัญญาณบัส I²C ของพอร์ต GPIO ของ Raspberry Pi 3

3.7 ตัวอย่างโปรแกรมแสดงข้อความบนจอแสดงผล OLED

ด้วยการใช้บอร์ด LeafHAT ทำให้การเชื่อมต่อกับจอแสดงผล OLED สำหรับ Raspberry Pi ทำได้ง่ายขึ้น การทดสอบเพื่อใช้งานจึงมีเพียงขั้นตอนการเขียนโปรแกรมเท่านั้น

(3.7.1) พิมพ์โปรแกรมตัวอย่างตามที่แสดงในโปรแกรมที่ 3-1

(3.7.2) ทำการบันทึกไฟล์ชื่อ **OLED_Hello.py** แล้วตรวจสอบว่า ไฟล์ของฟอนต์ถูกเก็บไว้ในโฟลเดอร์เดียวกับโปรแกรมหรือไม่ จากตัวอย่างทำการเก็บไฟล์ **OLED_Hello.py** และ **VCR_OSD_MONO_1.001.ttf** ไว้ในโฟลเดอร์ **Example** ดังรูป



```

import time
import Adafruit_SSD1306

from PIL import Image, ImageDraw, ImageFont

# Raspberry Pi pin configuration:
RST = 1

# 128x64 display with hardware I2C:
disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)

# Note you can change the I2C address by passing an i2c_address
parameter like:
# disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST, i2c_address=0x3C)

# Initialize library.
disp.begin()
# Clear display.
disp.clear()
count=0
while(1):
    count=count+1
    # Create blank image for drawing.
    # Make sure to create image with mode '1' for 1-bit color.
    width = disp.width
    height = disp.height
    image = Image.new('1', (width, height))

    # Get drawing object to draw on image.
    draw = ImageDraw.Draw(image)

    # Load default font.
    # font = ImageFont.load_default()
    # Alternatively load a TTF font.
    # Make sure the .ttf font file is in the same directory as the python script!
    # Some other nice fonts to try: http://www.dafont.com/bitmap.php
    font = ImageFont.truetype(VCR_OSD_MONO_1.001.ttf, 22)
    draw.text((0,0), 'Hello', font=font, fill=1)
    draw.text((30,20), 'World!', font=font, fill=1)
    draw.text((0,40), str(count), font=font, fill=1)

    # Display image.
    disp.image(image)
    disp.display()

```

โปรแกรมที่ 3-1 ไฟล์ OLED_Hello.py โปรแกรมภาษา Python สำหรับ Raspberry Pi 3 ในการใช้งานกับจอแสดงผล OLED เพื่อแสดงข้อความ (มีต่อ)

คำอธิบายโปรแกรมเพิ่มเติม

การทำงานของโปรแกรมแบ่งเป็น 2 ส่วนดังนี้

1. ส่วนของกระบวนการสร้างรูปภาพขึ้นมาเพื่อรอการแสดงผลบนจอ OLED ประกอบด้วย

1.1 สร้างรูปภาพขนาดเท่ากับหน้าจอแสดงผล OLED ด้วยคำสั่ง

```
image = Image.new('1', (width, height))
```

1.2 กำหนดฟอนต์ตัวอักษรและขนาดตัวอักษรด้วยคำสั่ง

```
font = ImageFont.truetype('VCR_OSD_MONO_1.001.ttf', 22)
```

โดยฟอนต์ตัวอักษรจะมีนามสกุลของไฟล์คือ ttf และต้องกำหนดที่อยู่ของไฟล์ให้ถูกต้องด้วย จากตัวอย่างในหัวข้อ 3.6 ถูกจัดเก็บไว้ในโฟลเดอร์เดียวกันกับโปรแกรม OLED_Hello.py นอกจากนั้นผู้ใช้งานยังสามารถดาวน์โหลดฟอนต์ตัวอักษรเพิ่มเติมตามที่ต้องการได้จากเว็บไซต์ <http://www.dafont.com/bitmap.php>

1.3 วาดตัวอักษรลงบนรูปภาพด้วยคำสั่ง

```
draw.text((0,0), 'Hello', font=font, fill=1)
```

โดยเริ่มที่พิกัด y=0 และ x=0 ตามลำดับเพื่อแสดงคำว่า Hello

2. ส่วนของการนำภาพมาแสดงผล ใช้คำสั่ง `disp.image(image)` และ `disp.display()` เพื่อแสดงภาพตามลำดับ

โปรแกรมที่ 3-1 ไฟล์ OLED_Hello.py โปรแกรมภาษา Python สำหรับ Raspberry Pi 3 ในการใช้งานกับจอแสดงผล OLED เพื่อแสดงข้อความ (จบ)

(3.7.3) ทำการรันไฟล์สคริปต์ OLED_Hello.py โดยไปที่เมนู **Bulid > Execute** หรือเปิดหน้าต่าง Terminal แล้วทำการรันด้วยคำสั่ง `sudo python3 OLED_Hello.py`

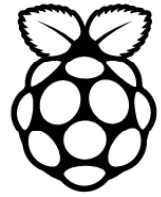
ที่จอแสดงผล OLED จะแสดงข้อความ

```
Hello
World!
777
```



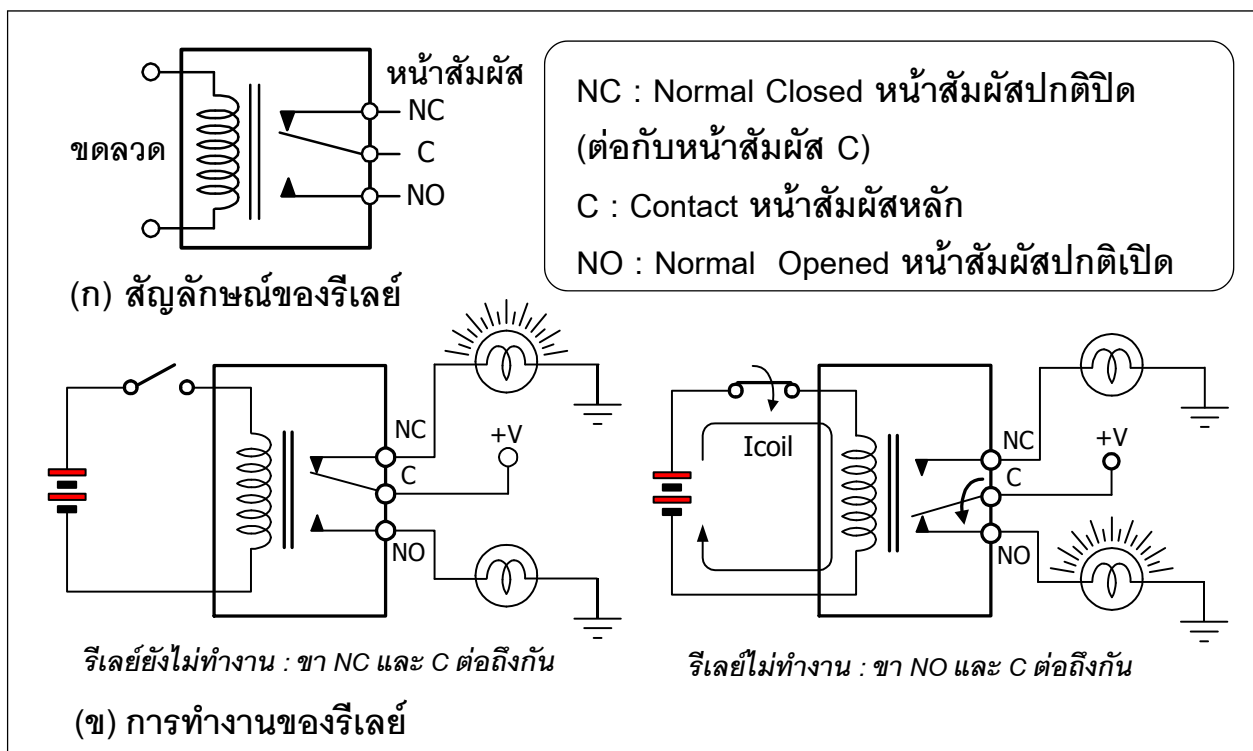
บทที่ 4

Raspberry Pi 3 กับการขับโหลดกระแสไฟฟ้าสูงด้วยรีเลย์

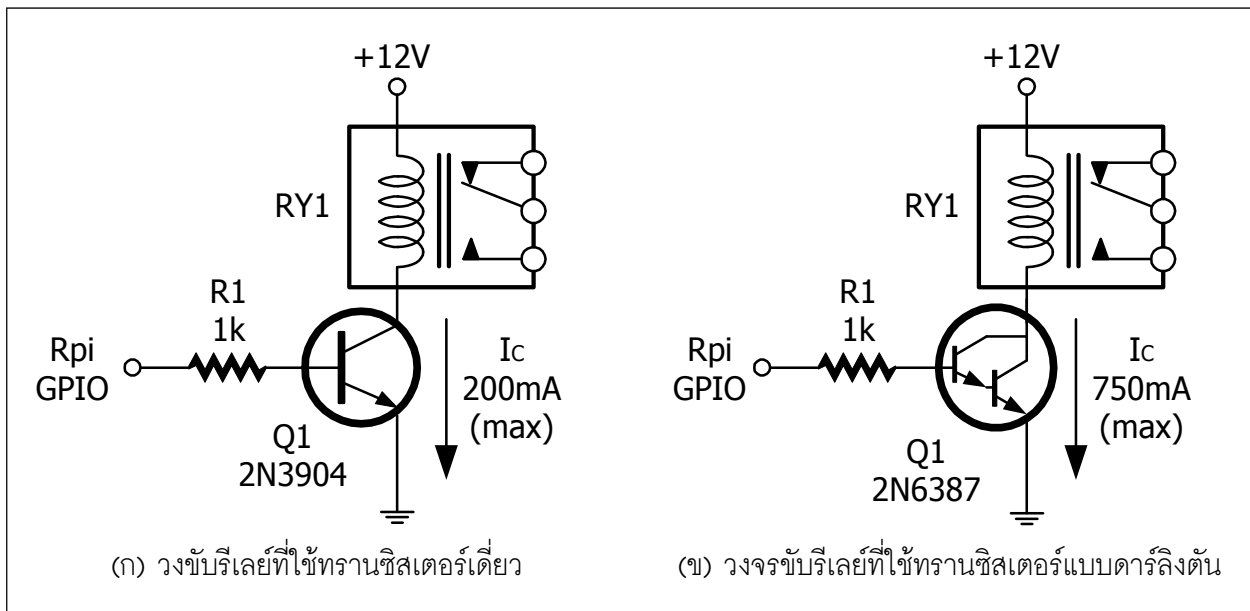


พอร์ต GPIO ของ Raspberry Pi 3 และรุ่นอื่นๆ มีความสามารถในการขับกระแสไฟฟ้าเมื่อทำงานเป็นพอร์ตเอาต์พุตได้สูงสุด 50mA ที่ +3.3V ซึ่งเพียงพอต่อกับโหลดกำลังไฟต่ำ อาทิ LED หรือ โมดูล LCD หากต้องการนำไปขับโหลดกำลังไฟฟ้าสูง เช่น หลอดไฟ หรือเครื่องใช้ไฟฟ้า จะต้องกระทำผ่านวงจรขับโหลดกระแสไฟฟ้าสูง ซึ่งมีอุปกรณ์หลักเป็นตัวช่วยคือ รีเลย์ หรือ โซลิดสเตตรีเลย์

ในบทนี้จะนำเสนอถึงข้อมูลเบื้องต้นและแนวทาง ตลอดจนตัวอย่างการเขียนโปรแกรมเพื่อควบคุมให้บอร์ด Raspberry Pi 3 ทำงานกับอุปกรณ์และวงจรขับโหลดกระแสไฟฟ้าสูง ทั้งแบบที่ใช้รีเลย์แบบกลไก (mechanical relay) และ โซลิดสเตตรีเลย์ (solid state relay) อันเป็นรีเลย์ที่ใช้อุปกรณ์อิเล็กทรอนิกส์กำลังทำหน้าที่ในการตัดต่อแรงดันไฟฟ้า ที่มีการแยกระบบกราวด์ของตัวควบคุมหรือบอร์ดไมโครคอนโทรลเลอร์ออกจากวงจรขับโหลดกระแสไฟฟ้าสูงออกจากกัน ทำให้ลดปัญหาด้านสัญญาณรบกวนลงได้อย่างมาก



รูปที่ 4-1 แสดงสัญลักษณ์และการทำงานเบื้องต้นของรีเลย์



รูปที่ 4-2 วงจรรีเลย์ที่ใช้ทรานซิสเตอร์

4.1 ความรู้เบื้องต้นเกี่ยวกับรีเลย์แบบกลไก

เป็นอุปกรณ์แม่เหล็กไฟฟ้าแบบหนึ่งที่ทำหน้าที่เป็นสวิตช์ตัดต่อหนึ่งชุดหรือมากกว่า ขึ้นอยู่กับจำนวนหน้าสัมผัสที่รีเลย์ตัวหนึ่ง ๆ บรรจุอยู่ รีเลย์มีสัญลักษณ์ตามรูปที่ 4-1 (ก) จะเห็นว่ารีเลย์ประกอบด้วยส่วนสำคัญ 2 ส่วนคือ ขดลวด (coil) และหน้าสัมผัส (contact) แบ่งเป็นหน้าสัมผัสปกติ (Normally Closed :NC) และปกติเปิดวงจรหรือไม่ต่อ (Normally Opened :NO)

การกระตุ้นให้รีเลย์ทำงานทำได้ง่ายมาก เพียงจ่ายแรงดันให้แก่ขดลวดในปริมาณที่ขดลวดนั้นต้องการ ก็จะทำให้แม่เหล็กไฟฟ้าเกิดขึ้นที่หน้าสัมผัส เกิดการดูดหน้าสัมผัสจากจุด NC มายังจุด NO ดังนั้นเมื่อรีเลย์ทำงานหน้าสัมผัส NO จะต่อวงจร ในขณะที่ NC จะเปิดวงจรแทน ในลักษณะนี้ทำงานเหมือนเป็นสวิตช์ 2 ทางที่ควบคุมด้วยแม่เหล็กไฟฟ้า ดังแสดงการทำงานในรูปที่ 4-1 (ข)

คุณสมบัติที่สำคัญของรีเลย์ ได้แก่

1. แรงดันตกคร่อมขดลวดที่ทำให้รีเลย์ทำงาน (V_{coil} หรือ Coil Voltage)
2. ค่าความต้านทานของขดลวด (Coil resistance) ปกติมีค่าประมาณ 100 ถึง 600 Ω
3. อัตราทนได้สูงสุดทั้งแรงดันและกระแสไฟฟ้าของหน้าสัมผัส (Contact rating)
4. อายุการใช้งาน (Operating time)
5. ตำแหน่งขาของหน้าสัมผัส NO, NC และ C รวมทั้งขาต่อใช้งานของขดลวด

4.2 วงจรขับรีเลย์

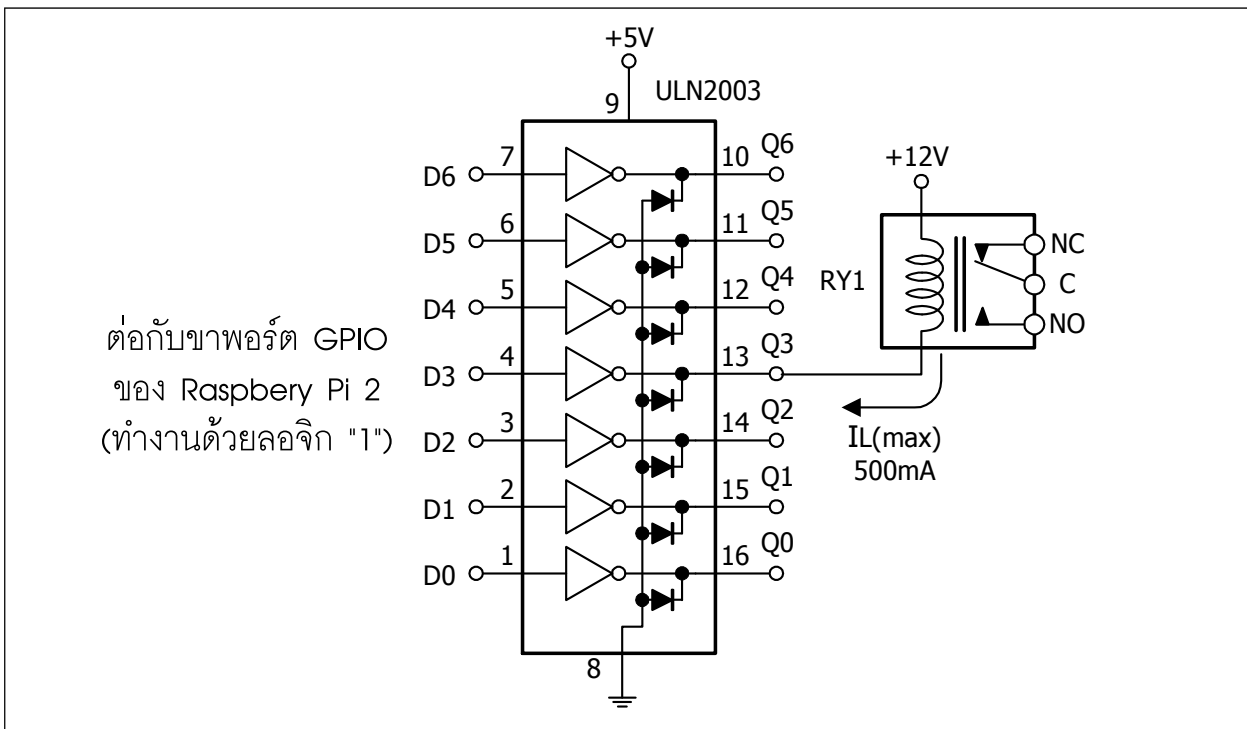
ในการทำงานปกติ พอร์ตเอาต์พุตของ Raspberry Pi และไมโครคอนโทรลเลอร์ไม่สามารถนำไปขับอุปกรณ์เอาต์พุตกระแสไฟฟ้าสูงได้โดยตรงได้ เนื่องจากข้อจำกัดด้านความสามารถในการจ่ายกระแสไฟฟ้า ดังนั้นถ้าต้องการนำ Raspberry Pi และไมโครคอนโทรลเลอร์ไปขับโหลดกระแสไฟฟ้าสูงจะต้องมีอุปกรณ์ที่ทำหน้าที่จ่ายแรงดันและกระแสไฟฟ้าสูงโดยเฉพาะ เรียกอุปกรณ์เหล่านี้ว่า อุปกรณ์ขับ หรือ ไดรเวอร์ (driver)

4.2.1 การใช้ทรานซิสเตอร์ขับรีเลย์

การขับโดยวิธีนี้เหมาะสมสำหรับโหลดที่มีความต้องการกระแสไฟฟ้าปานกลาง ตั้งแต่ 30 ถึง 200mA อาทิ รีเลย์กำลังต่ำไปจนถึงปานกลางที่มีค่าความต้านทานของขดลวดภายในรีเลย์ไม่ต่ำกว่า 100Ω , หลอดไฟกำลังต่ำ และมอเตอร์ไฟตรงขนาดเล็ก มีวงจรตัวอย่างตามรูปที่ 4-2

ในรูปที่ 4-2 (ก) เป็นการต่อทรานซิสเตอร์เข้ากับขาพอร์ต GPIO ของ Raspberry Pi โดยมีตัวต้านทาน R1 ทำหน้าที่จำกัดกระแสไฟฟ้าที่ไหลเข้าขาเบสของทรานซิสเตอร์ Q1 ซึ่งจะทำงานก็ต่อเมื่อขาพอร์ตของไมโครคอนโทรลเลอร์มีสถานะลอจิกเป็น “1” เมื่อ Q1 ทำงาน ก็จะเกิดกระแสไฟฟ้าไหลผ่าน RL ซึ่งเป็นโหลดต่ออยู่ทางเอาต์พุตที่ขาคอลเล็กเตอร์ของ Q1 กระแสไหลสูงสุด (ILmax) มีค่าเท่ากับ $12V/300\Omega = 40mA$ ถึงแม้ว่า Q1 เบอร์ 2N3904 มีค่ากระแสคอลเล็กเตอร์สูงสุดถึง 100mA แต่ในทางปฏิบัติจริงไม่ควรออกแบบให้ทรานซิสเตอร์ทำงานถึงพิกัดสูงสุด ย่านปลอดภัยของทรานซิสเตอร์ควรอยู่ไม่เกินครึ่งหนึ่งของอัตราการทนได้สูงสุด ด้วยการจัดวงจรตามรูปที่ 4-2 สามารถใช้สัญญาณจากพอร์ตเอาต์พุตกระตุ้นให้ทรานซิสเตอร์ทำงานเพื่อขับรีเลย์ขนาดเล็กได้อย่างปลอดภัย

ในรูปที่ 4-2 (ข) เป็นวงจรขับรีเลย์ที่ใช้ทรานซิสเตอร์คาร์ลิงตัน ที่บรรจุทรานซิสเตอร์ 2 ตัวต่อกันแบบคาร์ลิงตันภายใต้ตัวถังเดียวกัน เพื่อขับกระแสไฟฟ้าทางเอาต์พุตให้สูงขึ้น และมีความเร็วในการทำงานสูงด้วย ด้วยการใช้อุปกรณ์เพียงตัวเดียว ส่งผลให้ขนาดของวงจรเล็กลง จากวงจร Q1 ซึ่งเป็นทรานซิสเตอร์แบบคาร์ลิงตันสามารถขับกระแสไฟฟ้าทางเอาต์พุตได้สูงถึง 750mA โดยผ่านตัวต้านทานจำกัดกระแสเพียงตัวเดียวและไม่ต้องต่อทรานซิสเตอร์แบบคลาสเคด ทำให้มีความเร็วในการทำงานสูง ตลอดจนสามารถขับกระแสไฟฟ้าทางเอาต์พุตได้สูงพอสมควร



รูปที่ 4-3 วงจรขับรีเลย์โดยใช้ไอซีขับโหลดกระแสไฟฟ้าสูงเบอร์ ULN2003

4.2.2 การใช้ไอซีขับ

ไอซีที่ใช้ในการขับโหลดกระแสสูงมักจะมีวงจรทางเอาต์พุตเป็นแบบคอลเล็กเตอร์เปิด ทำให้ใช้กับแรงดันไฟฟ้าที่สูงได้ สำหรับไอซีขับหรือไอซีไดรเวอร์ที่ยกมาอธิบายคือเบอร์ ULN2003 เป็นไอซีที่อยู่ในบรรจุอินเวอร์เตอร์เกต 7 ตัว มีรูปแบบการจัดขาและวงจรภายในแสดงในรูปที่ 4-3 ใช้กับแรงดันได้สูงสุด +30V กระแสไฟฟ้าเอาต์พุตสูงสุดในแต่ละขาเท่ากับ 500mA ทั้งนี้ขึ้นอยู่กับความสามารถในการจ่ายกระแสไฟฟ้าของแหล่งจ่ายไฟด้วย นอกจากนี้ ULN2003 ยังมีการต่อไดโอดป้องกันแรงดันย้อนกลับจากอุปกรณ์เอาต์พุตที่มีโครงสร้างเป็นขดลวดไว้ที่ทุกขาเอาต์พุต ทำให้ใช้ขับโหลดที่เป็นขดลวด อาทิ รีเลย์ หรือมอเตอร์ไฟตรงขนาดเล็กถึงขนาดกลางได้

การกระตุ้นให้ทำงาน ทำได้ง่ายๆ โดยต่อขาพอร์ต GPIO ของ Raspberry Pi เข้าที่อินพุตของ ULN2003 โดยตรง หากต้องการให้วงจรขับชุดนั้นๆ ทำงาน เพียงส่งสัญญาณลอจิกสูง หรือ “1” มายังขาอินพุตของ ULN2003 วงจรขับที่ได้รับสัญญาณลอจิก “1” จะทำงาน กลับลอจิกที่เอาต์พุตเป็น “0” ทำให้เกิดกระแสไฟฟ้าไหลผ่านโหลดที่เป็นขดลวดรีเลย์ได้ รีเลย์ก็จะทำงาน และจะหยุดทำงานเมื่ออินพุตของ ULN2003 ได้รับลอจิกต่ำหรือ “0”

ในกรณีที่มีความต้องการขับรีเลย์หรือโหลดจำนวน 8 ช่อง ไอซี ULN2003 จะรองรับความต้องการนี้ไม่ได้ แนะนำให้เลือกใช้เบอร์ ULN2803 แทน โดย ULN2803 มีวงจรขับ 8 ช่อง โดยการกระตุ้นให้ทำงานและความสามารถในการขับโหลดเหมือนกับ ULN2003 ทุกประการ

4.3 แนะนำ RELAY4i บอร์ดขับรีเลย์ 4 ช่อง

RELAY4i เป็นแผงวงจรขับรีเลย์ 4 ช่อง ราคาประหยัด มีคุณสมบัติทางเทคนิคโดยสรุปและหน้าตาของบอร์ดแสดงในรูปที่ 4-4 ส่วนในรูปที่ 4-5 แสดงวงจรสมบูรณ์ของบอร์ด RELAY4i

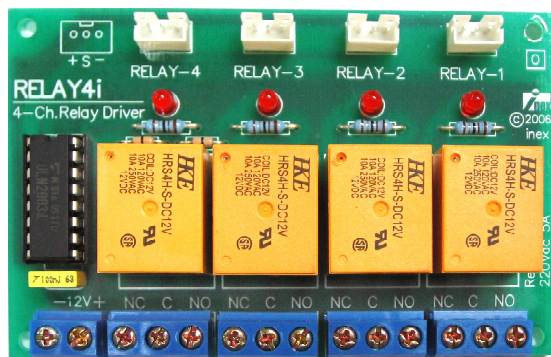
การใช้งานบอร์ดขับรีเลย์ RELAY4i สรุปเป็นขั้นตอนได้ดังนี้ (ใช้รูปที่ 4-6 อ้างอิง)

(1) ต่อโหลดที่ต้องการควบคุมเข้าที่จุดต่อหน้าสัมผัสของรีเลย์ ซึ่งมี 4 ช่อง แต่ละช่องเลือกให้ทำงานแบบต่อหรือตัดวงจรก็ได้ ปกติแล้วจะเลือกใช้งานแบบต่อวงจรมากกว่า นั่นคือ เมื่อรีเลย์ทำงานจะเป็นการต่อวงจรเพื่อจ่ายไฟเลี้ยงไปยังโหลดหรืออุปกรณ์ไฟฟ้าเพื่อให้ทำงานต่อไป จากรูปที่ 4-6 จะเห็นว่าผู้ใช้งานสามารถต่อหน้าสัมผัสรีเลย์เข้ากับเครื่องใช้ไฟฟ้าได้สูงสุด 220Vac 600W (วัตต์) โดยต่อผ่านเต้าเสียบ ในขณะที่อีกช่องหนึ่งนั้นจะต่อกับหลอดไฟ 12V ในแต่ละช่องของหน้าสัมผัสรีเลย์ต่อกับโหลดได้ทั้งแบบไฟฟ้ากระแสตรงหรือกระแสสลับ รวมถึงการต่อวงจรเพื่อทำหน้าที่เป็นเหมือนสวิตช์ธรรมดาก็สามารถทำได้

(2) หน้าสัมผัส NO หมายถึง ปกติเปิดวงจร (Normally Open) เมื่อรีเลย์ทำงานจะต่อวงจรเข้ากับขา C ดังนั้นหากต้องการใช้งานในแบบต่อวงจร ต้องเลือกต่อใช้งานหน้าสัมผัส NO และ C

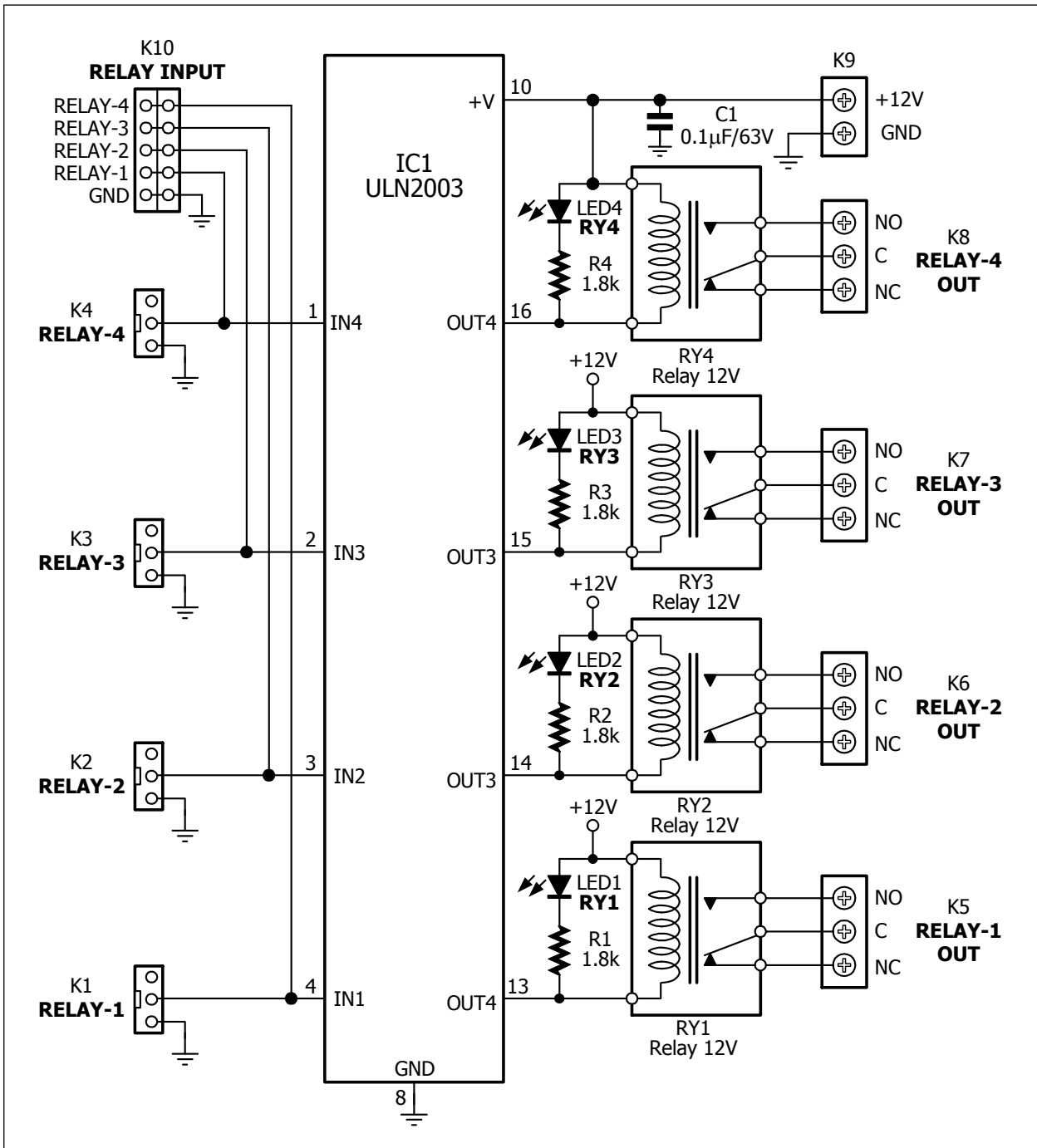
(3) หน้าสัมผัส NC หมายถึง ปกติปิดวงจร (Normally Close) เมื่อรีเลย์ทำงานจะเปิดวงจรออกจากกับขา C หากต้องการใช้งานแบบตัดวงจร ต้องเลือกต่อใช้งานหน้าสัมผัส NC และ C

(4) จุดต่อหน้าสัมผัสรีเลย์เป็นแบบขันสกรู ทำให้สามารถต่อใช้งานได้อย่างสะดวก



- ใช้ไอซีขับโหลดกระแสสูงเบอร์ ULN2003 บนบอร์ดจัดวงจรเพื่อขับรีเลย์ 12V 4 ช่อง
- ใช้ไฟเลี้ยง +12V แยกต่างหาก
- รับสัญญาณลอจิก “1” จากไมโครคอนโทรลเลอร์หรือวงจรภายนอกในการกระตุ้นให้รีเลย์ทำงาน
- มีไฟแสดงการทำงานของรีเลย์
- จุดต่อหน้าสัมผัสรีเลย์เป็นแบบขันสกรู ทำให้สามารถต่อใช้งานได้อย่างสะดวก
- อัตราทนได้ของหน้าสัมผัสรีเลย์ 220Vac 5A รองรับโหลดได้ไม่เกิน 600 วัตต์

รูปที่ 4-4 คุณสมบัติทางเทคนิคของ RELAY4i บอร์ดขับรีเลย์ 4 ช่อง



รูปที่ 4-5 วงจรสมบูรณ์ของ RELAY4i บอร์ดขับรีเลย์ 4 ช่อง

(5) อัตราทนได้ของหน้าสัมผัสรีเลย์ 220Vac 5A สามารถรองรับโหลดได้ไม่เกิน 600 วัตต์

(6) ต่อไฟเลี้ยง +12V สำหรับเลี้ยงวงจรแยกต่างหากจากไฟเลี้ยงของแผงวงจรควบคุม

(7) เมื่อต้องการให้วงจรขับรีเลย์ชุดใดทำงาน ให้ป้อนสัญญาณลอจิก “1” จากไมโครคอนโทรลเลอร์เข้าที่จุดต่ออินพุต RELAY-1 ถึง RELAY-4 โดยต่อใช้งานพร้อมกันทั้ง 4 ช่อง หรือควบคุมแยกช่องก็ได้

(8) เมื่อวงจรขับได้รับสัญญาณลอจิก “1” ไอซีขับบนบอร์ด RELAY4i ทำงาน จะได้ยิน เสียงหน้าสัมผัสรีเลย์ตัดต่อ พร้อมกับไฟแสดงการทำงานของรีเลย์ติดสว่าง หากต้องการหยุดการทำงาน ให้ส่งสัญญาณลอจิก “0” เข้ามาที่อินพุตของวงจร

4.4 ตัวอย่างการทดลองใช้งาน Raspberry Pi 3 ขับโหลดกระแสไฟฟ้าสูงด้วยรีเลย์แบบกลไก

4.4.1 เตรียมอุปกรณ์

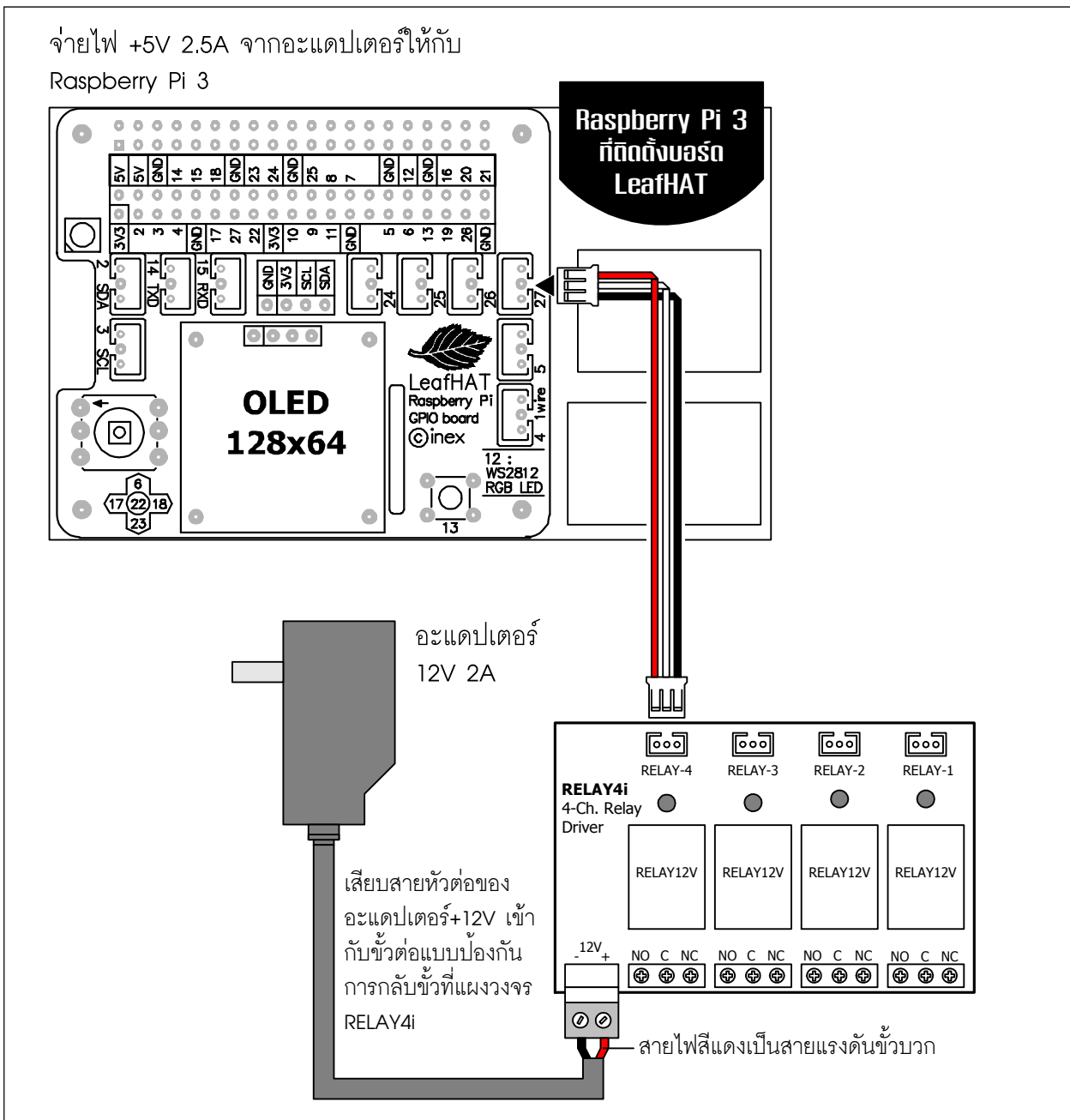
ประกอบด้วย

1. บอร์ด Raspberry Pi 3
2. บอร์ด LeafHAT ติดตั้งบน Raspberry Pi 3 ผ่านคอนเน็กเตอร์ GPIO 40 ขา
3. บอร์ด RELAY4i
4. อะแดปเตอร์ 12V 1A
5. สาย JST3AA-8 จำนวน 4 เส้น
6. โหลดกำลังไฟฟ้าสูง พร้อมสายต่อไฟสลับ 220Vac

4.4.2 วงจรสวิตช์ควบคุมรีเลย์ด้วย Raspberry Pi 3

มีขั้นตอนดังนี้

- (1) เชื่อมต่อบอร์ด Raspberry Pi 3 กับอุปกรณ์ทั้งหมดเพื่อสร้างเป็นระบบควบคุมโหลดกระแสไฟฟ้าสูงอย่างง่าย ดังรูปที่ 4-6
- (2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์
- (3) พิมพ์คำสั่งตามโปรแกรมที่ 4-1 แล้วบันทึกชื่อไฟล์เป็น **RelaySW2.py**



รูปที่ 4-6 วงจรทดลองการเชื่อมต่อ Raspberry Pi 3 กับวงจรขับโหลดกระแสไฟฟ้าสูงด้วยรีเลย์แบบกลไก


```

import RPi.GPIO as GPIO
import Adafruit_SSD1306
from PIL import Image, ImageDraw, ImageFont
import time

disp = Adafruit_SSD1306.SSD1306_128_64(rst=1)
disp.begin()
disp.clear()
disp.display()

Rel_1=27
sw_on=22
sw_off=13
state=0;

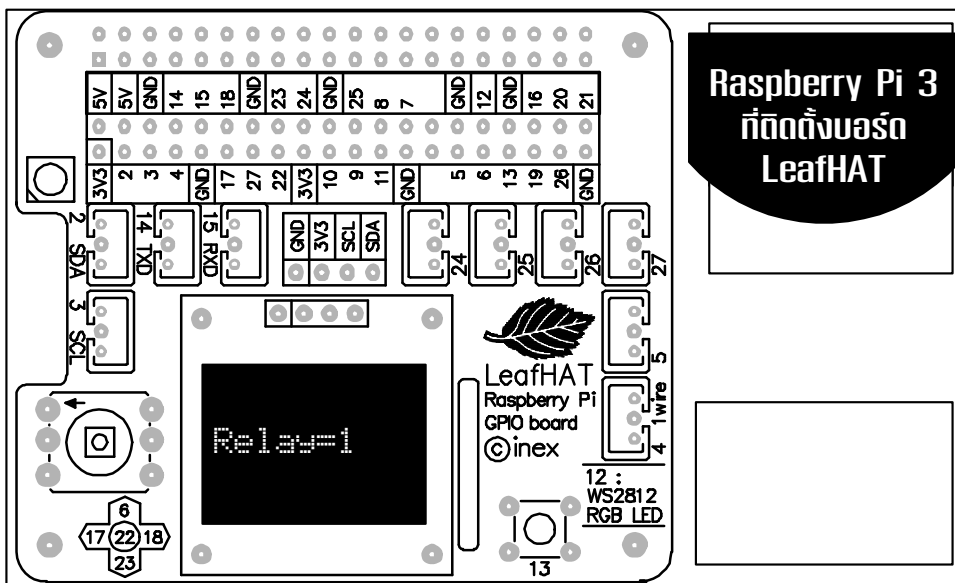
GPIO.setmode(GPIO.BCM)
GPIO.setup(Rel_1,GPIO.OUT)
GPIO.setup(sw_on,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(sw_off,GPIO.IN,pull_up_down=GPIO.PUD_UP)

while (1):
    if(GPIO.input(sw_on)==0):
        state=1
    if(GPIO.input(sw_off)==0):
        state=0
    GPIO.output(Rel_1,state)
    msg="Relay=%d" % state

    # Display image.
    width = disp.width
    height = disp.height
    image = Image.new('1', (width, height))
    draw = ImageDraw.Draw(image)
    font = ImageFont.truetype('VCR_OSD_MONO_1.001.ttf',22)
    draw.text((0,20),msg,font=font, fill=1)
    disp.image(image)
    disp.display()

```

โปรแกรมที่ 4-1 โค้ด Python สำหรับบอร์ด Raspberry Pi 3 เพื่อใช้งานขาพอร์ต GPIO ในการรับค่าจากสวิตช์ 2 ตัวเพื่อนำมาขับโหลดกระแสไฟฟ้าสูงผ่านวงจรขั้วรีเลย์ 1 ช่องโดยใช้ RELAY4i บอร์ดขั้วรีเลย์ 4 ช่อง



รูปที่ 4-7 แสดงสถานะการทำงานของรีเลย์จากการทำงานของโปรแกรมที่ 4-1 บนจอแสดงผล OLED ของบอร์ด LeafHAT ที่เชื่อมต่อกับบอร์ด Raspberry Pi 3 ผ่านทางพอร์ต GPIO 40 ขา

(4) ตั้งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง `sudo python3 RelaySW2.py`

ในตัวอย่างนี้ จะใช้สวิทช์ 2 ตัวในการควบคุมการทำงานของรีเลย์ 1 ช่อง เมื่อเริ่มต้นการทำงาน รีเลย์ไม่ทำงานจนกว่าจะมีการกดสวิทช์ที่ต่อกับขา GPIO22 ซึ่งก็คือปุ่มกลางของสวิทช์จอยสติ๊กบนบอร์ด LeafHAT เมื่อมีการกดสวิทช์เกิดขึ้น โปรแกรมจะส่งสัญญาณลอจิก “1” ไปยังขาพอร์ต GPIO27 เพื่อกระตุ้นให้วงจรขับรีเลย์ทำงาน สังเกตได้จาก LED ของวงจรขับรีเลย์ที่ทำงานจะติดสว่าง

หากต้องการให้รีเลย์หยุดทำงาน ทำได้ด้วยการกดสวิทช์ที่ต่อกับขา GPIO13 บนบอร์ด LeafHAT เมื่อมีการกดสวิทช์เกิดขึ้น โปรแกรมจะส่งสัญญาณลอจิก “0” ไปยังขา GPIO22 เพื่อหยุดการทำงานของรีเลย์ทำงาน พร้อมกันนั้น ที่จอแสดงผล OLED บนบอร์ด LeafHAT จะแสดงสถานะการทำงานของรีเลย์ดังรูปที่ 4-7 โดย **Relay=1** หมายถึง รีเลย์ทำงาน และ **Relay=0** หมายถึง รีเลย์หยุดทำงาน

4.4.3 วงจรสวิทช์ควบคุมรีเลย์แบบกดเปิดกดปิด

ในตัวอย่างนี้พัฒนาต่อจากหัวข้อ 4.4.2 ซึ่งใช้สวิทช์ 2 ตัวสำหรับเลือกให้รีเลย์ทำงานและหยุดทำงาน โดยในตัวอย่างนี้จะลดสวิทช์เหลือ 1 ตัวเพื่อควบคุมการทำงานของรีเลย์ มีขั้นตอนการทดลองดังนี้

```

import RPi.GPIO as GPIO
import Adafruit_SSD1306
from PIL import Image, ImageDraw, ImageFont
import time

disp = Adafruit_SSD1306.SSD1306_128_64(rst=1)
disp.begin()
disp.clear()
disp.display()

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

sw_onoff = 22
Rel_1=27

GPIO.setup(Rel_1,GPIO.OUT)
GPIO.setup(sw_onoff,GPIO.IN,pull_up_down=GPIO.PUD_UP)

def sw_pressed(channel):
    global msg
    state=not(GPIO.input(Rel_1))
    GPIO.output(Rel_1,state)

GPIO.add_event_detect(sw_onoff,GPIO.FALLING,callback=sw_pressed,bouncetime=300)

while(1):

    msg="Relay=%d" % GPIO.input(Rel_1)

    width = disp.width
    height = disp.height
    image = Image.new('1', (width, height))
    draw = ImageDraw.Draw(image)
    font = ImageFont.truetype('VCR_OSD_MONO_1.001.ttf',22)
    draw.text((0,20),msg,font=font, fill=1)
    disp.image(image)
    disp.display()

```

คำอธิบายโปรแกรมเพิ่มเติม

ในโปรแกรมนี้ หัวใจสำคัญคือ คำสั่ง `GPIO.add_event_detect(sw_onoff,GPIO.FALLING, callback=sw_pressed,bouncetime=300)` ซึ่งใช้ในการรักษาค่าสถานะของขาพอร์ตก่อนหน้าไว้ เมื่อมีการเปลี่ยนแปลงที่ขาพอร์ต GPIO22 ซึ่งต่อกับสวิตช์อีกครั้ง ก็จะทำการกลับสถานะของขาพอร์ตเอาต์พุต จาก “1” เป็น “0” ทำให้วงจรขั้วรีเลย์ที่ต่อกับขาพอร์ต GPIO27 หยุดทำงาน

โปรแกรมที่ 4-2 โค้ด Python ไฟล์ RelayToggleSW.py สำหรับบอร์ด Raspberry Pi 3 เพื่อใช้งานขาพอร์ต GPIO 1 ขาในการรับค่าจากการกดสวิตช์ เพื่อนำมาขับโหลดกระแสไฟฟ้าสูงผ่านวงจรขั้วรีเลย์ 1 ช่องโดยใช้ RELAY4i บอร์ดขั้วรีเลย์ 4 ช่อง

(1) ยังคงใช้วงจรในรูปที่ 4-6 ในการทดลอง โดยเลือกใช้สวิตช์ 1 ตัวที่ต่อกับขาพอร์ต GPIO22 และวงจรขั้วรีเลย์ 1 ช่องซึ่งต่อกับขาพอร์ต GPIO27

(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เทกซ์เอดิเตอร์

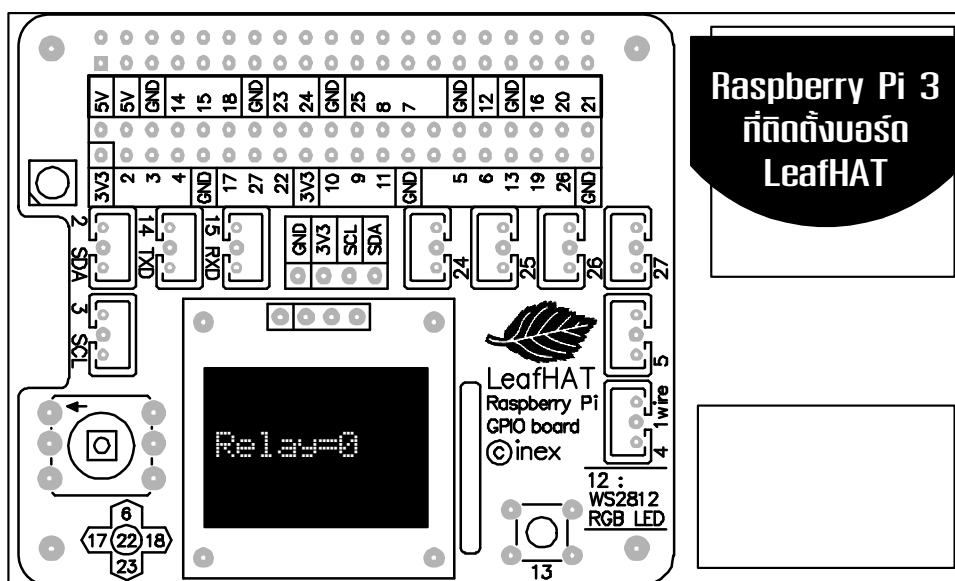
(3) พิมพ์คำสั่งตามโปรแกรมที่ 4-2 แล้วบันทึกชื่อไฟล์เป็น **RelayToggleSW.py**

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่าง Terminal แล้วสั่งรันด้วยคำสั่ง **sudo python3 RelayToggleSW.py**

ในตัวอย่างนี้ จะใช้สวิตช์ 1 ตัวในการควบคุมการทำงานของรีเลย์ 1 ช่อง เมื่อเริ่มต้นการทำงาน รีเลย์ไม่ทำงานจนกว่าจะมีการกดสวิตช์ที่ต่อกับขา GPIO22 เมื่อมีการกดสวิตช์เกิดขึ้น โปรแกรมจะส่งสัญญาณลอจิก “1” ไปยังขาพอร์ต GPIO27 เพื่อกระตุ้นให้วงจรขั้วรีเลย์ทำงาน สังเกตได้จาก LED ของวงจรขั้วรีเลย์ที่ทำงานจะติดสว่าง

จากนั้นทดลองกดสวิตช์ซ้ำ โปรแกรมจะทำการส่งสัญญาณลอจิก “0” ไปยังขาพอร์ต GPIO16 เพื่อหยุดการทำงานของวงจรขั้วรีเลย์

นอกจากนั้น ในขณะทำงาน จอแสดงผล OLED บนบอร์ด LeafHAT จะแสดงสถานะการทำงานของรีเลย์ด้วย ดังรูปที่ 4-8 โดย **Relay=1** หมายถึง รีเลย์ทำงาน และ **Relay=0** หมายถึง รีเลย์ หยุดทำงาน



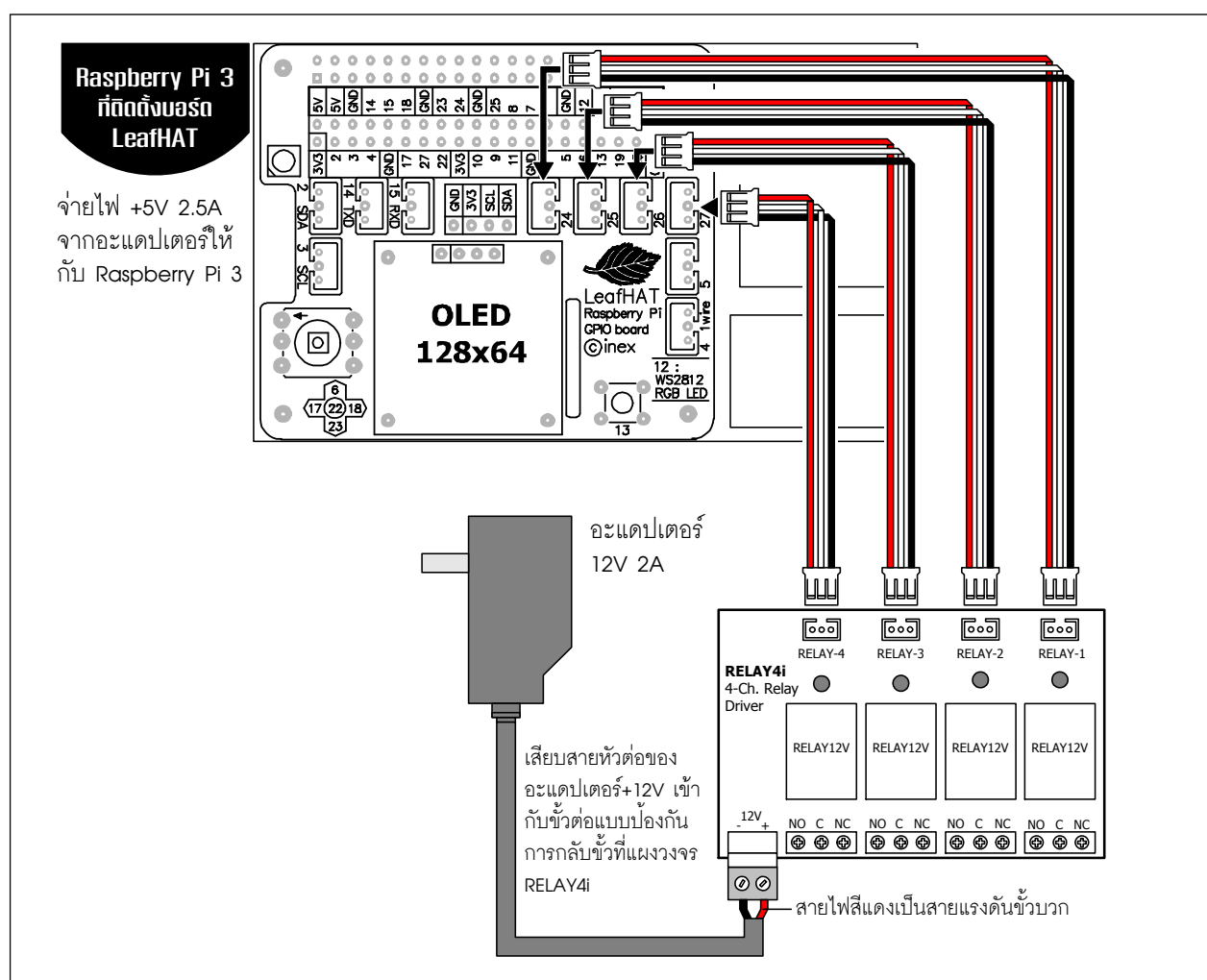
รูปที่ 4-8 แสดงสถานะการทำงานของรีเลย์จากการทำงานของโปรแกรมที่ 4-2 บนจอแสดงผล OLED ของบอร์ด LeafHAT ที่เชื่อมต่อกับบอร์ด Raspberry Pi 3 ผ่านทางพอร์ต GPIO 40 ขา

4.4.4 วงจรชีวิตควบคุมรีเลย์แบบกดเปิดกดปิด 4 ช่อง

ในตัวอย่างนี้พัฒนาต่อจากหัวข้อ 4.4.3 โดยจะทำการควบคุมรีเลย์ 4 ตัว โดยใช้สวิตช์ 4 ตัวแยกควบคุมกันอย่างอิสระ มีขั้นตอนการทดลองดังนี้

(1) ใช้วงจรในรูปที่ 4-9 ในการทดลอง โดยในตัวอย่างนี้ใช้สวิตช์จอยสติ๊ก 4 ทิศทางที่ต่อกับขาพอร์ต์ GPIO6, 18, 23 และ 17 ร่วมกับการใช้วงจรขั้วรีเลย์ 4 ช่องซึ่งต่อกับขาพอร์ต์ GPIO24, 25, 26 และ 27 เรียงตามลำดับ

ขาพอร์ตที่ต่อกับสวิทช์	ขาพอร์ตที่เชื่อมต่อกับบอร์ด RELAY4
GPIO6 : S1	GPIO24 : RELAY-1
GPIO18 : S2	GPIO25 : RELAY-2
GPIO23 : S3	GPIO26: RELAY-3
GPIO17 : S4	GPIO27 : RELAY-4



รูปที่ 4-9 วงจรทดลองการเชื่อมต่อ Raspberry Pi 3 กับวงจรขับโหลดกระแสไฟฟ้าสูงด้วยรีเลย์ 4 ช่อง

```

import RPi.GPIO as GPIO
import Adafruit_SSD1306
from PIL import Image, ImageDraw, ImageFont
import time

disp = Adafruit_SSD1306.SSD1306_128_64(rst=1)
disp.begin()
disp.clear()
disp.display()

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

sw_1 = 6
sw_2 = 18
sw_3 = 23
sw_4 = 17

Rel_1 = 24
Rel_2 = 25
Rel_3 = 26
Rel_4 = 27

msgRel_1 = ""
msgRel_2 = ""
msgRel_3 = ""
msgRel_4 = ""

GPIO.setup(Rel_1, GPIO.OUT)
GPIO.setup(Rel_2, GPIO.OUT)
GPIO.setup(Rel_3, GPIO.OUT)
GPIO.setup(Rel_4, GPIO.OUT)

GPIO.setup(sw_1, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(sw_2, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(sw_3, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(sw_4, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def sw_pressed(channel):
    if(channel==sw_1):
        GPIO.output(Rel_1, not(GPIO.input(Rel_1)))

```

โปรแกรมที่ 4-3 โค้ด Python ไฟล์ RelayToggleSW4.py สำหรับบอร์ด Raspberry Pi 3 เพื่อใช้งานขาพอร์ต GPIO 4 ขาในการรับค่าจากการกดสวิตช์ เพื่อนำมาขับโหลดกระแสไฟฟ้าสูง 4 ช่องโดยใช้บอร์ด RELAY4I

```

elif(channel==sw_2):
    GPIO.output(Rel_2,not(GPIO.input(Rel_2)))
elif(channel==sw_3):
    GPIO.output(Rel_3,not(GPIO.input(Rel_3)))
elif(channel==sw_4):
    GPIO.output(Rel_4,not(GPIO.input(Rel_4)))

GPIO.add_event_detect(sw_1,GPIO.FALLING,callback=sw_pressed,bouncetime=300)
GPIO.add_event_detect(sw_2,GPIO.FALLING,callback=sw_pressed,bouncetime=300)
GPIO.add_event_detect(sw_3,GPIO.FALLING,callback=sw_pressed,bouncetime=300)
GPIO.add_event_detect(sw_4,GPIO.FALLING,callback=sw_pressed,bouncetime=300)
while(1):
    msgRel_1="Relay1=%d" % GPIO.input(Rel_1)
    msgRel_2="Relay2=%d" % GPIO.input(Rel_2)
    msgRel_3="Relay3=%d" % GPIO.input(Rel_3)
    msgRel_4="Relay4=%d" % GPIO.input(Rel_4)

    width = disp.width
    height = disp.height
    image = Image.new('1', (width, height))
    draw = ImageDraw.Draw(image)
    font = ImageFont.truetype('VCR_OSD_MONO_1.001.ttf',14)
    draw.text((0,0),msgRel_1,font=font, fill=1)
    draw.text((0,16),msgRel_2,font=font, fill=1)
    draw.text((0,32),msgRel_3,font=font, fill=1)
    draw.text((0,48),msgRel_4,font=font, fill=1)
    disp.image(image)
    disp.display()

```

คำอธิบายโปรแกรมเพิ่มเติม

- ที่บรรทัด `GPIO.setup(Rel_1,GPIO.OUT)` ต่อเนื่อง 4 บรรทัดเป็นคำสั่งกำหนดให้มีการพูลอัปที่ขาพอร์ตทั้ง 4 ขาที่เชื่อมต่อกับสวิตช์จอยสติ๊ก นั่นคือ ขา 6, 18, 23 และ 17
- ที่บรรทัด `font = ImageFont.truetype('VCR_OSD_MONO_1.001.ttf',14)` เป็นการกำหนดให้จอแสดงผล OLED แสดงตัวอักษรด้วยขนาด 14 ซึ่งเล็กลงมา เนื่องจากมีข้อความแสดงผล 4 บรรทัด
- ที่บรรทัด `GPIO.add_event_detect(sw_1,GPIO.FALLING,callback=sw_pressed,bouncetime=300)` ใช้รักษาค่าสถานะของขาพอร์ตก่อนหน้าไว้ เมื่อมีการเปลี่ยนแปลงที่ขาพอร์ตซึ่งต่อกับสวิตช์อีกครั้ง ก็จะทำการกลับสถานะของขาพอร์ตเอาต์พุต จาก “1” เป็น “0” ทำให้วงจรนับปริมาตรที่ต่อกับขาพอร์ตแต่ละขาหยุดทำงาน

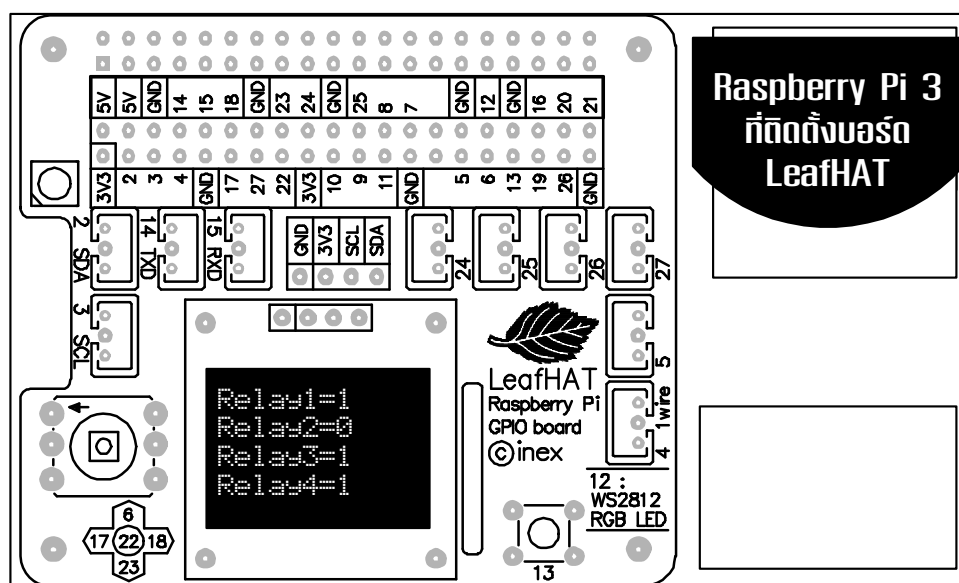
โปรแกรมที่ 4-3 โค้ด Python ไฟล์ `RelayToggleSW4.py` สำหรับบอร์ด Raspberry Pi 3 เพื่อใช้งานขาพอร์ต GPIO 4 ขาในการรับค่าจากการกดสวิตช์ เพื่อนำมาขับโหลดกระแสไฟฟ้าสูง 4 ช่องโดยใช้บอร์ด RELAY4I

(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) พิมพ์คำสั่งตามโปรแกรมที่ 4-3 แล้วบันทึกชื่อไฟล์เป็น **RelayToggleSW4.py**

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่าง Terminal แล้วสั่งรันด้วยคำสั่ง **sudo python3 RelayToggleSW4.py**

ในตัวอย่างนี้ จะใช้สวิตช์ 1 ตัวในการควบคุมการทำงานของรีเลย์ 1 ช่อง แต่จะบรรจุครบ 4 ช่องตามความสามารถของบอร์ดขับรีเลย์ โดยสวิตช์ทุกตัวจะทำงานในลักษณะกดหนึ่งครั้ง ส่งลอจิก “1” เมื่อกดอีกครั้ง จะส่งลอจิก “0” ออกไปแทน นอกจากนี้ ในขณะที่ทำงาน จอแสดงผล OLED บนบอร์ด LeafHAT จะแสดงสถานะการทำงานของรีเลย์ทั้ง 4 ช่องด้วย



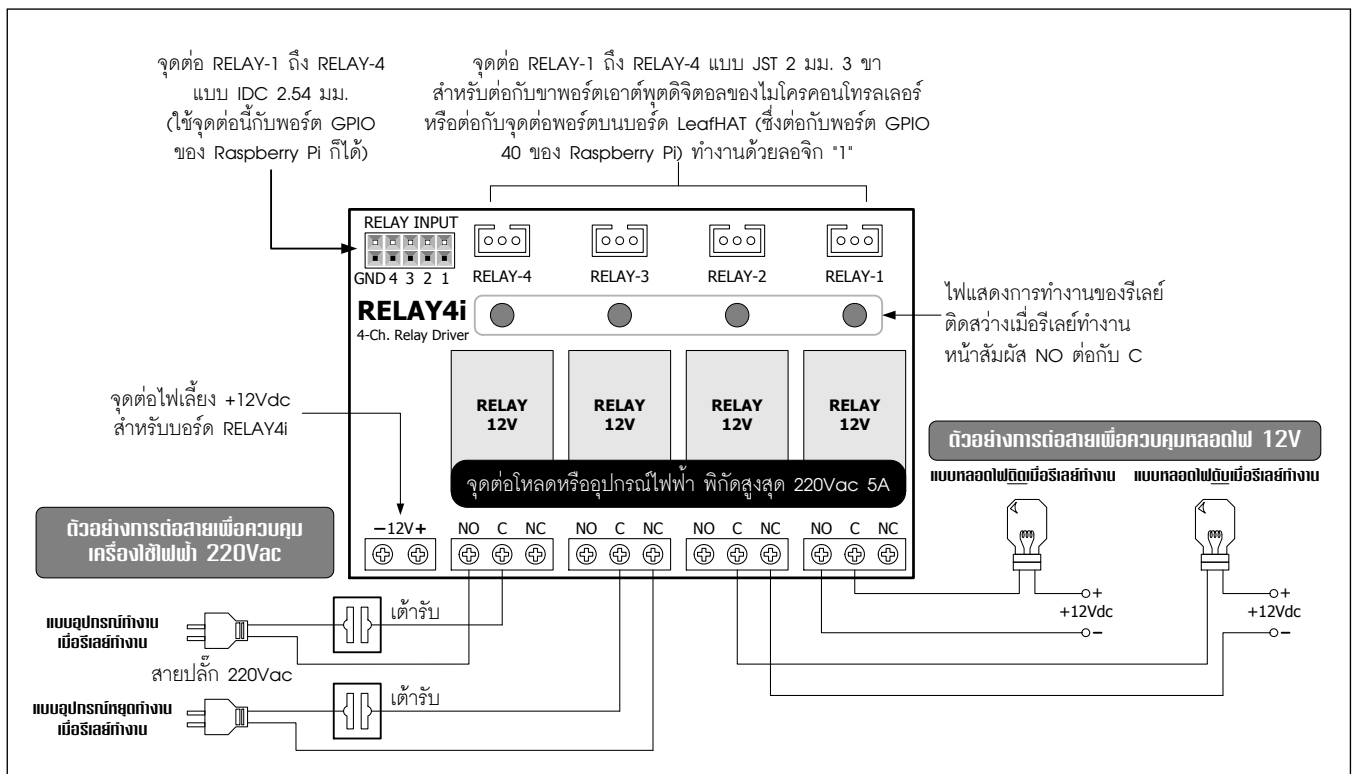
รูปที่ 4-10 แสดงสถานะการทำงานของรีเลย์จากการทำงานของโปรแกรมที่ 4-3 บนจอแสดงผล OLED ของบอร์ด LeafHAT ที่เชื่อมต่อกับบอร์ด Raspberry Pi 3 ผ่านทางพอร์ต GPIO 40 ขา

4.5 เกี่ยวกับการต่อใช้งานหน้าสัมผัสของรีเลย์แบบกลไก

เนื่องจากรีเลย์แบบกลไกมีหน้าสัมผัสสำหรับต่อกับโหลดและแรงดันไฟเลี้ยงของโหลดในแบบปกติปิดหรือ NC และแบบปกติเปิดหรือ NO ดังนั้นการใช้งานจึงมี 2 แนวทางหลักๆ คือ

1. กำหนดให้โหลดหรือเครื่องใช้ไฟฟ้าทำงานเมื่อรีเลย์ทำงาน ในแนวทางนี้จะเลือกต่อใช้งานหน้าสัมผัส NO และ C
2. กำหนดให้โหลดหรือเครื่องใช้ไฟฟ้าหยุดทำงานเมื่อรีเลย์ทำงาน ในแนวทางนี้จะเลือกต่อใช้งานหน้าสัมผัส NC และ C แทน

รูปที่ 4-11 แสดงแนวทางการใช้งานบอร์ดขับรีเลย์เพื่อขับโหลดในลักษณะต่างๆ



รูปที่ 4-11 แนวทางการใช้งานบอร์ดขับรีเลย์ RELAY4i เพื่อขับโหลดในแบบทำงานเมื่อรีเลย์ทำงานและแบบหยุดทำงานเมื่อรีเลย์ทำงาน โดยแสดงการใช้งานกับโหลดที่ใช้ไฟฟ้ากระแสตรงอย่างหลอดไฟและเครื่องใช้ไฟฟ้า 220Vac ผ่านตัวรับ

4.6 ความรู้เบื้องต้นเกี่ยวกับโซลิดสเตตรีเลย์

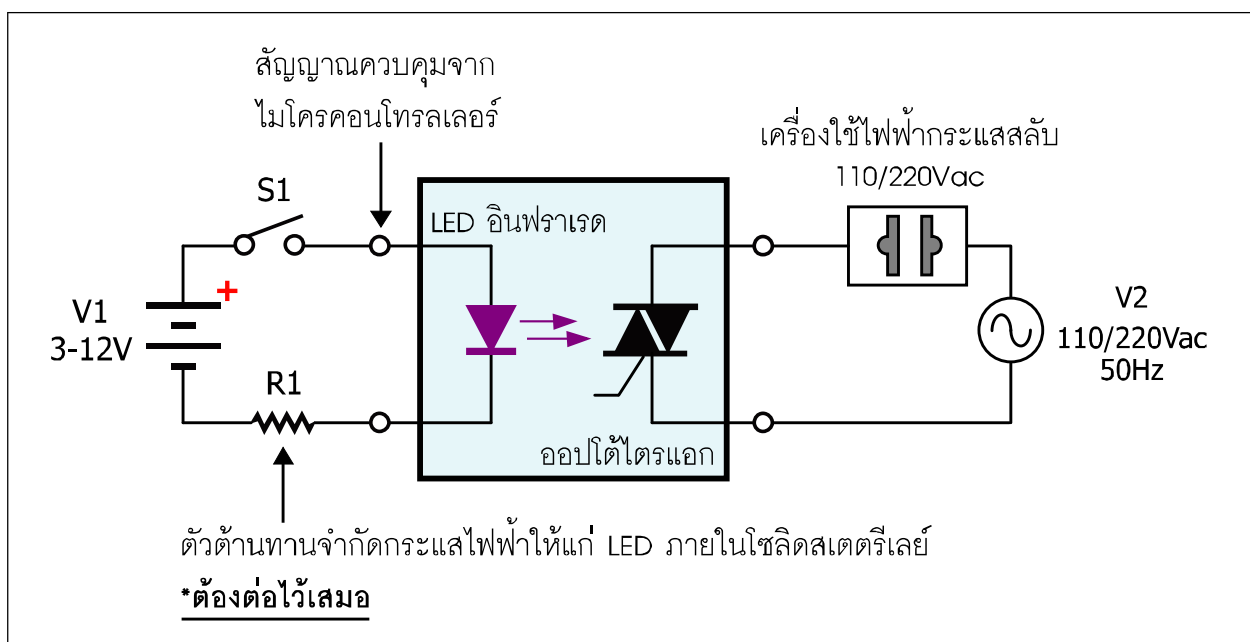
โซลิดสเตตรีเลย์ (solid-state relay) คือรีเลย์ที่ไม่มีกลไกเคลื่อนไหว ภายในจะเป็นอุปกรณ์สารกึ่งตัวนำทั้งหมด ประกอบด้วย LED อินฟราเรด และออปโตไทรแอก ดังแสดงวงจรภายในตามรูปที่ 4-12 สัญญาณควบคุมแรงดันต่ำจะถูกส่งมายัง LED อินฟราเรด เมื่อ LED ทำงานจะขับแสงอินฟราเรดไปยังออปโตไทรแอก เพื่อให้มีกระแสไฟฟ้า ที่ขาเอาต์พุตของออปโตไทรแอกก็จะมิต่อกับไฟสลับแรงดันสูง ทำให้แรงดันต่ำจากวงจรควบคุมและแรงดันไฟสูงทางฝั่งของโหลดหรืออุปกรณ์ไฟฟ้าที่นำมาต่อจะถูกแยกออกจากกันอย่างเด็ดขาด และใช้การเชื่อมโยงทางแสงในการควบคุมการทำงานแทนปกติจะพบอุปกรณ์แบบนี้ในเครื่องจักรอุตสาหกรรม แต่ปัจจุบันมีการนำมาใช้ในเครื่องใช้ไฟฟ้าในบ้าน อาทิ ในเครื่องซักผ้าอัตโนมัติ เป็นต้น

4.7 แนะนำโซลิดสเตตรีเลย์สำหรับการทดลองและใช้งานได้จริง

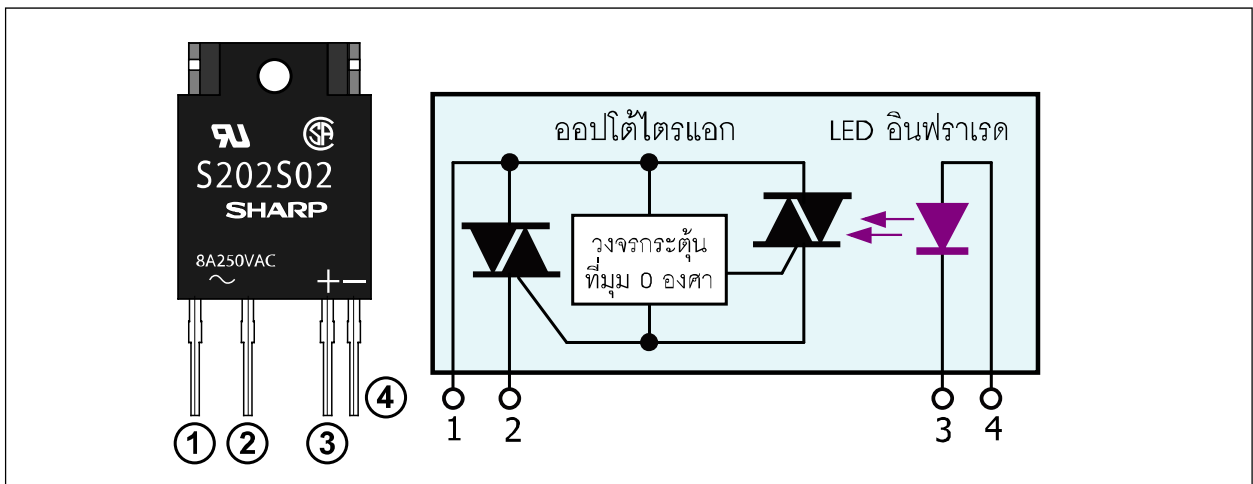
โซลิดสเตตรีเลย์ที่นำมาแนะนำในบทนี้มี 2 เบอร์ คือ S202S02 เป็นแบบอุปกรณ์เดี่ยวทางแผ่นวงจรพิมพ์ ผลิตโดย Sharp Microelectronics ประเทศญี่ปุ่น (<http://sharp-world.com>) และ SSR-25DA ซึ่งเป็นแบบโมดูลผลิตโดย Fotek Controls LTD. จากไต้หวัน (www.fotek.com.tw)

4.7.1 S202S02

มีตัวถัง, การจัดขา และไดอะแกรมการทำงานภายในแสดงดังรูปที่ 4-13 จะเห็นว่ามีแบ่งส่วนการทำงานเป็น 2 ส่วนที่แยกจากกัน นั่นคือ LED อินฟราเรดสำหรับรับสัญญาณควบคุมแรงดันต่ำ และ



รูปที่ 4-12 วงจรภายในและการทำงานของโซลิดสเตตรีเลย์



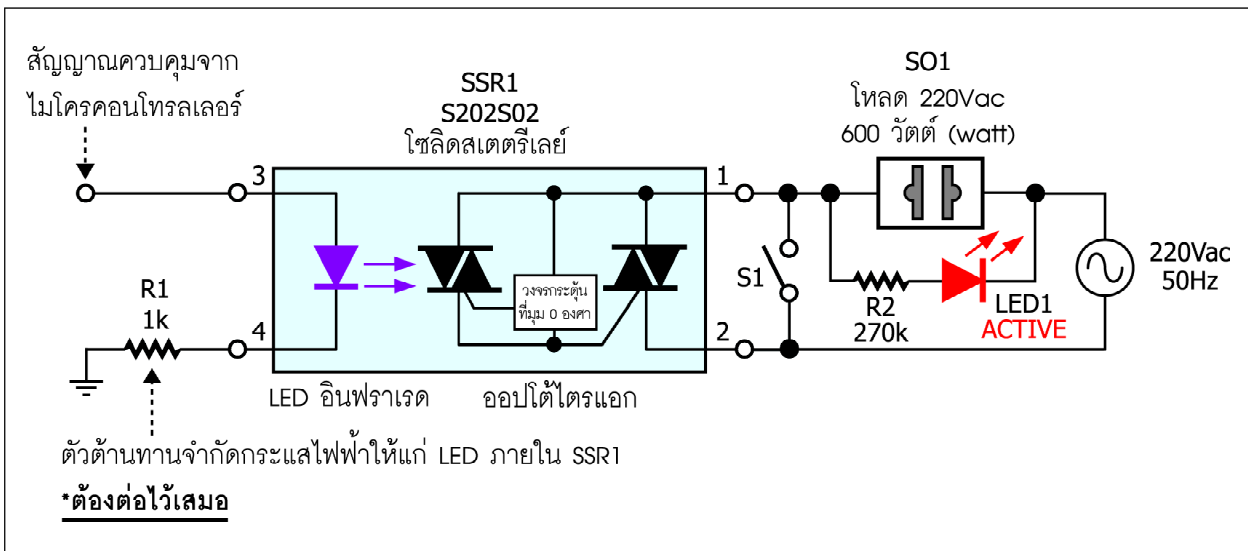
รูปที่ 4-13 ตัวถัง การจัดขา และไดอะแกรมการทำงานของโซลิดสเตตรีเลย์เบอร์ S202S20

ส่วนของวงจรออปโตไดรแอกหรือสวิตช์ไฟฟ้าแรงดันสูงที่ทำงานด้วยแสง วงจรทั้งสองส่วนจะเชื่อมโยงกันด้วยแสง ทำให้การวัดของแรงดันต่ำไม่ได้เชื่อมต่อกับวงจรไฟฟ้ากระแสสลับโดยตรง ทำให้ไม่เกิดการรบกวนหรือแรงดันกระชากย้อนกลับไปทำให้วงจรส่วนควบคุมเสียหายได้ ในส่วนของวงจรออปโตไดรแอกภายในโซลิดสเตตรีเลย์ยังมีวงจรเล็กๆ ที่เรียกว่า วงจรกระตุ้นที่มุม 0 องศาหรือ zero crossing trigger ซึ่งทำหน้าที่จัดการให้สัญญาณกระตุ้นการทำงานของไดรแอกเกิดขึ้นที่มุม 0 องศาของสัญญาณไฟสลับ เพื่อช่วยลดสัญญาณรบกวนและทำให้การนำกระแสไฟฟ้าของไดรแอกทำได้เต็มที่

คุณสมบัติทางเทคนิคที่สำคัญมีดังนี้

- อินพุตรับแรงดันไฟตรงได้ตั้งแต่ 1.2 ถึง 35Vdc ต้องต่อตัวต้านทานจำกัดกระแสไฟฟ้าเพิ่มเติม เพื่อให้เกิดแรงดันตกคร่อม LED อินฟราเรดภายในไม่เกิน 1.4V
- กระแสไฟฟ้ากระตุ้นทางอินพุตต่ำสุด 8mA
- กระแสไฟฟ้าอินพุตสำหรับการกระตุ้นให้ทำงานควรมีค่า 16 ถึง 24mA และกระแสไฟฟ้าอินพุตสำหรับหยุดการทำงานมีค่า 0 ถึง 0.1mA
- ขั้วโหลดไฟสลับได้ตั้งแต่ 80 ถึง 240Vac กระแสไฟฟ้าสูงสุด 8A ความถี่ 47 ถึง 63Hz
- มีตัวถังเป็นแบบ SIP 4 ขา ติดตั้งลงบนแผ่นวงจรพิมพ์ได้
- หากนำมาใช้ขับโหลดที่ต้องการกระแสไฟฟ้าสูงกว่า 3A จะต้องติดตั้งแผ่นระบายความร้อนที่ด้านหลังของตัวถัง

ตัวอย่างวงจรสำหรับใช้งาน S202S20 แสดงในรูปที่ 4-14 เป็นวงจรเด้ารับไฟสลับแบบควบคุมด้วยไมโครคอนโทรลเลอร์ โดยเด้ารับนี้จะทำงานเมื่อได้รับสัญญาณลอจิก “1” จากไมโครคอนโทรลเลอร์ และหยุดทำงานเมื่อได้รับสัญญาณลอจิก “0” นอกจากนั้น ยังมีสวิตช์ S1 ต่อไว้เพื่อการเปิดปิดเด้ารับแบบปกติเพื่อให้เด้ารับไฟสลับนี้ใช้งานได้เหมือนเด้ารับปกติในกรณีที่ไมใช้งานร่วมกับไมโครคอนโทรลเลอร์



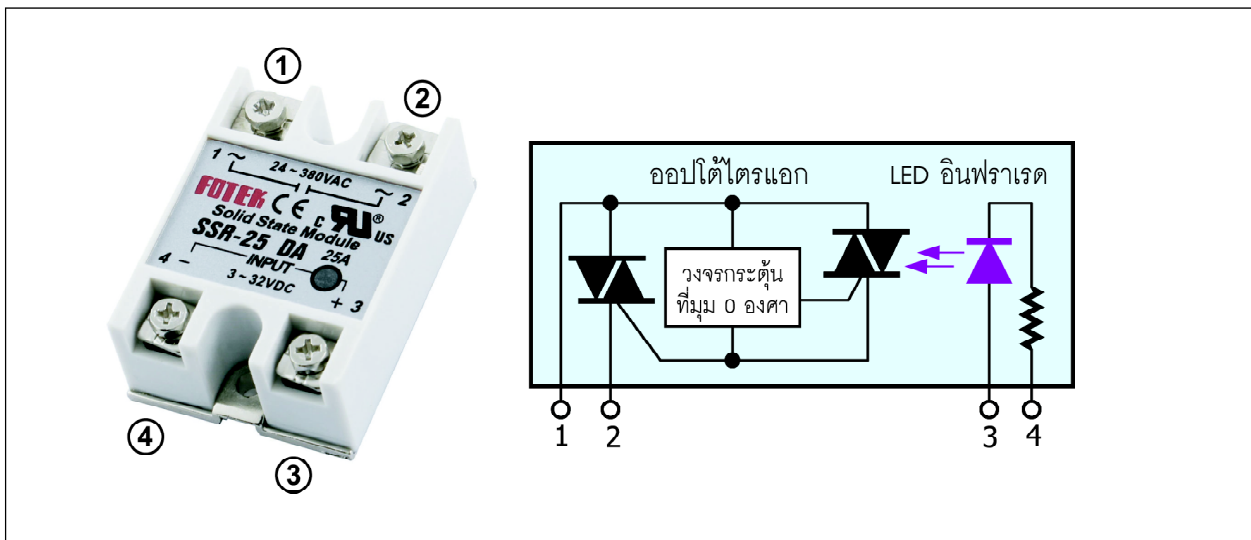
รูปที่ 4-14 ตัวอย่างวงจรใช้งานโซลิดสเตตรีเลย์ S202S20 นำมาสร้างเป็นเต้ารับไฟสลับ 220Vac แบบควบคุมได้จาก Raspberry Pi และไมโครคอนโทรลเลอร์

4.7.2 โมดูลโซลิดสเตตรีเลย์ SSR-25A

มีตัวถัง, การจัดหา และไดอะแกรมการทำงานภายในแสดงดังรูปที่ 4-15 คุณสมบัติทางเทคนิคที่สำคัญมีดังนี้

- อินพุตรับแรงดันไฟตรงได้ตั้งแต่ 3 ถึง 32Vdc ต่อตรงเข้ากับไมโครคอนโทรลเลอร์และขาพอร์ต GPIO ของ Raspberry Pi 3 ได้โดยตรง ไม่ต้องต่อตัวต้านทานจำกัดกระแสไฟฟ้าเพิ่มเติม
- แรงดันอินพุตที่กระตุ้นให้ทำงานต้องมากกว่า 2.4V และแรงดันอินพุตสำหรับหยุดทำงานต้องน้อยกว่า 1V
- กระแสไฟฟ้ากระตุ้นทางอินพุต 7.5mA
- ขั้วโหลดไฟสลับได้ตั้งแต่ 24 ถึง 380Vac กระแสไฟฟ้าสูงสุด 25A
- จุดต่อสัญญาณอินพุตและเอาต์พุตเป็นแบบขันสกรู
- หากขั้วโหลดที่ต้องการกระแสไฟฟ้าสูงกว่า 5A จะต้องติดตั้งแผ่นระบายความร้อนที่ด้านหลังของตัวถัง

จากรูปที่ 4-15 จะเห็นว่า SSR-25DA มีการทำงานคล้ายกับ S202S20 จุดที่แตกต่างคือ การรับสัญญาณทางอินพุต SSR-25DA สามารถรับสัญญาณกระตุ้นให้ทำงานได้โดยตรง เนื่องจากภายในมีตัวต้านทานต่ออนุกรมเพื่อจำกัดกระแสไฟฟ้าให้แก่ LED อินฟราเรดไว้แล้ว



รูปที่ 4-15 ตัวถัง, การจัดขา และไดอะแกรมการทำงานของโซลิดสเตตรีเลย์เบอร์ SSR-25DA

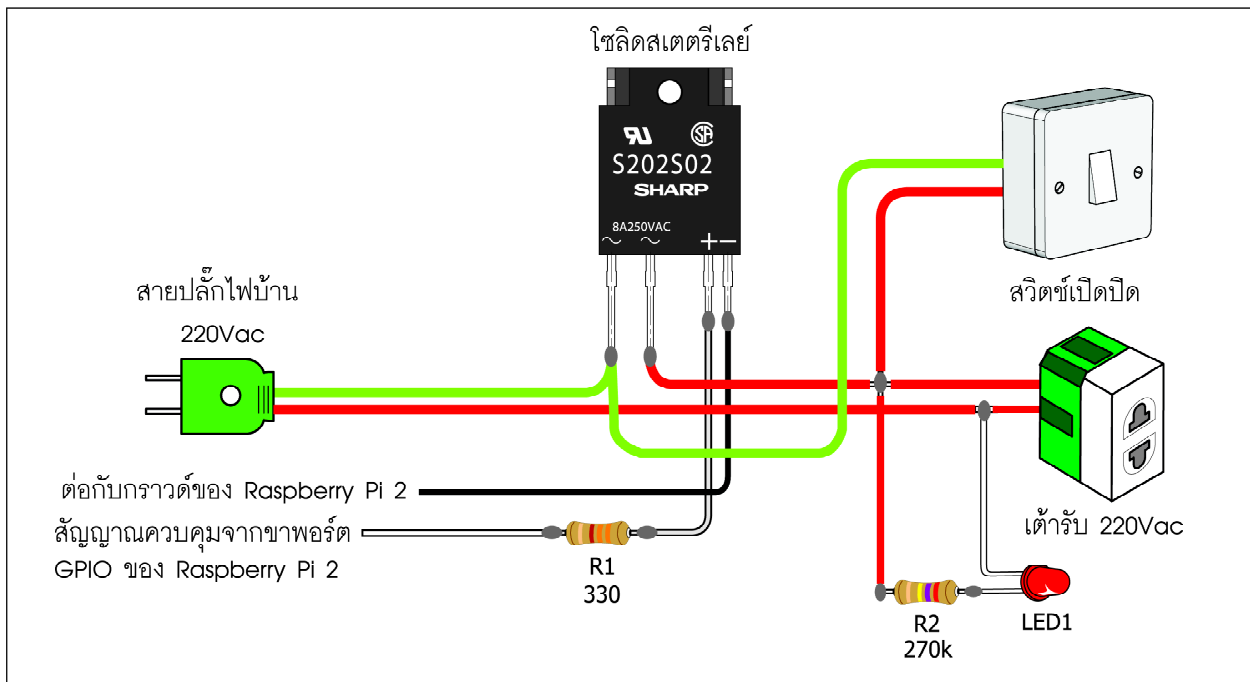
4.7 แนวทางการใช้งาน Raspberry Pi 3 กับโซลิดสเตตรีเลย์

โซลิดสเตตรีเลย์ที่แนะนำในบทนี้ทั้งสองรุ่นเมื่อนำมาใช้งานกับบอร์ด Raspberry Pi 3 จะมีแนวทางใกล้เคียงกัน ความแตกต่างคือ ลักษณะหรือวิธีการเชื่อมต่อ ทั้งนี้เนื่องจากอุปกรณ์ทั้งสองรุ่นมีตัวถังที่แตกต่างกันนั่นเอง ในรูปที่ 4-16 เป็นวงจรตัวอย่างใช้งานบอร์ด Raspberry Pi 3 กับโซลิดสเตตรีเลย์ S202S20 ส่วนในรูปที่ 4-17 เป็นการต่อใช้งานบอร์ด Raspberry Pi 3 กับโมดูลโซลิดสเตตรีเลย์ SSR-25DA โดยทั้งสองตัวอย่างเป็นการนำโซลิดสเตตรีเลย์มาสร้างเป็นเด้าร์บไฟสลับแบบควบคุมได้โดยใช้บอร์ด Raspberry Pi 3

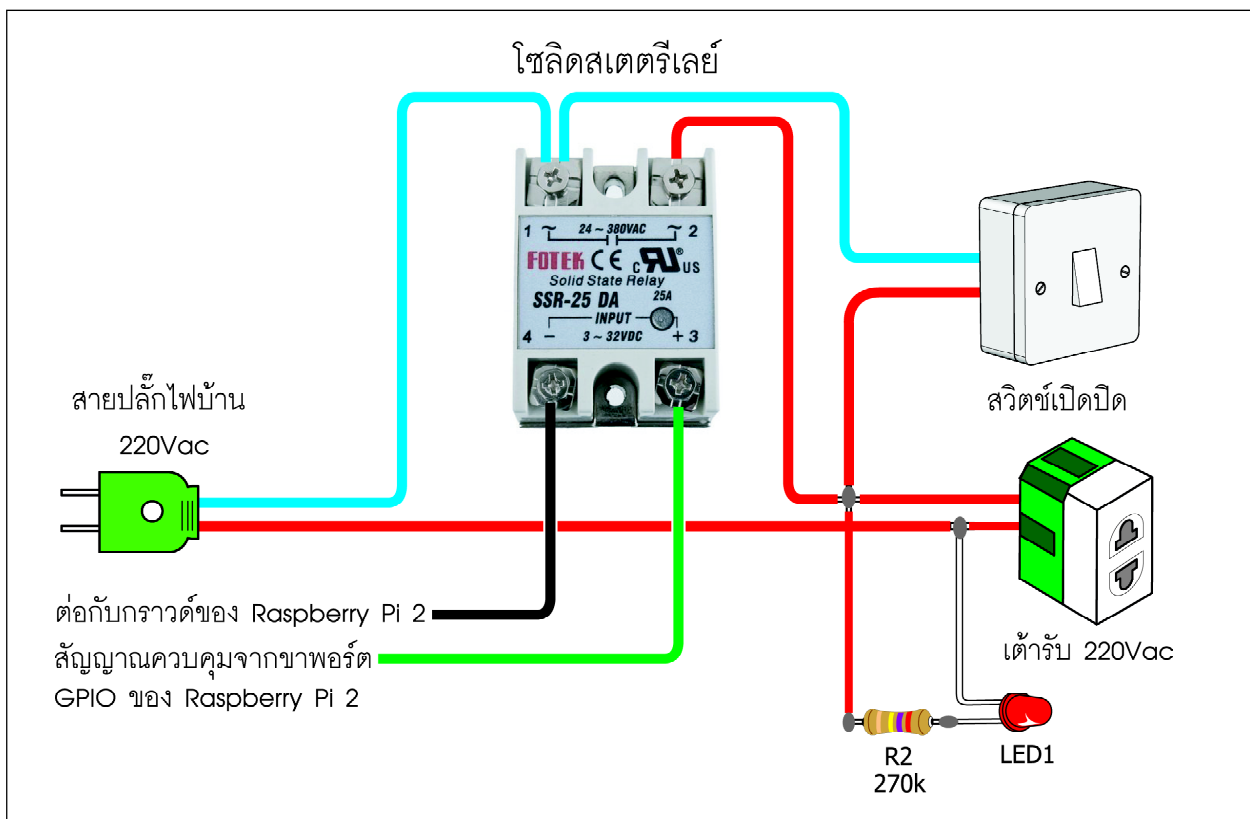
ในด้านการเขียนโปรแกรมควบคุม ทำได้ง่ายๆ โดยมีขั้นตอนดังนี้

1. กำหนดให้ขาพอร์ต GPIO ที่ใช้งานเป็นเอาต์พุต
2. เขียนข้อมูล “1” ไปยังขาพอร์ต GPIO ที่เชื่อมต่อกับอินพุตของโซลิดสเตตรีเลย์หากต้องการให้โซลิดสเตตรีเลย์ทำงาน
3. เขียนข้อมูล “0” ไปยังขาพอร์ต GPIO ที่เชื่อมต่อกับอินพุตของโซลิดสเตตรีเลย์หากต้องการให้โซลิดสเตตรีเลย์หยุดทำงาน

โดยการเขียนโปรแกรมเพื่อส่งข้อมูลไปยังขาพอร์ต GPIO ดูได้จากตัวอย่างโปรแกรมในบทที่ 2 ก่อนหน้านี้



รูปที่ 4-16 ตัวอย่างการเชื่อมต่อเพื่อใช้งานบอร์ด Raspberry Pi กับไซลิตสเตตรีเลย์ S202S20



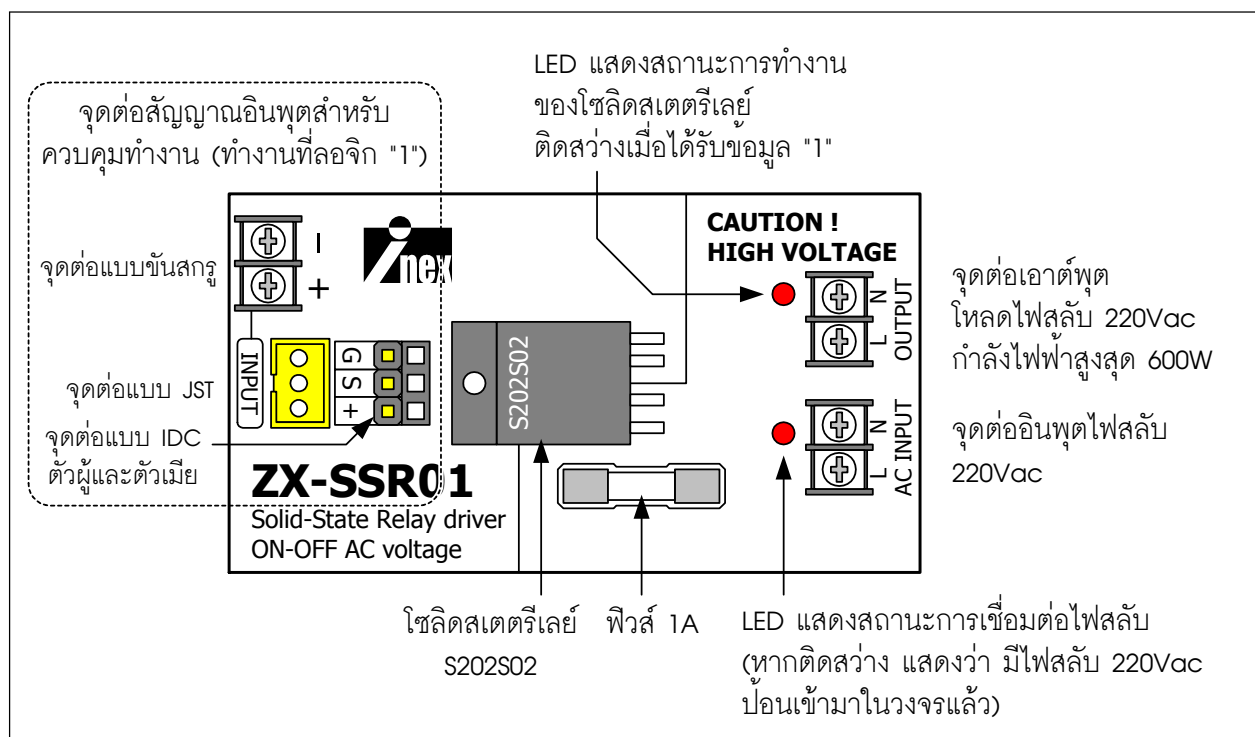
รูปที่ 4-17 ตัวอย่างการเชื่อมต่อเพื่อใช้งานบอร์ด Raspberry Pi กับโมดูลไซลิตสเตตรีเลย์ SSR-25DA

4.8 แนะนำบอร์ดโซลิดสเตตรีเลย์สำหรับการทดลองและใช้งานได้จริง

บอร์ดโซลิดสเตตรีเลย์ที่นำมาแนะนำในบทนี้คือ ZX-SSR01 ดังแสดงหน้าตาและส่วนประกอบต่างๆ ในรูปที่ 4-18 โดยบอร์ดขับโซลิดสเตตรีเลย์ ZX-SSR01 ใช้โซลิดสเตตรีเลย์เบอร์ S202S02 ผลิตโดย Sharp Microelectronics ประเทศญี่ปุ่น (<http://sharp-world.com>) เป็นอุปกรณ์หลัก มีข้อมูลทางเทคนิคที่สำคัญดังนี้

คุณสมบัติทางเทคนิคที่สำคัญมีดังนี้

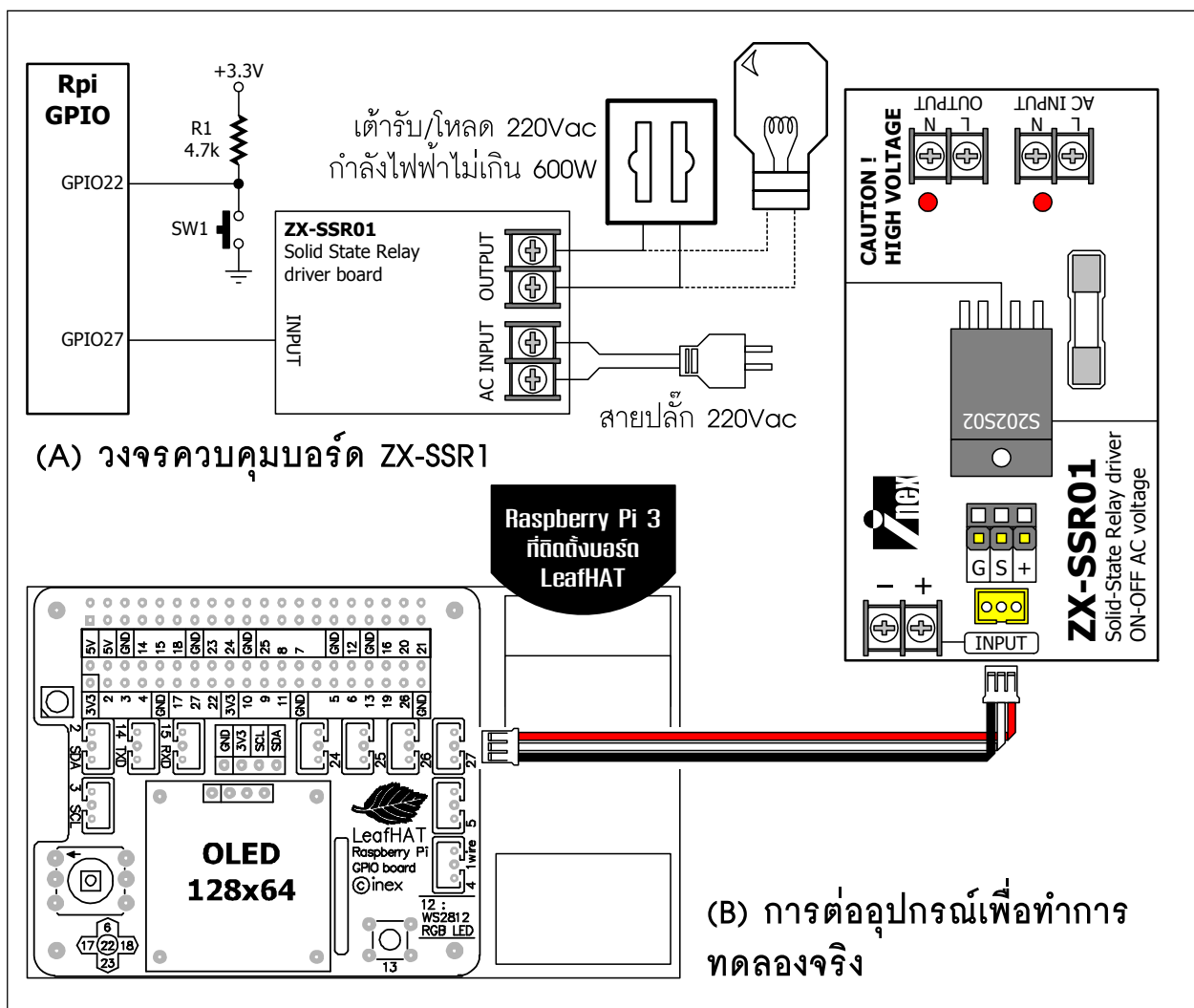
- อินพุตรับแรงดันไฟตรงได้ตั้งแต่ 1.2 ถึง 35Vdc ต้องต่อตัวต้านทานจำกัดกระแสไฟฟ้าเพิ่มเติม เพื่อให้เกิดแรงดันตกคร่อม LED อินพุตภายในไม่เกิน 1.4V
- กระแสไฟฟ้ากระตุ้นทางอินพุตต่ำสุด 8mA
- กระแสไฟฟ้าอินพุตสำหรับการกระตุ้นให้ทำงานควรมีค่า 16 ถึง 24mA และกระแสไฟฟ้าอินพุตสำหรับหยุดการทำงานมีค่า 0 ถึง 0.1mA
- ขั้วโหดไฟสลับได้ตั้งแต่ 80 ถึง 240Vac กระแสไฟฟ้าสูงสุด 8A ความถี่ 47 ถึง 63Hz
- มีตัวถังเป็นแบบ SIP 4 ขา ติดตั้งลงบนแผ่นวงจรพิมพ์ได้
- หากนำมาใช้ขับโหนดที่ต้องการกระแสไฟฟ้าสูงกว่า 3A จะต้องติดตั้งแผ่นระบายความร้อนที่ด้านหลังของตัวถัง



รูปที่ 4-18 แสดงส่วนประกอบสำคัญของบอร์ดขับโซลิดสเตตรีเลย์ ZX-SSR01

4.9 ตัวอย่างการทดลองใช้งาน Raspberry Pi เพื่อขับโหลดกระแสไฟฟ้าสูงด้วย ZX-SSR1 บอร์ดขับโซลิดสเตตรีเลย์

ตัวอย่างวงจรสำหรับใช้งาน Raspberry Pi 3 กับบอร์ดขับโซลิดสเตตรีเลย์ ZX-SSR01 แสดงในรูปที่ 4-19 อาจกล่าวได้ว่า นี่คืวงจรได้รับไฟสลับแบบควบคุมด้วยบอร์ดคอมพิวเตอร์ โดยได้รับนี้จะทำงานเมื่อได้รับสัญญาณลอจิก “1” จากพอร์ต GPIO ของ Raspberry Pi และหยุดทำงานเมื่อได้รับสัญญาณลอจิก “0”



รูปที่ 4-19 วงจรทดลองใช้งาน Raspberry Pi 3 เพื่อควบคุมบอร์ดขับโซลิดสเตตรีเลย์ ZX-SSR01 ผ่านทางบอร์ด LeafHAT ที่ต่อกับพอร์ต GPIO 40 ขาของ Raspberry Pi 3 เป็นแนวทางในการพัฒนาต่อเป็นวงจรได้รับไฟสลับ 220Vac ควบคุมด้วยบอร์ดคอมพิวเตอร์ และสามารถเพิ่มจำนวนช่องของการขับโหลดได้ด้วยการเพิ่มจำนวนบอร์ดขับโซลิดสเตตรีเลย์

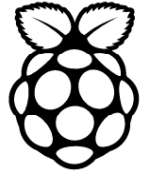
ในการเขียนโปรแกรมควบคุม ทำได้ง่ายๆ โดยมีขั้นตอนดังนี้

- (1) กำหนดให้ขาพอร์ต GPIO ที่ใช้งานเป็นเอาต์พุต
- (2) เขียนข้อมูล “1” ไปยังขาพอร์ตที่ต่อกับอินพุตของ ZX-SSR01 หากต้องการให้โซลิดสเตตรีเลย์ทำงาน
- (3) เขียนข้อมูล “0” ไปยังขาพอร์ตที่ต่อกับอินพุตของ ZX-SSR01 หากต้องการให้โซลิดสเตตรีเลย์หยุดทำงาน

สำหรับโปรแกรมควบคุมการทำงานที่พัฒนาด้วยภาษา Python นำโปรแกรมที่ 4-2 ในบทนี้มาใช้ได้ โดยสวิทช์ที่ใช้ควบคุมคือ ปุ่มกลางของสวิทช์จอยสติ๊กบนบอร์ด LeafHAT ซึ่งต่อกับขาพอร์ต GPIO22 ส่วนขาควบคุมบอร์ดขับโซลิดสเตตรีเลย์ ZX-SSR01 คือ ขาพอร์ต GPIO27



บทที่ 10

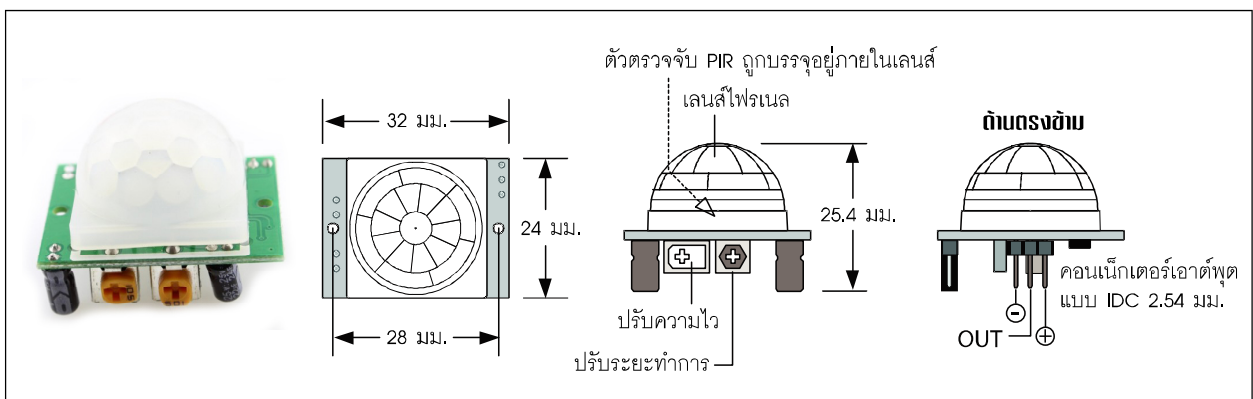
Raspberry Pi 3 กับการติดต่อ
ตัวตรวจจับสภาพแวดล้อม

การเรียนรู้และใช้งานระบบสมองกลฝังตัวสมัยใหม่ จำเป็นอย่างยิ่งที่ต้องรู้จักและใช้งานอุปกรณ์ต่อพ่วงที่มีความสามารถแตกต่างออกไป โดยเฉพาะอย่างยิ่งตัวตรวจจับหรือเซนเซอร์ (sensor) ในแบบต่างๆ และยิ่งเมื่อตัวควบคุมหลักมีความสามารถสูงขึ้นมากอย่างบอร์ดคอมพิวเตอร์ Raspberry Pi 3 ทำให้การใช้ประโยชน์จากตัวตรวจจับทำได้มีประสิทธิภาพมากขึ้น กอปรกับการพัฒนาอย่างต่อเนื่องของผู้ผลิต ทำให้ผู้เริ่มต้นเรียนรู้ได้ใช้งานตัวตรวจจับปริมาณทางกายภาพแบบใหม่ๆ ในราคาที่เข้าถึงได้

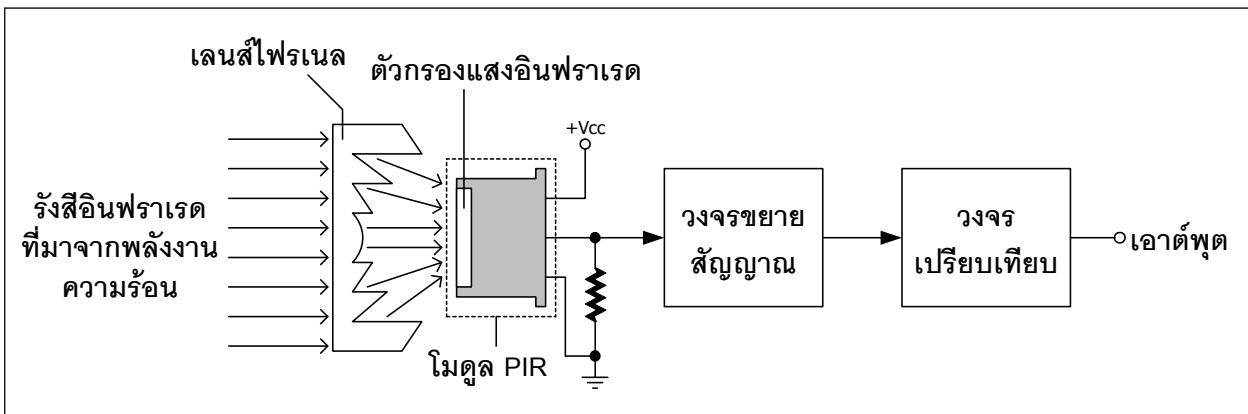
ในบทนี้จะได้นำมาถึงการใช้งานบอร์ด Raspberry Pi 3 กับตัวตรวจจับสภาพแวดล้อมที่น่าสนใจ 3 ตัวคือ **PIR** หรือโมดูลตรวจจับความเคลื่อนไหว, **DHT11** โมดูลตรวจจับความชื้นและอุณหภูมิ และ **BH1750** โมดูลวัดความเข้มแสงที่ให้ผลการทำงานเป็นหน่วยลักซ์

5.1 ใช้งานโมดูลตรวจจับความเคลื่อนไหว

อุปกรณ์ตรวจจับความเคลื่อนไหวที่นำมาเสนอเพื่อเป็นตัวอย่างในที่นี้คือ โมดูล **ZX-PIR2.0** ซึ่งใช้ตัวตรวจจับที่เรียกว่า **PIR (Passive Infrared)** ซึ่งสามารถตรวจจับการแผ่รังสีอินฟราเรด โดยทำงานร่วมกับเลนส์ชนิดพิเศษที่เรียกว่า **เลนส์ไฟรเนลหรือเฟรสเนล (fresnel lens)** ซึ่งทำหน้าที่รวมรังสีอินฟราเรดที่ตัวตรวจจับได้รับ เพื่อส่งผ่านไปยังตัวตรวจจับ PIR เพื่อทำการประมวลผลต่อไป



รูปที่ 5-1 แสดงลักษณะทางกายภาพของ ZX-PIR2.0 โมดูลตรวจจับความเคลื่อนไหว



รูปที่ 5-2 ไดอะแกรมการทำงานของตัวตรวจจับการแผ่รังสีอินฟราเรดซึ่งใช้ตรวจจับความเคลื่อนไหว

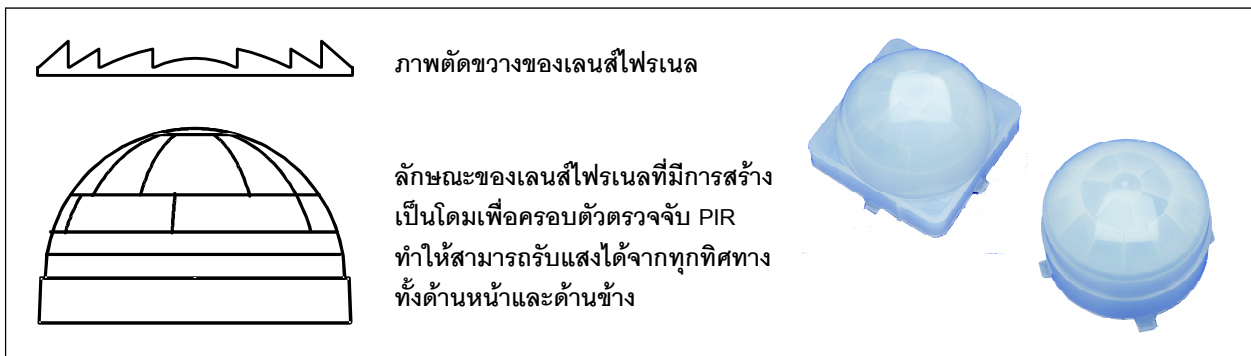
5.1.1 หลักการตรวจจับความเคลื่อนไหว

ตัวตรวจจับความเคลื่อนไหวของสิ่งมีชีวิต (motion sensor) ที่ได้รับความนิยมและใช้งานง่ายคือ ตัวตรวจจับแบบอินฟราเรด ใช้หลักการตรวจจับที่เรียกว่า ไพโรอิเล็กทริก (pyro-electric) อันเป็นการตรวจจับการแผ่รังสีอินฟราเรด หากระดับของการแผ่รังสีไม่เปลี่ยนแปลง แสดงว่า สิ่งมีชีวิตที่ต้องการตรวจจับนั้น ไม่มีการเคลื่อนไหว แต่ถ้าหากมีการเคลื่อนไหวเกิดขึ้น ระดับของการแผ่รังสีอินฟราเรดจะเปลี่ยนแปลง

สิ่งมีชีวิตไม่ว่าจะเป็นมนุษย์หรือสัตว์เลี้ยงลูกในภาวะที่ยังมีชีวิตอยู่จะกระจายพลังงานความร้อนออกมาจากตัวเองในรูปของการแผ่รังสีอินฟราเรดตลอดเวลา โดยจะมีปริมาณมากหรือน้อยขึ้นอยู่กับสภาพของร่างกายในขณะนั้น เมื่อมีการเคลื่อนไหว ปริมาณของการแผ่รังสีก็จะเปลี่ยนแปลง รังสีอินฟราเรดจากมนุษย์หรือสัตว์เลี้ยงลูกที่มึระดับความเข้มสูงสุดจะมีความยาวคลื่นประมาณ 9.4 ไมโครเมตร

ในรูปที่ 5-2 เป็นไดอะแกรมแสดงหลักการทำงานพื้นฐานของตัวตรวจจับ PIR พลังงานความร้อนจากมนุษย์หรือสัตว์เลี้ยงลูกทำให้เกิดการแผ่รังสีอินฟราเรดขึ้น รังสีจะถูกรวมหรือโฟกัสไปยังตัวตรวจจับหลักโดยใช้เลนส์แบบพิเศษที่เรียกว่า เลนส์เฟรเนล (Fresnel lens) จากนั้นตัวตรวจจับหลักจะทำการขยายสัญญาณแล้วส่งไปยังวงจรเปรียบเทียบเพื่อสร้างสัญญาณเอาต์พุตต่อไป

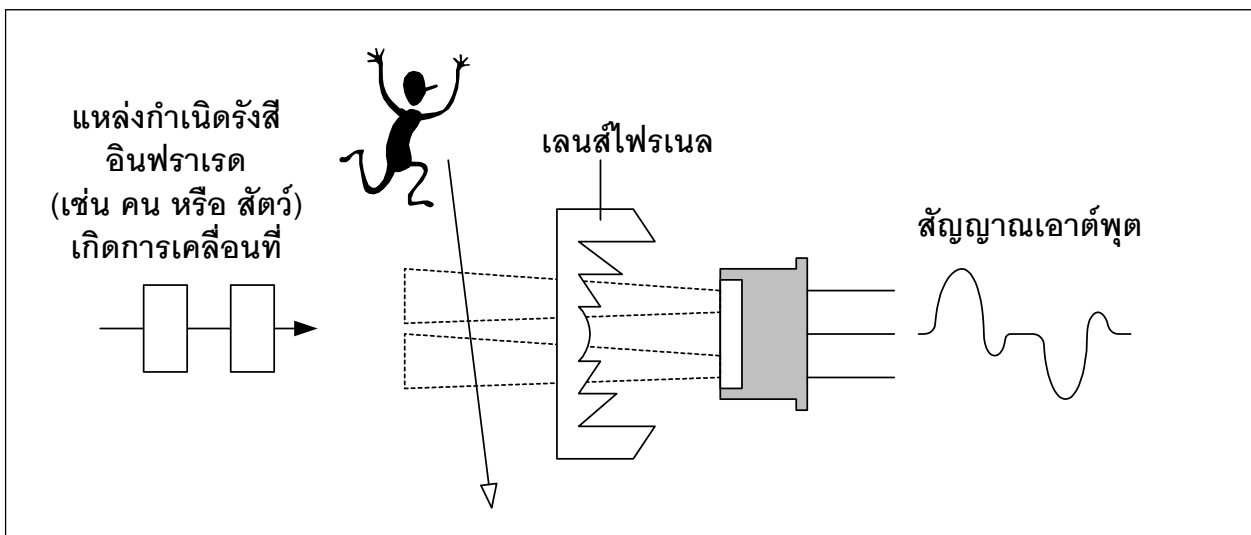
เลนส์ไฟรเนลเป็นเลนส์แบบพิเศษที่ได้รับการค้นคิดจากนักฟิสิกส์ชาวฝรั่งเศสชื่อ *ออกัสติน* ของ *ไฟรเนล (Augustin-Jean Fresnel)* โดยแนวคิดของเลนส์แบบนี้คือ เป็นเลนส์แบบขั้นบันไดที่ยอมให้แสงผ่านได้มากและจากทุกทิศทาง ดังมีโครงสร้างตามรูปที่ 53 ทั้งนี้เนื่องจากตัวเลนส์ได้ถูกสร้างขึ้นโดยลดเนื้อวัสดุในส่วนที่ไม่มีผลกับการหักเหของแสงลงไป ทำให้สามารถทำเลนส์ขนาดใหญ่ที่มีน้ำหนักเบาได้ เดิมทีเลนส์ไฟรเนลนี้ได้รับการออกแบบเพื่อให้นำมาใช้ในการกระจายในประภาคาร เพื่อให้สามารถมองเห็นประภาคารได้จากระยะไกล ต่อมาได้มีการพัฒนาให้มีขนาดเล็กลง แล้วนำมาครอบหลอดไฟเพื่อทำเป็นตะเกียง ทำให้ตะเกียงสามารถส่องแสงได้สว่างและมองเห็นได้จากระยะไกล



รูปที่ 5-4 แสดงโครงสร้างและหน้าตาของเลนส์ไฟรเนลซึ่งนำมาใช้ในโมดูล PIR

แต่ในโมดูล ZX-PIR นำเลนส์ไฟรเนลมาใช้งานในลักษณะกลับกันคือ ใช้เลนส์ไฟรเนลในการรวมแสงเข้ามาจากทุกทิศทางเพื่อโฟกัสลงไปยังส่วนตรวจจับแสงอินฟราเรดของตัวตรวจจับ PIR เพื่อให้การตรวจจับการเปลี่ยนแปลงของรังสีอินฟราเรดมีความไวสูง

ในรูปที่ 5-5 แสดงสถานการณ์ที่แหล่งกำเนิดรังสีอินฟราเรด (อาจเป็นมนุษย์หรือสัตว์เลื้อยคลาน) เกิดการเคลื่อนไหวภายในระยะทำการของตัวตรวจจับ จะทำให้ตัวตรวจจับ PIR ตรวจจับพบการแผ่รังสีอินฟราเรดที่แตกต่างกัน จึงให้สัญญาณเอาต์พุตเป็นลอจิกสูง (high) อยู่ชั่วขณะเมื่อตรวจจับพบการเคลื่อนไหว จากนั้นกลับมาเป็นลอจิกต่ำ (low) จนกว่าจะตรวจจับพบการเปลี่ยนแปลงของระดับรังสีอินฟราเรดอีกครั้ง



รูปที่ 5-5 แสดงการทำงานของโมดูล PIR เมื่อนำมาใช้ในการตรวจจับความเคลื่อนไหว

5.1.2 คุณสมบัติทางเทคนิคของโมดูล ZX-PIR2.0

ZX-PIR2.0 เป็นโมดูลตรวจจับความเคลื่อนไหวที่ติดตั้งเลนส์ไฟเรเนลครอบตัวโมดูล PIR มีวงจรขยายสัญญาณและวงจรเปรียบเทียบรวมกันไว้บนแผงวงจรเดียวกัน คุณสมบัติทางเทคนิคมีดังนี้

- ระยะเวลาตรวจจับสูงสุด 20 พุ่ต
- เมื่อตรวจพบความเคลื่อนไหวจะให้แรงดันเอาต์พุตที่สภาวะสูงที่ขาเอาต์พุต
- ใช้เวลาในการปรับตัว 10 ถึง 60 วินาทีหลังจากได้รับไฟเลี้ยง
- ใช้ไฟเลี้ยงในย่าน +3.3 ถึง +5V กระแสไฟฟ้าน้อยกว่า 100 mA

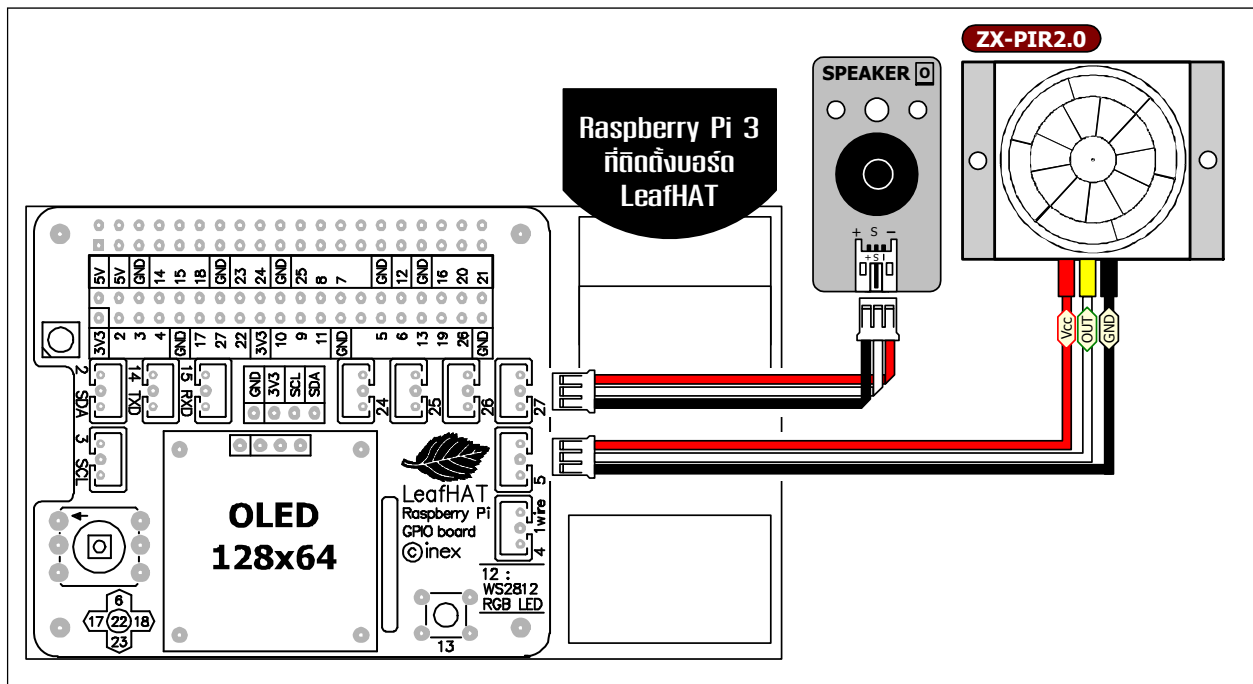
5.1.3 การใช้งานโมดูล ZX-PIR

จากรูปที่ 5-1 ซึ่งแสดงลักษณะทางกายภาพของโมดูล ZX-PIR2.0 และขาต่อใช้งาน โดยมีข้อมูลเพิ่มเติมดังนี้

ตำแหน่งขา	ชื่อขา	คำอธิบาย
—	GND	ต่อกับกราวด์
+	Vcc	ต่อกับไฟเลี้ยง +3V ถึง +5V
OUT	Output	เป็นขาเอาต์พุตของโมดูล ใช้ต่อกับไมโครคอนโทรลเลอร์หรืออินพุตของวงจรจับโหนดใดๆ

5.1.4. การเชื่อมต่อ ZX-PIR2.0 กับ Raspberry Pi 3

เนื่องจากเอาต์พุตของ ZX-PIR2.0 เป็นสัญญาณดิจิทัลที่มีสองสถานะคือ ลอจิกสูง หรือ “1” และลอจิกต่ำหรือ “0” จึงสามารถเชื่อมต่อกับขาพอร์ตดิจิทัลของไมโครคอนโทรลเลอร์และพอร์ต GPIO ของบอร์ด Raspberry Pi 3 ได้ทุกขา โดยต้องกำหนดให้ขาพอร์ตที่เชื่อมต่อนั้นเป็นอินพุตดิจิทัลก่อน และไม่ต้องต่อตัวต้านทานพูลอัพที่ขาพอร์ต แต่อย่างใด



รูปที่ 5-6 วงจรและตัวอย่างการเชื่อมต่อบอร์ด Raspberry Pi 3 กับโมดูล ZX-PIR2.0 เพื่อสร้างระบบตรวจจับความเคลื่อนไหวของมนุษย์โดยมีการแจ้งเตือนด้วยเสียงสัญญาณผ่านลำโพงเปียโซ

5.1.5 ทดลองอ่านค่าการตรวจจับความเคลื่อนไหวของโมดูล ZX-PIR2.0

ขั้นตอนการทดลองและทดสอบมีดังนี้

- (1) ต่อยังตามรูปที่ 5-6
- (2) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์
- (3) เขียนคำสั่งตามโปรแกรมที่ 5-1 บันทึกไฟล์เป็น MovementAlarm.py
- (4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง `sudo python3 MovementAlarm.py` เพื่อรันโปรแกรม
- (5) เลือกจัมเปอร์ของโมดูล ZX-PIR2.0 ไว้ที่ตำแหน่ง H

เมื่อตัวตรวจจับตรวจพบความเคลื่อนไหว บอร์ด Raspberry Pi 3 จะสร้างสัญญาณ PWM ส่งออกมาทางขา GPIO24 เพื่อขับให้ไต้ยีนผ่านลำโพงเปียโซ

```

import RPi.GPIO as GPIO
import time
from datetime import datetime
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(27,GPIO.OUT)
blink = GPIO.PWM(27,500)
blink.start(0)
GPIO.setup(5,GPIO.IN)
st=0

while (1):
    da=datetime.now()
    microsec=da.microsecond
    if microsec > 700000:
        blink.ChangeDutyCycle(st)
    else:
        blink.ChangeDutyCycle(0)

    state=GPIO.input(5)

    if (state==1):
        st=50
    else:
        st=0

```

โปรแกรมที่ 5-1 โค้ด Python ไฟล์ MovementAlarm.py สำหรับบอร์ด Raspberry Pi 3 ติดต่อกับโมดูล ZX-PIR2.0 เพื่อตรวจจับความเคลื่อนไหว หากตรวจพบ จะขับสัญญาณเสียงเตือนออกมาทางลำโพงเปียโซ

5.2 ใช้งาน ZX-DHT11 บอร์ดวัดความชื้นสัมพัทธ์และอุณหภูมิ

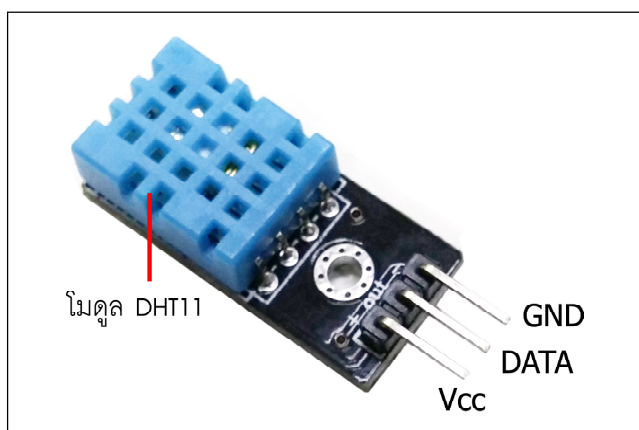
ZX-DHT11 เป็นแผงวงจรขนาดเล็กที่บรรจุโมดูลตรวจจับและวัดความชื้นสัมพัทธ์เบอร์ DHT11 ซึ่งนอกจากจะวัดความชื้นสัมพัทธ์ได้แล้ว ยังให้ค่าของอุณหภูมิของพื้นที่ที่ตรวจวัดความชื้นด้วย การติดต่อเป็นแบบหนึ่งสาย นั่นคือใช้ขาพอร์ตของไมโครคอนโทรลเลอร์หรือ GPIO ของ Raspberry Pi เพียงหนึ่งขาในการทำงาน

5.2.1 คุณสมบัติทางเทคนิคของ ZX-DHT11

ZX-DHT มีคุณสมบัติทางเทคนิคที่ควรทราบเพื่อเป็นข้อมูลประกอบในการใช้งานดังนี้

- ใช้โมดูล DHT11 ติดตั้งบนแผ่นวงจรพิมพ์
- มีตัวต้านทานต่อพูลอัพที่ขา DATA ทำให้เชื่อมต่อกับขาพอร์ตของไมโครคอนโทรลเลอร์หรือ GPIO ของ Raspberry Pi 3 ได้โดยไม่ต้องต่อตัวต้านทานเพิ่ม
- ใช้ไฟเลี้ยง +3 ถึง +5.5V ต้องการกระแสไฟฟ้า 2.5mA ในขณะทำการวัดค่า และ 0.5mA ในโหมดสลีป
- วัดความชื้นสัมพัทธ์ได้ 20 ถึง 80%RH มีความผิดพลาด $\pm 5\%$ RH และมีความละเอียดในการวัด 1 % ขนาดของข้อมูล 8 บิต
- วัดอุณหภูมิได้ 0 ถึง 50 องศาเซลเซียส มีความผิดพลาด ± 2 องศาเซลเซียส ความละเอียดในการวัด 1 องศาเซลเซียส ขนาดของข้อมูล 8 บิต
- อัตราการสุ่มวัด 1 วินาที
- ความเร็วในการตอบสนองต่อการเปลี่ยนแปลงในการวัด 6 ถึง 30 วินาที
- ขนาด 12 x 28 มิลลิเมตร

รูปที่ 5-7 แสดงหน้าตาของ ZX-DHT11 และการจัดขา



รูปที่ 5- 7 หน้าตาและการจัดขาของ ZX-DHT11
แผงวงจรวัดความชื้นสัมพัทธ์และอุณหภูมิ
หมายเหตุ : มีผู้ผลิตแผงวงจรตรวจจับที่ใช้โมดูล DHT11 หลายราย อาจมีการจัดขาที่ต่างไปจากนี้ ดังนั้น จึงควรตรวจสอบตำแหน่งขาให้ถูกต้องก่อนเชื่อมต่อเพื่อใช้งาน

5.2.2 การติดต่อกับ DHT11

โมดูล DHT11 ติดต่อกับ Raspberry Pi หรือไมโครคอนโทรลเลอร์ด้วยสายสัญญาณเพียง 1 เส้น บางครั้งเรียกการติดต่อแบบนี้ว่า การสื่อสารแบบบัสเดี่ยว (single bus communication) ดังนั้นจึงต้องมีการกำหนดจังหวะระหว่าง Raspberry Pi หรือไมโครคอนโทรลเลอร์กับ DHT11 ให้เหมาะสม (synchronization) จึงจะอ่านและเขียนค่ากับ DHT11 ได้อย่างถูกต้อง

5.2.2.1 รูปแบบข้อมูลของ DHT11

ข้อมูลใน 1 รอบของการติดต่อกับมีทั้งสิ้น 40 บิต และ DHT11 จะส่งข้อมูลบิตนี้สำคัญสูงสุด ออกมาก่อนเป็นบิตแรก รูปแบบของข้อมูลเป็นดังนี้

บิตที่	รายละเอียด	ขนาด
1	ข้อมูลจำนวนเต็มของความชื้นสัมพัทธ์ (%RH)	8 บิต
2	ข้อมูลทศนิยมของความชื้นสัมพัทธ์ (%RH)	8 บิต
3	ข้อมูลจำนวนเต็มของอุณหภูมิ (°C)	8 บิต
4	ข้อมูลทศนิยมของอุณหภูมิ (°C)	8 บิต
5	ข้อมูลตรวจสอบผลรวม (check sum)	8 บิต



รูปที่ 5-8 ไตอะแกรมเวลาแสดงกระบวนการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์กับ DHT11

5.2.2.2 กระบวนการสื่อสารกับ DHT11

ในรูปที่ 5-8 เป็นไคอะแกรมเวลาแสดงกระบวนการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์กับ DHT11 โดยเริ่มจาก

(1) ไมโครคอนโทรลเลอร์ส่งสัญญาณเริ่มต้น ด้วยการทำให้บัสเป็นลอจิกต่ำนานอย่างน้อย 18 ไมโครวินาที จากนั้นทำให้บัสกลับมาเป็นลอจิกสูงเพื่อรอการตอบสนองจาก DHT11 โดยจะรอนานประมาณ 20 ถึง 40 ไมโครวินาที

(2) เมื่อ DHT11 ได้รับสัญญาณเริ่มต้นการติดต่อ ก็จะตอบสนองกลับมา ด้วยการทำให้สถานะของบัสเป็นลอจิกต่ำนาน 80 ไมโครวินาทีและลอจิกสูงนาน 80 ไมโครวินาทีเพื่อแจ้งกลับมายังไมโครคอนโทรลเลอร์จะเริ่มต้นส่งข้อมูลกลับมา 40 บิต

(3) DHT11 เริ่มต้นส่งข้อมูลด้วยบิตเริ่มต้นเป็นลอจิกต่ำนาน 50 ไมโครวินาที จากนั้นทยอยส่งข้อมูลต่อเนื่อง 40 บิต โดยข้อมูล “0” จะทำให้บัสมีสถานะลอจิกสูงนาน 26 ถึง 28 ไมโครวินาที ในขณะที่ข้อมูล “1” จะทำให้บัสมีสถานะลอจิกสูงนาน 70 ไมโครวินาที ข้อมูลแต่ละบิตจะถูกค้นด้วยสถานะลอจิกต่ำนาน 50 ไมโครวินาที

(4) เมื่อส่งข้อมูลครบ DHT11 จะทำให้บัสมีสถานะลอจิกต่ำนาน 50 ไมโครวินาทีอีกครั้งเพื่อแจ้งว่าข้อมูลส่งครบแล้ว จากนั้น DHT11 จะทำให้บัสกลับมาเป็นลอจิกสูงอีกครั้ง เพื่อกลับเข้าสู่สภาวะพร้อมทำงานในรอบใหม่ต่อไป

5.2.3 การเตรียมการเพื่อพัฒนาโปรแกรมด้วยภาษา Python สำหรับ Raspberry Pi 3 ในการติดต่อกับตัวตรวจจับ DHT11

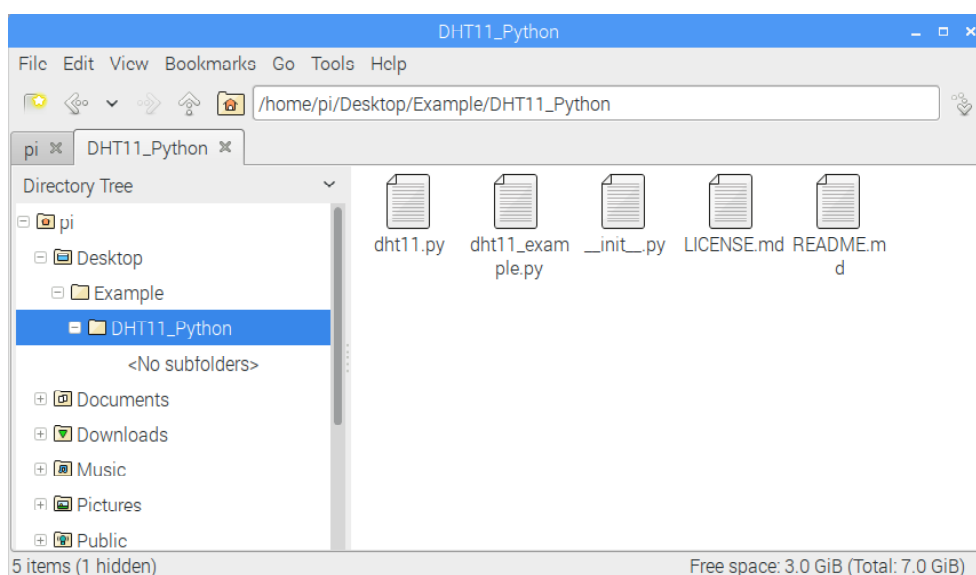
DHT11 เป็นโมดูลที่ใช้ในการวัดอุณหภูมิกับความชื้นในอากาศที่มีการรับส่งข้อมูลด้วยสายเส้นเดียว โดยเดิมทีการพัฒนาโปรแกรมสำหรับ Raspberry Pi 3 ด้วยภาษา Python เพื่อติดต่อและอ่านค่าจาก DHT11 จะค่อนข้างยุ่งยากและมีความซับซ้อนในการติดตั้งไลบรารีพอสมควร จนกระทั่งในระบบปฏิบัติการ Raspbian Jessie with Pixel รุ่นใหม่ได้ช่วยอำนวยความสะดวกในการจัดการติดต่อกับกับอุปกรณ์ระบบบัส 1 สาย อย่างไรก็ตาม ก็ยังต้องมีการติดตั้งไลบรารีเพิ่มเติมก่อน

มีขั้นตอนดังนี้

- (1) เชื่อมต่อบอร์ด Raspberry Pi กับเครือข่ายอินเทอร์เน็ต
- (2) เปิดหน้าต่าง Terminal
- (3) พิมพ์คำสั่งต่อไปนี้เพื่อดาวน์โหลดไลบรารี

```
git clone https://github.com/szazo/DHT11_Python.git
```

- (4) เมื่อดาวน์โหลดแล้ว ให้เปิดโฟลเดอร์ที่ชื่อว่า **DHT11_Python**



จะพบไฟล์ไลบรารีชื่อว่า **dht11.py** ส่วนไฟล์ตัวอย่างคือ **dht11_example.py** ทุกครั้งที่สร้างโปรแกรมอ่านค่าจาก DHT11 จะต้องนำไฟล์ **dht11.py** ไปเก็บไว้ในโฟลเดอร์เดียวกันด้วยทุกครั้ง

5.2.4 ตัวอย่างการทดสอบอ่านค่าจาก DHT11 ด้วยบอร์ด Raspberry Pi 3

ตัวอย่างที่นำเสนอต่อไปนี้ เป็นการติดต่อกับ ZX-DHT11 โดยใช้บอร์ด Raspberry Pi 3 เพื่ออ่านค่าความชื้นสัมพัทธ์และอุณหภูมิจากโมดูลตรวจจับ ZX-DHT11 พัฒนาโปรแกรมด้วย Python3

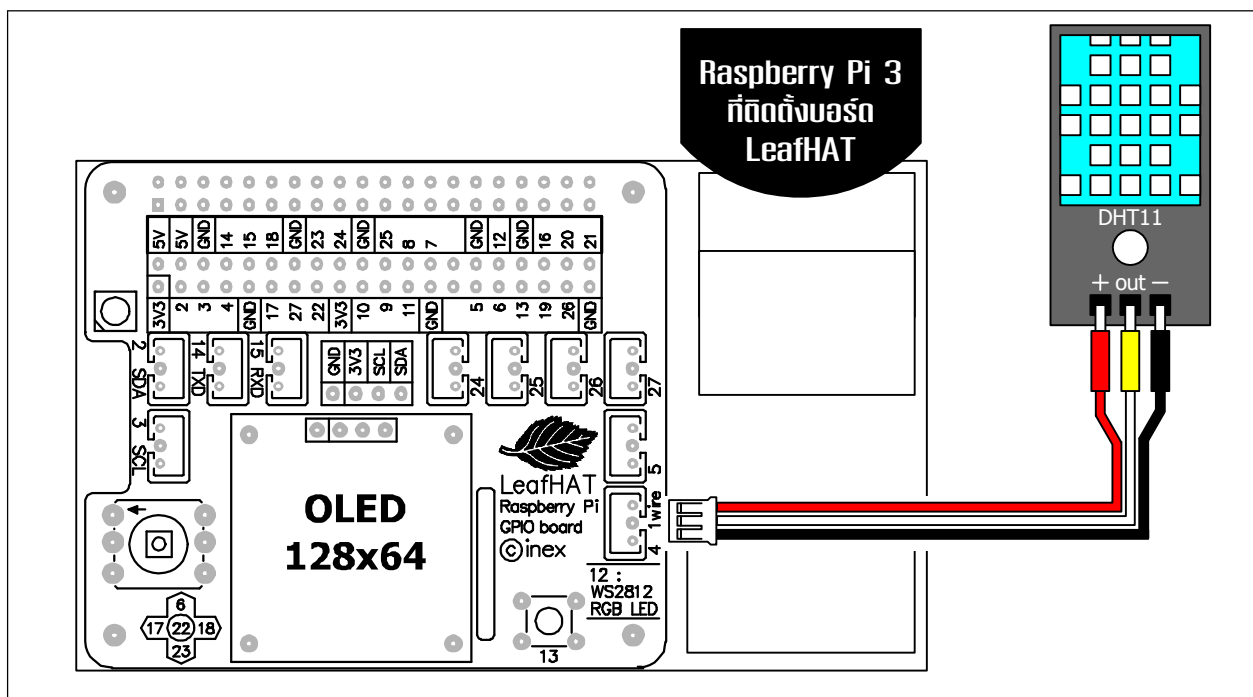
(1) ต่อยังจตามรูปที่ 5-9 เพื่อทดสอบการทำงาน

(2) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) เขียนคำสั่งตามโปรแกรมที่ 5-2 บันทึกไฟล์เป็น **WeatherMonitor.py**

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่าง Terminal แล้วสั่งรันด้วยคำสั่ง **sudo python3 WeatherMonitor.py**

เมื่อ Raspberry Pi 3 ติดต่อกับโมดูล DHT11 ได้ จะอ่านค่าความชื้นสัมพัทธ์และอุณหภูมิที่ตัวตรวจจับวัดได้ นำมาแสดงผลที่หน้าต่าง Terminal ดังในรูปที่ 5-10



รูปที่ 5-9 วงจรการเชื่อมต่อบอร์ด Raspberry Pi 3 กับบอร์ด ZX-DHT11 เพื่ออ่านค่าความชื้นสัมพัทธ์และอุณหภูมิจากโมดูล DHT11 มาแสดงที่หน้าต่างเทอร์มินอล

```

import RPi.GPIO as GPIO
import dht11
import time
import datetime

# initialize GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()

# read data using pin 4
dht11 = dht11.DHT11(pin=4)

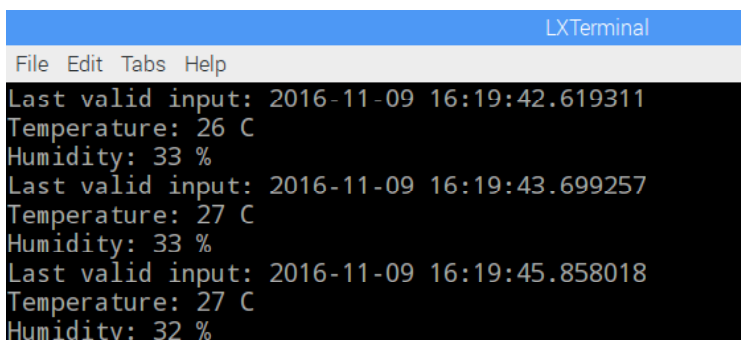
while True:
    result = dht11.read()
    if result.is_valid():
        print("Last valid input: " + str(datetime.datetime.now()))
        print("Temperature: %d C" % result.temperature)
        print("Humidity: %d %" % result.humidity)
        time.sleep(1)

```

คำอธิบายโปรแกรมเพิ่มเติม

- เรียกใช้งานไลบรารี DHT11 ด้วยคำสั่ง `import dht11`
- กำหนดขาพอร์ตของ RaspberryPi ที่ต่อกับตัวตรวจจับด้วยคำสั่ง `dht11 = dht11.DHT11-(pin=4)` ซึ่งในที่นี้คือ ขา GPIO4
- อ่านค่า DHT11 ด้วยคำสั่ง `result = dht11.read()`
- ตรวจสอบความถูกต้องโดยใช้คำสั่ง `result.is_valid()` เป็นเงื่อนไข ถ้าข้อมูลถูกต้องจะคืนค่า True ถ้าไม่ถูกต้องจะคืนค่าเป็น False
- เมื่อต้องการค่าอุณหภูมิใช้คำสั่ง `result.temperature` และอ่านค่าความชื้นใช้คำสั่ง `result.humidity`

โปรแกรมที่ 5-2 ไฟล์ `WeatherMonitor.py` โปรแกรมภาษา Python สำหรับบอร์ด Raspberry Pi 3 เพื่ออ่านค่าจากมินิบอร์ดตัวตรวจจับความชื้นสัมพัทธ์และอุณหภูมิ ZX-DHT11 มาแสดงผลที่หน้าต่าง Terminal



```

LXTerminal
File Edit Tabs Help
Last valid input: 2016-11-09 16:19:42.619311
Temperature: 26 C
Humidity: 33 %
Last valid input: 2016-11-09 16:19:43.699257
Temperature: 27 C
Humidity: 33 %
Last valid input: 2016-11-09 16:19:45.858018
Temperature: 27 C
Humidity: 32 %

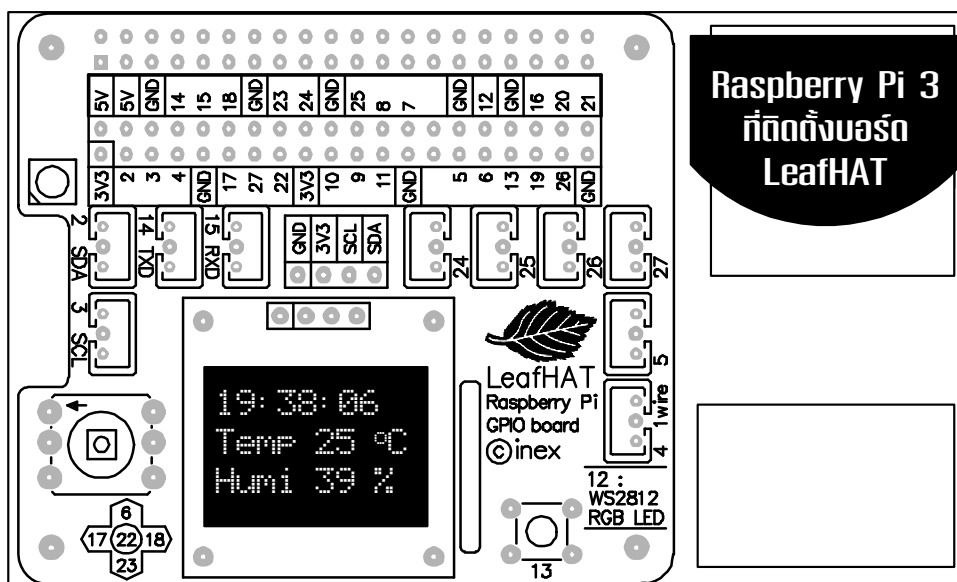
```

รูปที่ 5-10 ผลการทำงานของโปรแกรมที่ 5-2 หน้าต่าง Terminal แสดงค่าความชื้นสัมพัทธ์และอุณหภูมิที่ได้จากโมดูลตรวจจับ DHT11

ลำดับต่อไปเป็นการเปลี่ยนการแสดงผลจากการส่งค่าแสดงที่หน้าต่าง Terminal เป็นแสดงที่จอแสดงผล OLED บนบอร์ด LeafHAT ที่เชื่อมต่อกับ Raspberry Pi 3 ผ่านทางพอร์ต GPIO 40 ขา

- (5) ยังคงใช้วงจรตามรูปที่ 5-9 ในการทดสอบการทำงาน
- (6) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์
- (7) เขียนคำสั่งตามโปรแกรมที่ 5-3 บันทึกไฟล์เป็น **WeatherMonitorOLED.py**
- (4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่าง Terminal แล้วสั่งรันด้วยคำสั่ง **sudo python3 WeatherMonitorOLED.py**

เมื่อ Raspberry Pi 3 ติดต่อกับโมดูล DHT11 ได้ จะอ่านความชื้นสัมพัทธ์และอุณหภูมิที่ตัวตรวจจับวัดได้ นำมาแสดงที่จอแสดงผล OLED บนบอร์ด LeafHAT ดังในรูปที่ 5-11



รูปที่ 5-11 แสดงผลการทำงานของโปรแกรมที่ 5-3 จอแสดงผล OLED ของบอร์ด LeafHAT ที่เชื่อมต่อกับบอร์ด Raspberry Pi 3 ผ่านทางพอร์ต GPIO 40 ขา แสดงค่าความชื้นสัมพัทธ์และอุณหภูมิที่ได้จากตัวตรวจจับ DHT11

```

import RPi.GPIO as GPIO
import Adafruit_SSD1306
from DHT11_Python import dht11
from PIL import Image, ImageDraw, ImageFont
import time
import datetime

# initialize GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()

# read data using pin 4
instance = dht11.DHT11(pin=4)
msgTime = ""
msgTemp = ""
msgHumi = ""

disp = Adafruit_SSD1306.SSD1306_128_64(rst=1)
disp.begin()
disp.clear()
disp.display()

while True:
    result = instance.read()
    datenow=datetime.datetime.now()
    if result.is_valid():
        msgTime=datetime.strftime("%H:%M:%S")
        msgTemp="Temp %d oC" % result.temperature
        msgHumi="Humi %d %" % result.humidity
    time.sleep(1)
    # Display image.
    width = disp.width
    height = disp.height
    image = Image.new('1', (width, height))
    draw = ImageDraw.Draw(image)
    font = ImageFont.truetype('VCR_OSD_MONO_1.001.ttf',22)
    draw.text((0,0),msgTime,font=font, fill=1)
    draw.text((0,20),msgTemp,font=font, fill=1)
    draw.text((0,40),msgHumi,font=font, fill=1)
    disp.image(image)
    disp.display()

```

โปรแกรมที่ 5-3 ไฟล์ WeatherMonitorOLED.py โปรแกรมภาษา Python สำหรับบอร์ด Raspberry Pi 3 เพื่ออ่านค่าจากมินิบอร์ดตัวตรวจจับความชื้นสัมพัทธ์และอุณหภูมิ ZX-DHT11 มาแสดงที่จอแสดง OLED บนบอร์ด LeafHAT ที่เชื่อมต่อกับ Raspberry Pi 3 ผ่านทางคอนเน็กเตอร์พอร์ต GPIO 40 ขา

5.3 BH1750 มินิบอร์ดตัวตรวจจับและวัดความเข้มแสงติดต่อผ่านบัส I²C

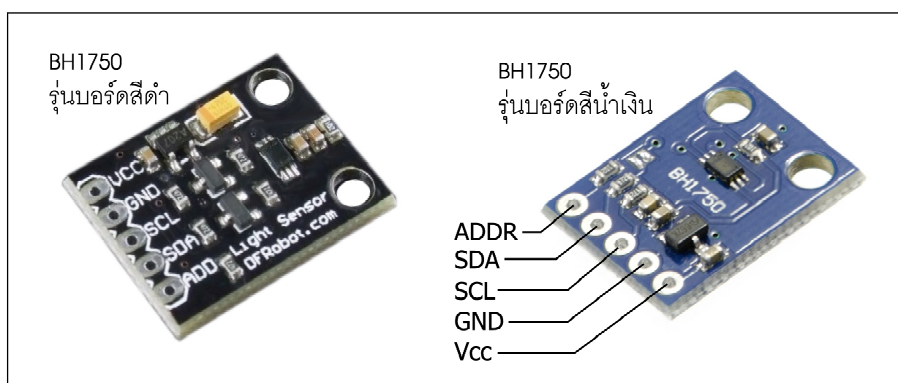
เป็นแผงวงจรขนาดเล็กที่ติดตั้งตัวตรวจจับแสงเบอร์ BH1750 โดย BH1750 เป็นผลงานของ ROHM Semiconductor (www.rohm.com) ผู้ผลิตอุปกรณ์สารกึ่งตัวนำชั้นนำของโลก BH1750 นับเป็นตัวตรวจจับแสงที่มีประสิทธิภาพสูง ใช้งานง่าย ด้วยการติดต่อผ่านบัส 2 สายหรือ I²C ให้ผลการวัดความเข้มแสงเป็นหน่วยลักซ์ (Lux) ทำให้นำข้อมูลที่ได้ไปใช้ประโยชน์ต่อได้ทันที โดยไม่ต้องพึ่งกระบวนการทางคณิตศาสตร์เพื่อแปลงหน่วยภายในตัวตรวจจับมีวงจรแปลงสัญญาณอะนาล็อกเป็นดิจิทัลความละเอียด 16 บิตทำให้ได้ข้อมูลดิจิทัลของความเข้มแสงที่มีความละเอียดและแม่นยำมากพอสำหรับการนำไปสร้างเครื่องวัดความเข้มแสงหรือลักซ์มิเตอร์ (Luxmeter)

5.3.1 คุณสมบัติทางเทคนิคของบอร์ดวัดความเข้มแสง BH1750

คุณสมบัติทางเทคนิคที่ควรทราบของบอร์ดวัดความเข้มแสง BH1750 มีดังนี้

- ติดตั้งตัวตรวจจับแสงเบอร์ BH1750 บนบอร์ด ภายในมีตัวรับแสงเป็นโฟโตไดโอดต่อร่วมกับวงจรขยายสัญญาณ, วงจรแปลงสัญญาณอะนาล็อกเป็นดิจิทัล และวงจรเชื่อมต่อบัส I²C
- มีตัวต้านทานต่อพูลอัพที่ขาเชื่อมต่อบัส I²C ไว้พร้อม ทำให้นำไปเชื่อมต่อกับไมโครคอนโทรลเลอร์หรือ Raspberry Pi ทำได้ทันที โดยไม่ต้องต่อตัวต้านทานเพิ่ม
- ใช้ไฟเลี้ยง +3 ถึง +5Vdc กินกระแสไฟฟ้าน้อยมาก ประมาณ 200µA เท่านั้น
- ย่านวัดความเข้มแสง 1 ถึง 65,535 ลักซ์ มีค่าความผิดพลาด 20%
- กำหนดแอดเดรสให้กับ BH1750 ได้ 2 รูปแบบผ่านทางขา ADDR
- ทนต่อการรบกวนจากแสงอินฟราเรด
- ขนาด 21 x 16 มม.

รูปที่ 5-12 แสดงหน้าตาและการจัดขาของบอร์ดวัดความเข้มแสงที่ใช้ตัวตรวจจับเบอร์ BH1750 จะเห็นได้ว่า มีการผลิตออกมา 2 แบบ มีการจัดขาสัญญาณสลับกันเล็กน้อย ดังนั้น เมื่อนำมาต่อใช้งานควรตรวจสอบตำแหน่งขาให้ถูกต้องก่อน



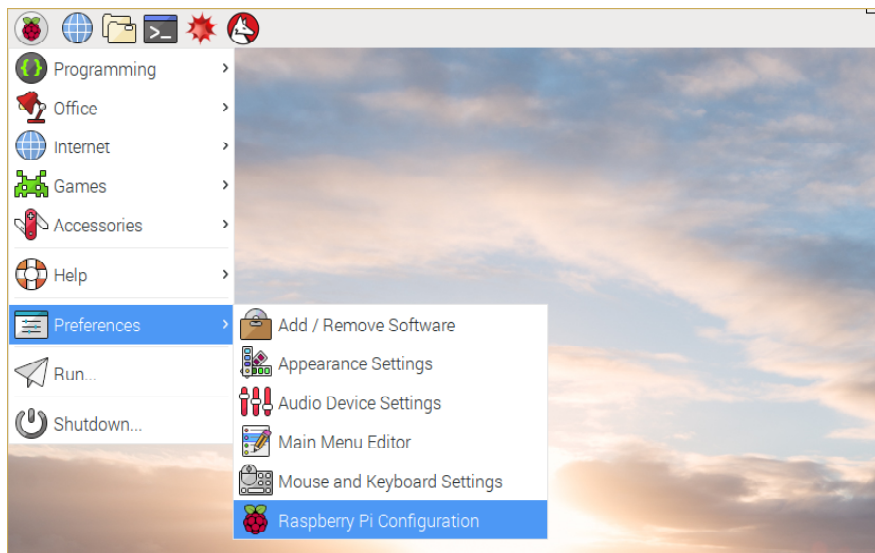
รูปที่ 5-12 หน้าตาและการจัดขาของบอร์ดวัดความเข้มแสง BH1750 มีการผลิตออกมาจำหน่าย 2 รูปแบบหลัก

5.3.2 การเตรียมการเพื่อติดต่อกับมินิบอร์ดตัวตรวจจับสนแสง BH1750

5.3.2.1 เปิดความสามารถการติดต่อกับอุปกรณ์ระบบบัส I2C

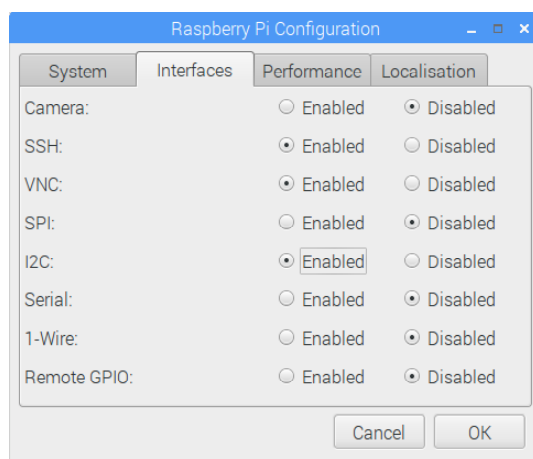
ตัวตรวจจับสนแสง BH1750 ใช้การติดต่อแบบบัส 2 สายหรือ I²C เมื่อต้องการใช้งานกับ Raspberry Pi 3 จะมีขั้นตอนต่างๆ ในการเปิดการติดต่อดังนี้

(1) เปิดความสามารถในการติดต่อบัส I2C ด้วยการเข้าไปที่ **Preferences > Raspberry Pi Configuration**



(2) เลือกเปิดแท็บ **Interfaces** แล้วคลิกทำเครื่องหมายเพื่อเลือกที่บรรทัดของ I2C โดย

- ถ้าต้องการเปิดใช้งานให้เลือก **Enabled**
- หากต้องการปิดการใช้งาน ให้เลือก **Disabled**



(3) คลิกปุ่ม **OK** เพื่อบันทึกการตั้งค่า

5.3.2.2 ตรวจสอบรหัสแอดเดรสของตัวตรวจจับ BH1750

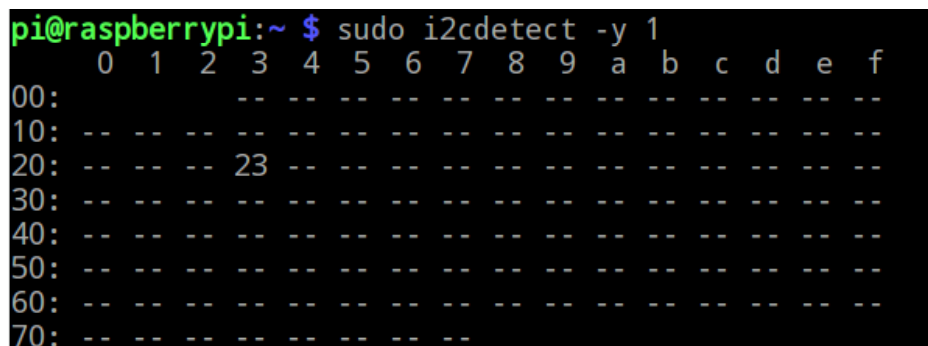
เนื่องจากตัวตรวจจับ BH1750 เป็นอุปกรณ์ที่ต้องติดต่อผ่านระบบบัส I²C มันจึงมีแอดเดรสประจำตัว ก่อนการใช้งานจึงต้องตรวจสอบรหัสแอดเดรสเสียก่อนว่า เป็นเลขรหัสใด

(1) เชื่อมต่อมินิบอร์ด BH1750 เข้ากับบอร์ด Raspberry Pi 3 ผ่านทางจุดต่อบัส I²C บนบอร์ด LeafHAT หรือจากจุดต่อกอนเน็กเตอร์ GPIO 40 ขาของ Raspberry Pi โดยตรง ดังรูปที่ 5-13

(2) เปิดหน้าต่าง Terminal พิมพ์คำสั่งต่อไปนี้ เพื่อตรวจสอบการเชื่อมต่อและรหัสแอดเดรสของ BH1750

```
sudo i2cdetect -y 1
```

(3) จะเห็นตารางแสดงการตรวจสอบและได้เลขรหัสแอดเดรสของตัวตรวจจับแสง BH1750 เป็นค่า 0x23 ดังรูป



```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  23  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

5.3.3 ตัวอย่างการทดลองใช้งาน BH1750 ด้วย Raspberry Pi 3

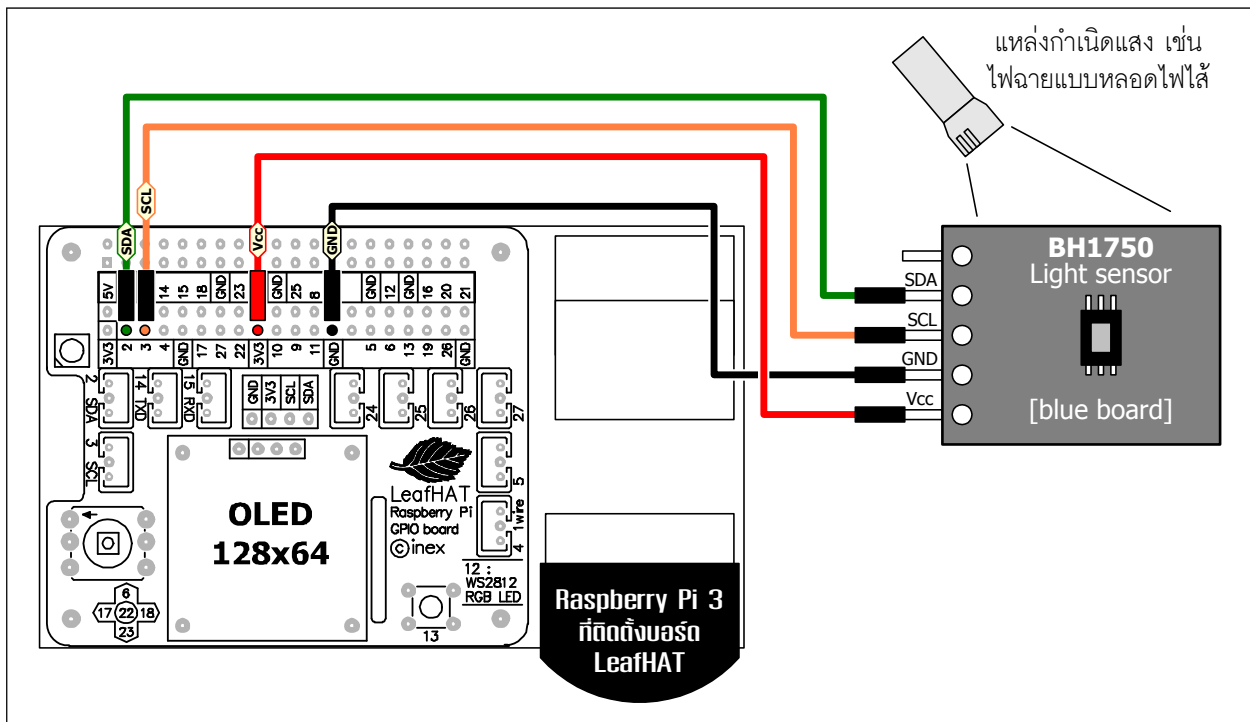
ตัวอย่างนี้เป็นการกำหนดให้ Raspberry Pi 3 ติดต่อกับตัวตรวจจับแสง BH1750 ผ่านบัส I²C เพื่ออ่านค่าความเข้มแสงที่วัดได้ในหน่วยลักซ์ (Lux) โดยเขียนโปรแกรมติดต่อด้วยภาษา Python

(1) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์ พิมพ์โค้ดโปรแกรมที่ 5-3

(2) ระบุแอดเดรสของอุปกรณ์จากที่ตรวจสอบได้ลงในบรรทัดคำสั่ง `addr = xxxx # i2c adress` ในที่นี้ได้ค่าแอดเดรสคือ **0x23**

(3) บันทึกไฟล์ชื่อ **SimpleLightMeter.py**

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **sudo python3 SimpleLightMeter.py**



รูปที่ 5-13 วงจรการเชื่อมต่อบอร์ด Raspberry Pi 3 กับบอร์ดวัดความเข้มแสง BH1750 เพื่ออ่านค่านำมาแสดงที่หน้าต่างเทอร์มินอล

ข้อมูลค่าความเข้มแสงในสภาพแวดล้อม

กลางคืน	มีค่าความเข้มแสง 0.001 ถึง 0.02 ลักซ์
คืนพระจันทร์เต็มดวง	มีค่าความเข้มแสง 0.02 ถึง 0.3 ลักซ์
พื้นที่ในร่มยามมีเมฆ	มีค่าความเข้มแสง 5 ถึง 50 ลักซ์
พื้นที่กลางแจ้งยามมีเมฆ	มีความเข้มแสง 50 ถึง 500 ลักซ์
ในร่มที่มีไฟสว่างมาก	มีความเข้มแสง 100 ถึง 1,000 ลักซ์
แสงไฟอ่านหนังสือ	มีความเข้มแสง 50 ถึง 60 ลักซ์
หน้าจอภาพของโฮมวิดีโอในบ้าน	มีความเข้มแสง 1400 ลักซ์

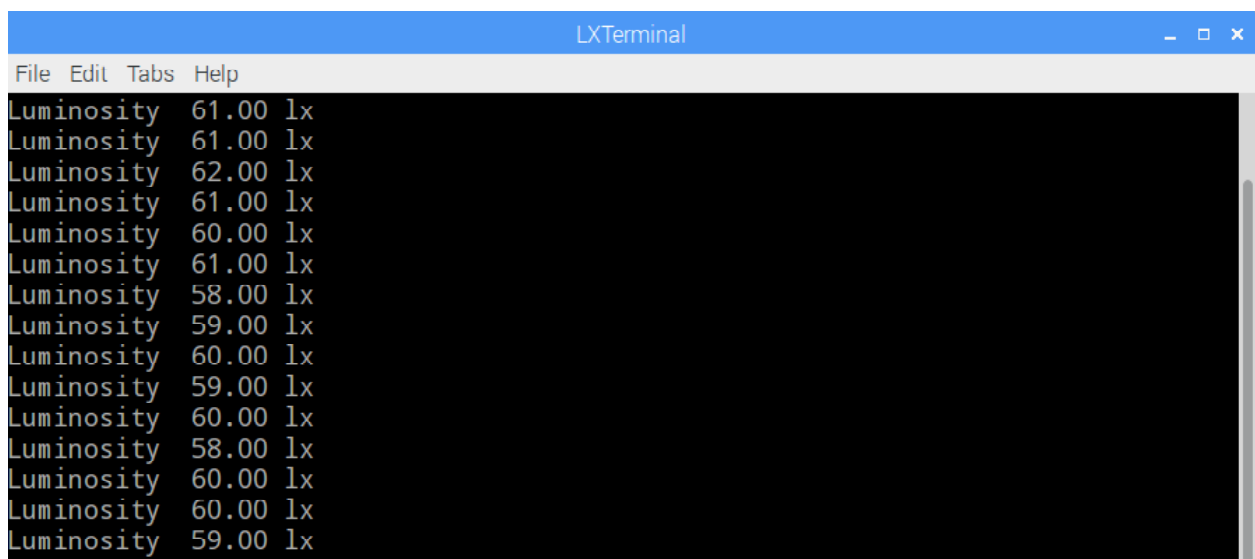
ข้อมูลนี้เป็นค่าตัวเลขโดยประมาณนำมาใช้ประกอบการอธิบายและให้เห็นในภาพรวมเท่านั้น

```
import time
import smbus
bus = smbus.SMBus(1) #(512MB ro 1024MB)
addr = 0x23 # i2c adress
while True:
    data = bus.read_i2c_block_data(addr,0x10)
    print ("Luminosity " , "%.2f" % (data[1] + (256 * data[0]) / 1.2), "lx")
    time.sleep(0.5)
```

โปรแกรมที่ 5-3 โค้ดภาษา Python ไฟล์ SimpleLightMeter.py สำหรับบอร์ด Raspberry Pi 3 เพื่ออ่านค่าความเข้มแสงจากบอร์ดวัดความเข้มแสง BH1750 มาแสดงผลที่หน้าต่าง Terminal

(5) นำแหล่งกำเนิดแสงมาส่องไปยังตัวตรวจจับ แล้วค่อยๆ ปรับลดและเพิ่มความเข้มแสง หรือใช้มือบังแสงวงจรตรวจจับ สังเกตการทำงานของวงจรผ่านทางหน้าต่างเทอร์มินอล

เมื่อรันโปรแกรม หน้าต่าง Terminal ปรากฏขึ้นมาแสดงค่าความเข้มแสงที่ตรวจวัดได้ (Luminosity) ดังรูปที่ 5-14



```
LXTerminal
File Edit Tabs Help
Luminosity 61.00 lx
Luminosity 61.00 lx
Luminosity 62.00 lx
Luminosity 61.00 lx
Luminosity 60.00 lx
Luminosity 61.00 lx
Luminosity 58.00 lx
Luminosity 59.00 lx
Luminosity 60.00 lx
Luminosity 59.00 lx
Luminosity 60.00 lx
Luminosity 58.00 lx
Luminosity 60.00 lx
Luminosity 60.00 lx
Luminosity 59.00 lx
Luminosity 59.00 lx
```

รูปที่ 5-14 ผลการทำงานของโปรแกรมที่ 5-3 หน้าต่าง Terminal แสดงค่าความเข้มแสงที่วัดได้จากบอร์ดวัดความเข้มแสง BH1750

(6) จากการทำงานของโปรแกรมที่ 5-3 ซึ่งกำหนดให้แสดงผลการตรวจจับและวัดความเข้มแสงไปที่หน้าต่าง Terminal ในการทดสอบต่อไปนี้จะเปลี่ยนอุปกรณ์แสดงผลเป็นจอแสดงผล OLED ที่ติดตั้งอยู่บนบอร์ด LeafHAT ซึ่งเชื่อมต่อกับ Raspberry Pi 3 ผ่านทางคอนเน็กเตอร์ GPIO 40 ขา การเชื่อมต่ออุปกรณ์ยังคงใช้ตามรูปที่ 5-13

(7) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เทกซ์เอดิเตอร์ จากนั้นเขียนคำสั่งตามโปรแกรมที่ 5-4 บันทึกไฟล์เป็น **LightMeterOLED.py**

(8) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่าง Terminal แล้วสั่งรันด้วยคำสั่ง **sudo python3 LightMeterOLED.py**

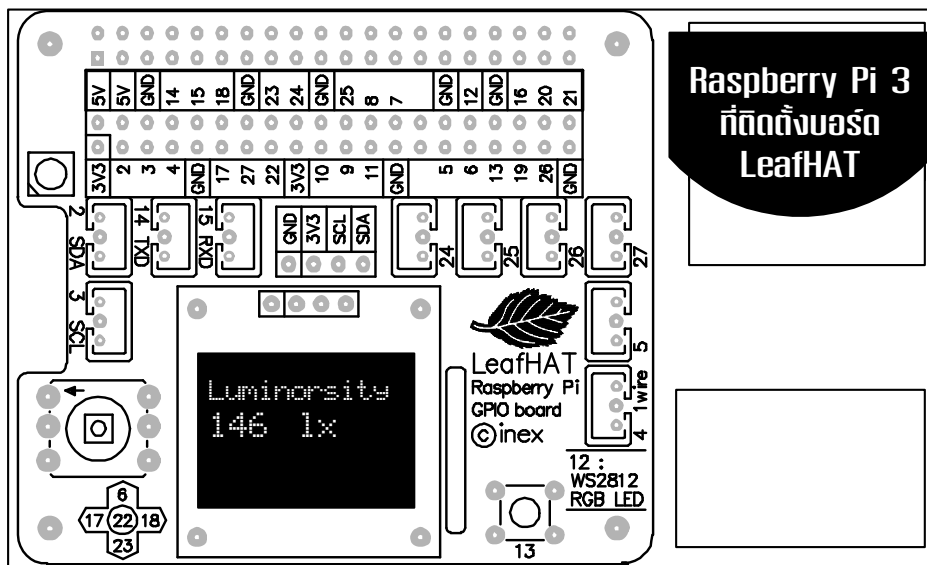
```
import smbus
import Adafruit_SSD1306
from PIL import Image, ImageDraw, ImageFont
import time

bus = smbus.SMBus(1) #(512MB ro 1024MB)
addr = 0x23 # i2c adress

disp = Adafruit_SSD1306.SSD1306_128_64(rst=1)
disp.begin()
disp.clear()
disp.display()

while True:
    data = bus.read_i2c_block_data(addr,0x10)
    msgLumi="%d lx" % (data[1] + (256 * data[0]) / 1.2)
    time.sleep(0.5)
    width = disp.width
    height = disp.height
    image = Image.new('1', (width, height))
    draw = ImageDraw.Draw(image)
    font = ImageFont.truetype('VCR_OSD_MONO_1.001.ttf',22)
    draw.text((0,0),"Luminosity",font=font, fill=1)
    draw.text((0,24),msgLumi,font=font, fill=1)
    disp.image(image)
    disp.display()
```

โปรแกรมที่ 5-4 โค้ดภาษา Python ไฟล์ LightMeterOLED.py สำหรับบอร์ด Raspberry Pi 3 อ่านค่าความเข้มแสงจากบอร์ดวัดความเข้มแสง BH1750 มาแสดงที่จอแสดงผล OLED บนบอร์ด LeafHAT ที่เชื่อมต่อกับ Raspberry Pi 3 ผ่านทางคอนเน็กเตอร์พอร์ต GPIO 40 ขา



รูปที่ 5-15 แสดงผลการทำงานของโปรแกรมที่ 5-4 จอแสดงผล OLED ของบอร์ด LeafHAT ที่เชื่อมต่อกับบอร์ด Raspberry Pi 3 ผ่านทางพอร์ต GPIO 40 ขา แสดงค่าความเข้มแสงในหน่วยลักซ์ที่ได้จากตัวตรวจจับและวัดความเข้มแสง BH1750

(9) นำแหล่งกำเนิดแสงมาส่องไปยังตัวตรวจจับ แล้วค่อยๆ ปรับลดและเพิ่มความเข้มแสง หรือใช้มือบังแสงวงจรตรวจจับ สังเกตผลการทำงานที่จอแสดงผล OLED บนบอร์ด LeafHAT

เมื่อรันโปรแกรม BH1750 ทำการวัดความเข้มแสงแล้วส่งไปยัง Raspberry Pi 3 เพื่อนำค่าไปแสดงที่จอแสดงผล OLED บนบอร์ด LeafHAT ดังรูปที่ 5-15

5.3.4 ระบบแจ้งเตือนเมื่อความเข้มแสงต่ำ

ตัวอย่างนี้เป็นการประยุกต์ใช้งานตัวตรวจจับแสงกับบอร์ด Raspberry Pi 3 เพื่อสร้างระบบแจ้งเตือนอัตโนมัติ เมื่อระดับความเข้มของแสงที่ทำการตรวจสอบลดต่ำกว่าค่าที่กำหนด

(1) ต่อยังจตามรูปที่ 5-16

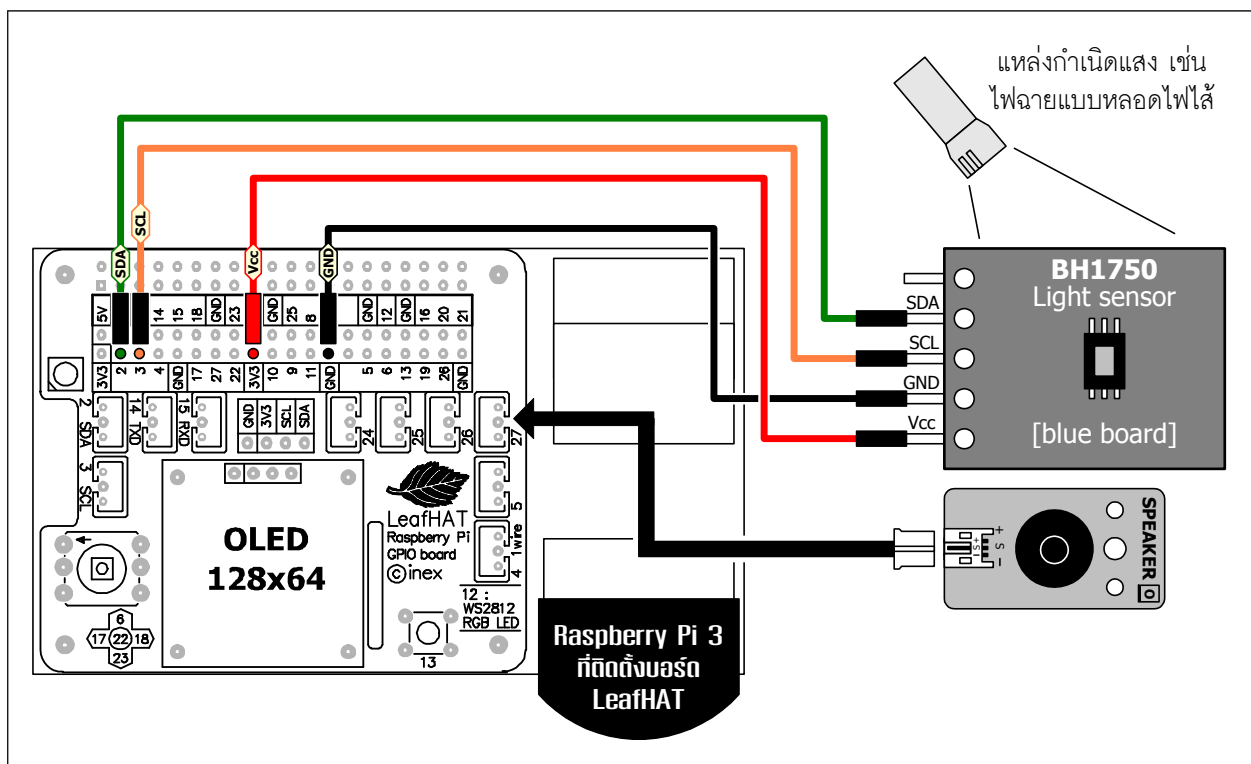
(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์ แล้วพิมพ์คำสั่งตามโปรแกรมที่ 5-5

(3) บันทึกไฟล์ชื่อ **LightAlarm.py**

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **sudo python3 LightAlarm.py**

(5) หาแหล่งกำเนิดแสงมาส่องไปยังตัวตรวจจับ แล้วค่อยๆ ลดความเข้มลง หรือใช้มือบังแผงวงจรตรวจจับ สังเกตการทำงานของวงจร

เมื่อตัวตรวจจับ BH1750 ตรวจจับความเข้มแสงได้ค่าน้อย (แสงหรี่ลงหรือถูกบังแสง) จะได้ยินเสียงสัญญาณเตือนดังขึ้นจากลำโพงปิโยโซ ค่าอ้างอิงที่ใช้ในการตัดสินใจคือ 90 ลักซ์



รูปที่ 5-16 วงจรการเชื่อมต่อบอร์ด Raspberry Pi 3 กับบอร์ดวัดความเข้มแสง BH1750 เพื่ออ่านค่ามาเปรียบเทียบกับค่าที่กำหนด เพื่อสร้างระบบแจ้งเตือนเมื่อความเข้มแสงลดลง


```

import RPi.GPIO as GPIO
import time
import smbus
import datetime
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(27, GPIO.OUT)
blink = GPIO.PWM(27,500)
multiply = 1
duty = 0
counttime=0;
st=0
blink.start(duty)
bus = smbus.SMBus(1) #(512MB)

addr = 0x23 # i2c adress

while True:
    da=datetime.datetime.now()
    microsec=da.microsecond
    if microsec > 700000:
        blink.ChangeDutyCycle(st)
    else:
        blink.ChangeDutyCycle(0)

    data = bus.read_i2c_block_data(addr,0x11)
    lum=(data[1] + (data[0]<<8) / 1.2)

    if lum<90:
        st=50
    else:
        st=0

    time.sleep(0.2)

```

โปรแกรมที่ 5-5 โค้ดภาษา Python ไฟล์ LightAlarm.py สำหรับบอร์ด Raspberry Pi 3 อ่านค่าความเข้มแสงจากบอร์ดวัดความเข้มแสง BH1750 ผ่านบัส I²C เพื่อเปรียบเทียบกับค่าที่กำหนดแล้วขับสัญญาณเสียงออกทางลำโพงเมื่อค่าความเข้มแสงต่ำกว่าค่าอ้างอิง

5.3.5 LightLogger บันทึกข้อมูลความเข้มแสงลงไฟล์

ตัวอย่างนี้เป็นการนำตัวอย่างในหัวข้อ 5.3.3 มาต่อยอด จากการแสดงค่าความเข้มแสงผ่านทางหน้าจอเทอร์มินอลเท่านั้น มาเพิ่มเติมให้ระบบทำการบันทึกข้อมูลของความเข้มแสงที่ระบบตรวจวัดได้ เก็บในรูปแบบของแฟ้มข้อมูลแบบตัวอักษรหรือเท็กซ์ไฟล์ (text file) ที่มีนามสกุล .txt

```
import time
import smbus
import datetime

bus = smbus.SMBus(1) #(512MB)
addr = 0x23 # i2c adress

while True:
    data = bus.read_i2c_block_data(addr,0x11)
    lum=(data[1] + (data[0]<<8) / 1.2)
    date=str(datetime.datetime.now())
    with open("BH1750.txt", "a") as text_file:
        text_file.write("DateTime: %s Luminosity: %.2f lx\n" % (date,lum,))
    time.sleep(0.5)
```

คำอธิบายโปรแกรมเพิ่มเติม

ในโปรแกรมมีคำสั่งที่สำคัญคือ คำสั่ง `open` ที่ใช้ในการเปิดไฟล์ข้อมูล หากไม่มีไฟล์อยู่ คำสั่งนี้จะสร้างไฟล์ให้โดยอัตโนมัติ จากนั้นจึงทำการเขียนข้อมูลลงในไฟล์ด้วยคำสั่ง `text_file.write` ต่อไป

รายละเอียดเพิ่มเติมของคำสั่ง `open` มีดังนี้

รูปแบบ

`open (/Directory/Filename ,mode)`

พารามิเตอร์

`/Directory/Filename` คือ ที่อยู่ของ File และชื่อ File ที่ต้องการทำงาน

`mode` คือ กระบวนการที่กระทำกับไฟล์ ประกอบด้วย

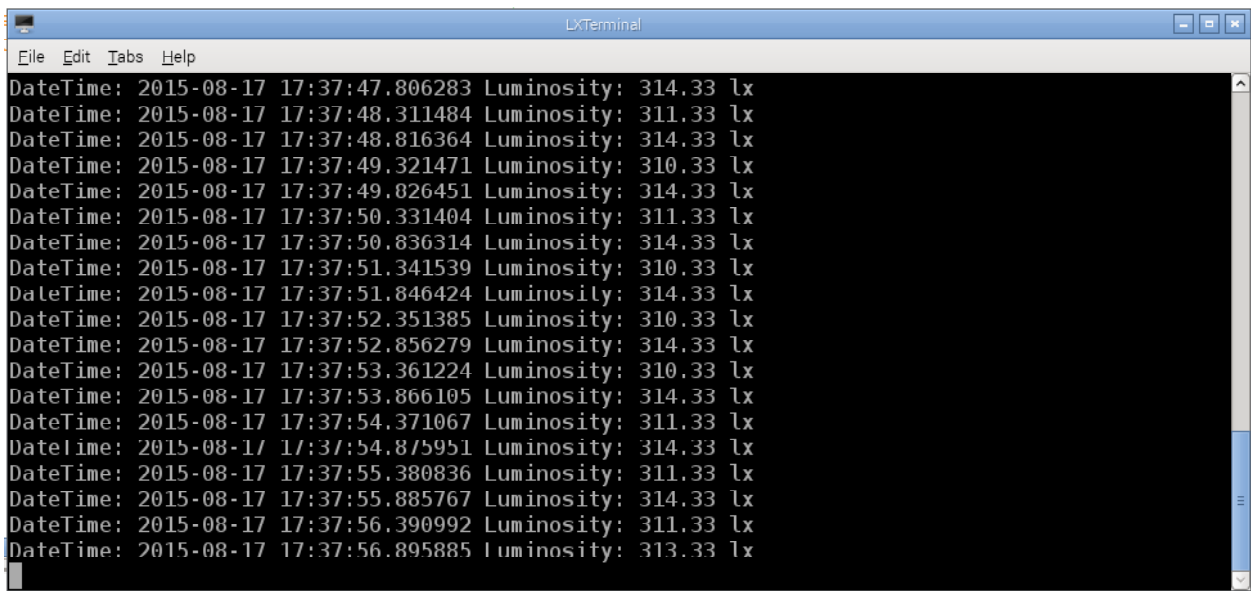
- `r` อ่านอย่างเดียว เริ่มต้นที่ต้นไฟล์
- `r+` อ่านและเขียน เริ่มต้นที่ต้นไฟล์
- `w` เขียนอย่างเดียว เมื่อเปิดไฟล์ขึ้นมา ลบข้อมูลเก่าทิ้ง ถ้าไม่มีไฟล์ จะสร้างไฟล์ขึ้นมา
- `w+` อ่านและเขียน เมื่อเปิดไฟล์ขึ้นมา ลบข้อมูลเก่าทิ้ง ถ้าไม่มีไฟล์ จะสร้างไฟล์
- `a` เขียนต่อท้าย เมื่อเปิดไฟล์ขึ้นมา ให้ทำการเขียนต่อท้ายไฟล์ ถ้าไม่มีไฟล์ จะสร้างไฟล์ขึ้นมา
- `a+` อ่านและเขียนต่อท้าย เมื่อเปิดไฟล์ขึ้นมา ให้เขียนต่อท้ายไฟล์ ถ้าไม่มีไฟล์จะสร้างไฟล์ขึ้นมา

โปรแกรมที่ 5-5 โค้ดภาษา Python ไฟล์ `LightLogger.py` สำหรับบอร์ด Raspberry Pi 3 อ่านค่าความเข้มแสงจากบอร์ดวัดความเข้มแสง BH1750 ผ่านบัส I2C แล้วบันทึกเป็นแฟ้มข้อมูลลงในไฟล์ `BH1750.txt`

- (1) ใช้วงจรในรูปที่ 5-13 ในการทดสอบ
- (2) สร้างไฟล์สคริปต์ Python ด้วย Geany พิมพ์คำสั่งตามโปรแกรมที่ 5-6
- (4) บันทึกไฟล์ชื่อ **LightLogger.py**
- (5) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **sudo python3 LightLogger.py**
- (6) หาแหล่งกำเนิดแสงมาส่องไปยังตัวตรวจจับ ปรับความสว่างในระดับต่างๆ กัน ดูผลการทำงานผ่านหน้าต่างเทอร์มินอล

ทันทีที่โปรแกรมทำงาน ระบบจะทำการบันทึกค่าที่ตัวตรวจจับ BH1750 วัดได้ ลงในแฟ้มข้อมูลที่ชื่อ *BH1750.txt* ข้อมูลที่จัดเก็บประกอบด้วย วัน เวลา และค่าที่อ่านได้ในขณะนั้น โดยค่าของวันและเวลาได้มาจากคำสั่ง `datetime.now()` ซึ่งเป็นการดึงค่าเวลาล่าสุดของระบบมาใช้

รูปแบบการบันทึกข้อมูลแสดงในรูปที่ 5-17



```

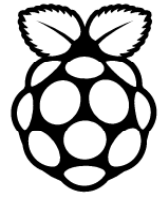
LxTerminal
File Edit Tabs Help
DateTime: 2015-08-17 17:37:47.806283 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:48.311484 Luminosity: 311.33 lx
DateTime: 2015-08-17 17:37:48.816364 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:49.321471 Luminosity: 310.33 lx
DateTime: 2015-08-17 17:37:49.826451 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:50.331404 Luminosity: 311.33 lx
DateTime: 2015-08-17 17:37:50.836314 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:51.341539 Luminosity: 310.33 lx
DateTime: 2015-08-17 17:37:51.846424 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:52.351385 Luminosity: 310.33 lx
DateTime: 2015-08-17 17:37:52.856279 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:53.361224 Luminosity: 310.33 lx
DateTime: 2015-08-17 17:37:53.866105 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:54.371067 Luminosity: 311.33 lx
DateTime: 2015-08-17 17:37:54.875951 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:55.380836 Luminosity: 311.33 lx
DateTime: 2015-08-17 17:37:55.885767 Luminosity: 314.33 lx
DateTime: 2015-08-17 17:37:56.390992 Luminosity: 311.33 lx
DateTime: 2015-08-17 17:37:56.895885 Luminosity: 313.33 lx

```

รูปที่ 5-17 ผลการทำงานของโปรแกรมที่ 5-6 ข้อมูลที่ได้จากการตรวจวัดความเข้มแสงของ BH1750 จะรวมกับข้อมูลของค่าวันเวลา จะถูกบันทึกลงไปไฟล์ *BH1750.txt* ซึ่งเป็นแฟ้มข้อมูลตัวอักษรหรือ text file

ภาคผนวก 1

Raspberry Pi 3 กับการพัฒนาโปรแกรม ภาษา Python ด้วยซอฟต์แวร์ Geany



ปกติแล้วในระบบปฏิบัติการของ Raspberry Pi จะมีโปรแกรมเท็กซ์เอดิเตอร์ที่ชื่อ Nano มาให้พร้อมใช้สำหรับเขียนโปรแกรม โดยมีลักษณะเป็นหน้าต่างแบบ DOS อาจไม่คุ้นเคยสำหรับผู้ใช้งานในสมัยใหม่ที่นิยมใช้เครื่องมือพัฒนาโปรแกรมในแบบกราฟิกหรือแบบที่เรียกว่า IDE (Integrated Development Environment) ที่รวบรวมคำสั่งและเครื่องมือต่างๆ ไว้ในหน้าต่างเดียว จึงขอแนะนำซอฟต์แวร์ที่ชื่อ **Geany** ซึ่งเป็นเท็กซ์เอดิเตอร์ที่รองรับคอมไพเลอร์หรือตัวแปลภาษาได้ทั้ง Python, C, C++ และในหนังสือชุด Raspberry Pi 3 ที่ INEX จัดทำขึ้นได้เลือกใช้ Geany เป็นหลักในการเป็นเท็กซ์เอดิเตอร์เพื่อเขียนและพัฒนาโปรแกรมเพื่อกำหนดการทำงานให้กับ Raspberry Pi

A1.1 รู้จักกับ Geany

Geany เป็นซอฟต์แวร์สำหรับเขียนข้อความหรือเท็กซ์เอดิเตอร์ (text editor) ที่บรรจุเครื่องมือต่างๆ ไว้พร้อมใช้งานภายใต้หน้าต่างเดียวกัน แต่มีขนาดของซอฟต์แวร์ที่ไม่ใหญ่เพื่อให้การประมวลผลทำได้อย่างรวดเร็ว ทั้งยังรองรับคอมไพเลอร์หรือตัวแปลภาษาได้อย่างกว้างขวาง

เว็บไซต์อย่างเป็นทางการของ Geany คือ <http://www.geany.org> ผู้สนใจใช้งานดาวน์โหลดมาใช้ได้โดยไม่มีค่าใช้จ่าย จึงมีนักพัฒนาที่นำ Geany มาติดตั้งคอมไพเลอร์ของภาษา Python สำหรับใช้กับบอร์ด Raspberry Pi เพื่อเขียนโปรแกรมควบคุมการทำงานด้วยภาษา Python ทั้งการประมวลผลทางซอฟต์แวร์และการติดต่อสั่งงานทางฮาร์ดแวร์ผ่านพอร์ตอินพุตเอาต์พุตเนกประสงค์หรือ GPIO (General Purpose Input Output) ของบอร์ด Raspberry Pi

สำหรับท่านที่จัดซื้อชุดเรียนรู้ Raspberry Pi 3 จาก INEX ในแผ่น microSD การ์ดได้ทำการติดตั้ง Geany ไว้พร้อมใช้งานบนระบบปฏิบัติการ Raspbian แล้ว

หากระบบปฏิบัติการ Raspbian ที่มีอยู่ไม่มีซอฟต์แวร์ Geany ติดตั้งไว้ ผู้ที่ต้องการใช้งานสามารถทำการติดตั้งได้เอง โดย

- (1) เชื่อมต่อบอร์ด Raspberry Pi 3 เข้ากับเครือข่ายอินเทอร์เน็ต
- (2) หลังจากบูตระบบเสร็จสิ้น ให้เข้าสู่หน้าต่าง **Terminal**

(3) ติดตั้งโปรแกรมด้วยการพิมพ์คำสั่ง

```
pi@raspberrypi ~/code $ sudo apt-get update
pi@raspberrypi ~/code $ sudo apt-get install geany
```

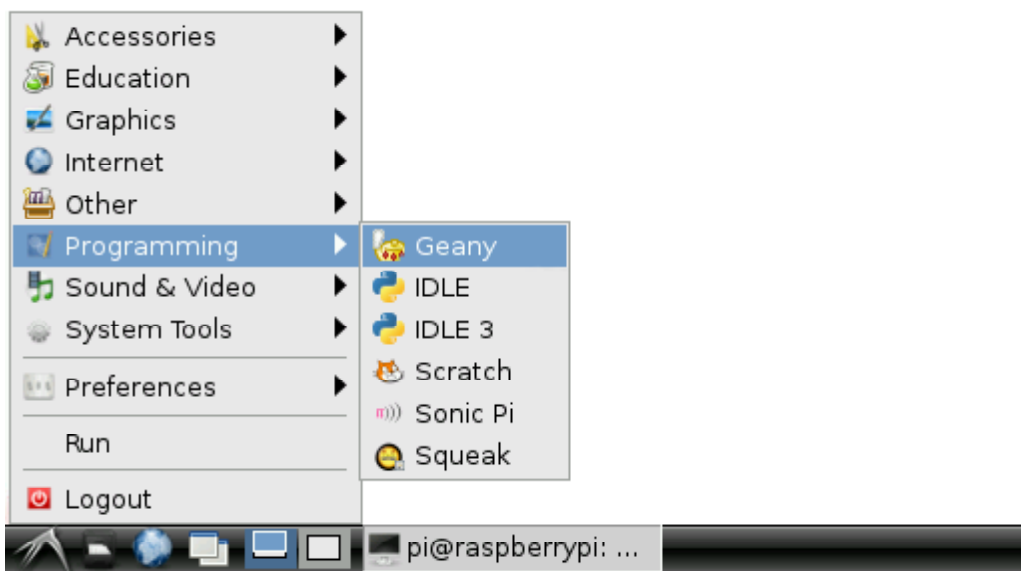
(4) จากนั้นรอนจนกระทั่งการติดตั้งโปรแกรมเสร็จสิ้น

A1.2 ส่วนประกอบของ Geany

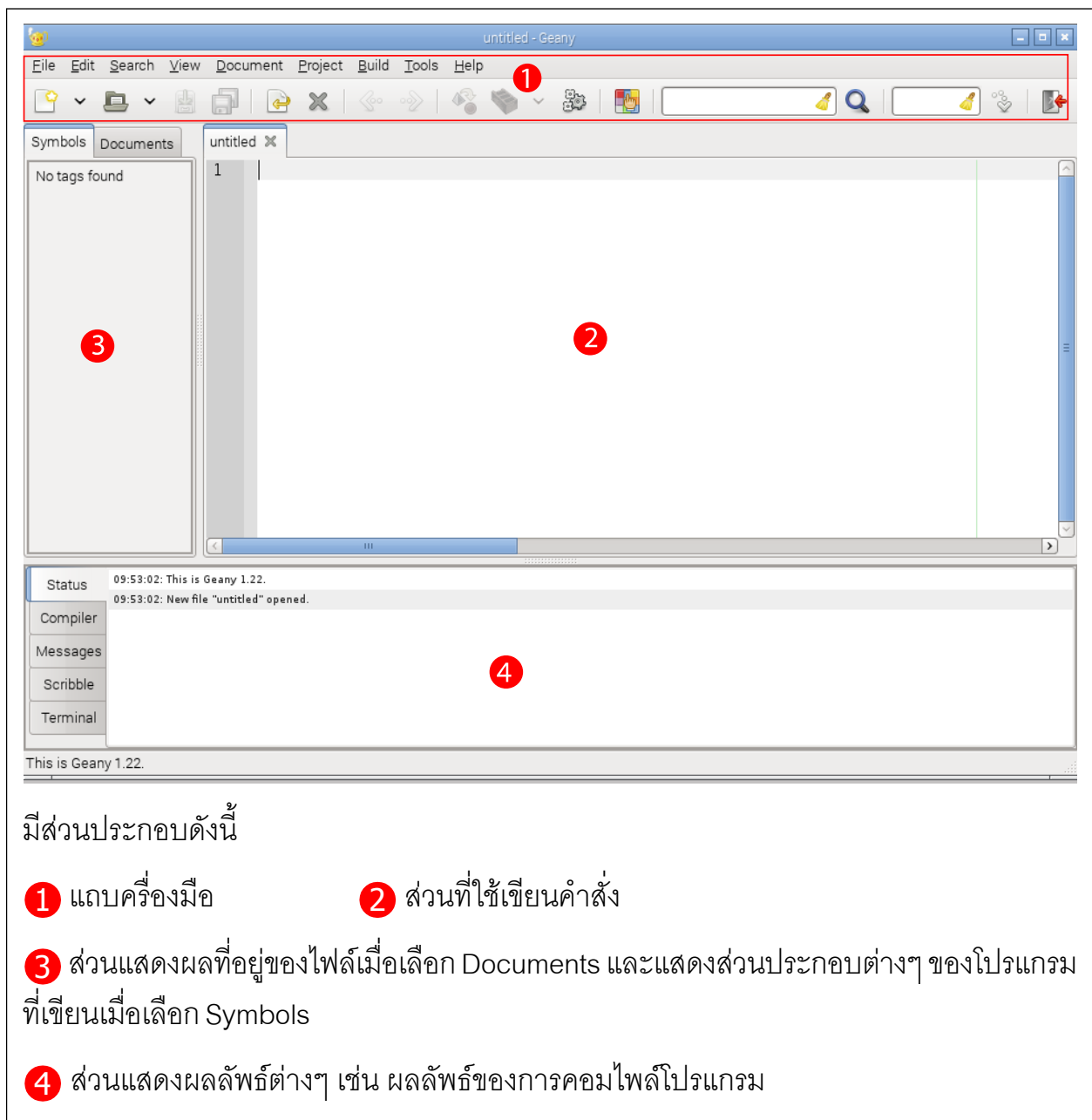
เมื่อติดตั้งโปรแกรมเสร็จแล้ว ทำการเรียกใช้งานระบบปฏิบัติการในแบบกราฟิก โดยการกลับออกไปยังเชลพร้อมพ์หลัก แล้วพิมพ์คำสั่ง **startx** เพื่อเลือกให้ทำงานบนระบบปฏิบัติการ Raspbian แบบกราฟิก จากนั้นค้นหาไอคอนของ Geany แล้วทำการดับเบิลคลิกเพื่อรันให้ทำงาน



หรือเลือกจากเมนู **Programming > Geany** ดังรูป



เมื่อเปิดโปรแกรมขึ้นมา หน้าต่างหลักของโปรแกรมเป็นดังรูปที่ A1-1



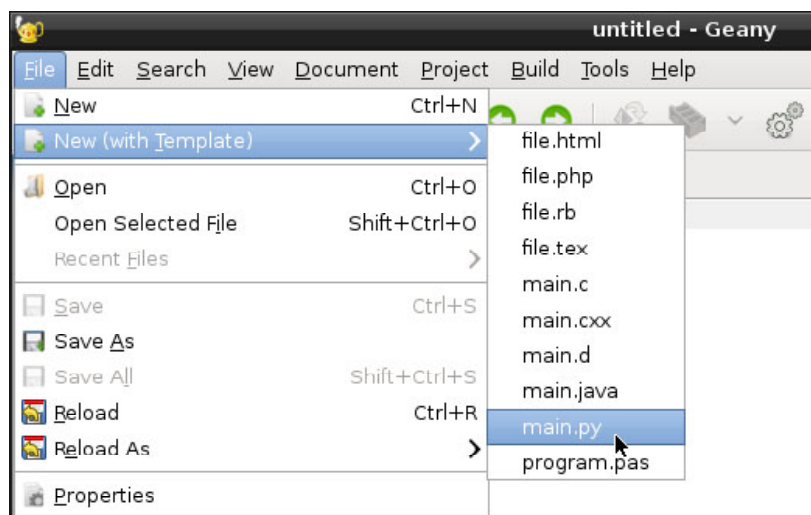
รูปที่ A1-1 ส่วนประกอบหน้าต่างหลักของ Geany ซอฟต์แวร์เท็กซ์เอดิเตอร์สำหรับพัฒนาโปรแกรมให้แก่ Raspberry Pi

A1.2.1 แอปเครื่องมือ



มีส่วนประกอบดังนี้

① สร้างไฟล์ใหม่โดยไม่ระบุนามสกุล ถ้าต้องการสร้างไฟล์ใหม่ที่จะระบุนามสกุลด้วย ให้เลือกที่รูปปลุกศรชี้ลงจะปรากฏเมนูให้เลือกดังรูป



② เปิดไฟล์ ถ้าต้องการเปิดไฟล์เดิม ให้เลือกที่รูปปลุกศรชี้ลง จะปรากฏเมนูขึ้นมาให้เลือก

③ บันทึกไฟล์ที่เลือกไว้

④ บันทึกไฟล์ทั้งหมดที่เปิด

⑤ เรียกคืนรูปแบบเดิมทั้งหมด จะใช้ในกรณีที่ยังไม่ได้บันทึกไฟล์

⑥ ปิดไฟล์ที่เลือกไว้

⑦ คอมไพล์ไฟล์สคริปต์

⑧ รันไฟล์สคริปต์

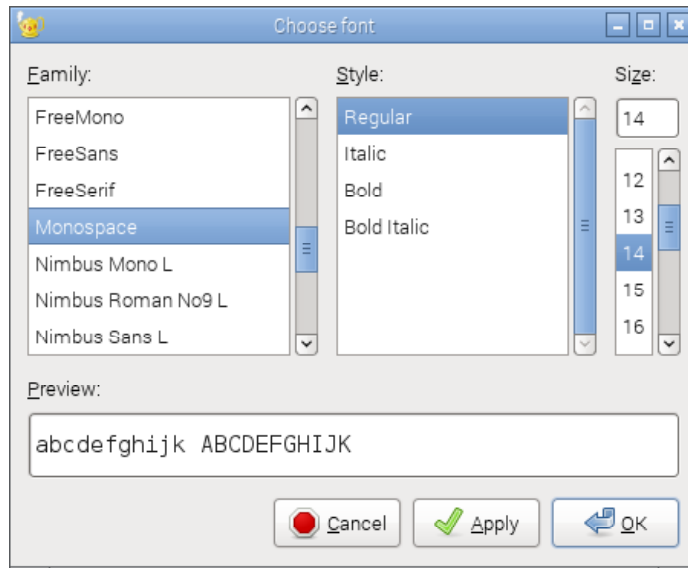
⑨ ค้นหาคำที่อยู่ในไฟล์

⑩ เลือกหมายเลขบรรทัดที่ต้องการดูคำสั่ง

⑪ ออกจากโปรแกรม

A2.2 การเลือกรูปแบบตัวอักษร

ผู้พัฒนาโปรแกรมสามารถเลือกรูปแบบหรือฟอนต์ (font) ของตัวอักษรที่แสดงผลบนพื้นที่เขียนคำสั่ง โดยเลือกไปที่เมนู **View > Change Font** จากนั้นจะปรากฏหน้าต่างสำหรับตั้งค่า ดังรูป

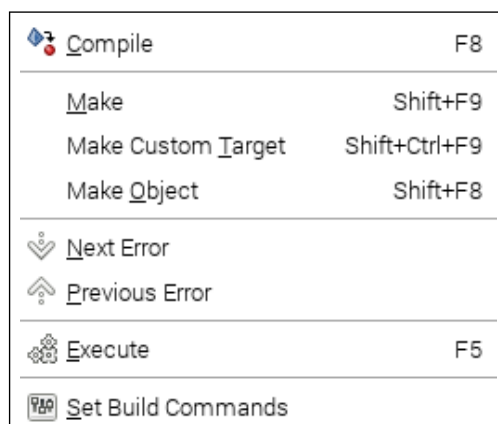


เมื่อตั้งค่าแล้ว คลิกปุ่ม **OK** เพื่อยืนยัน

A1.2.3 ตั้งค่าการคอมไพล์และรันโปรแกรมหรือเอ็กซิกิวต์

เนื่องจากภาษา Python ที่ใช้กับบอร์ด Raspberry Pi 3 ในขณะจัดทำหนังสือเล่มนี้มี 2 เวอร์ชันหลักคือ Python2 และ Python3 ผู้พัฒนาโปรแกรมอาจมีโค้ดของภาษา Python ทั้ง 2 เวอร์ชัน หากทำการคอมไพล์ (compile) และเอ็กซิกิวต์ (execute) หรือรันโปรแกรมไม่ตรงเวอร์ชัน อาจทำให้ไฟล์สคริปต์ทำงานไม่ได้ หรือคอมไพล์ไม่ผ่าน จึงขอแนะนำการเพิ่มเมนู เพื่อให้ผู้ใช้งาน Geany ได้มีทางเลือกในการคอมไพล์และเอ็กซิกิวต์หรือรันโปรแกรมดังนี้

(1) ไปที่ **Build** จะปรากฏเมนูให้เลือกดังรูป



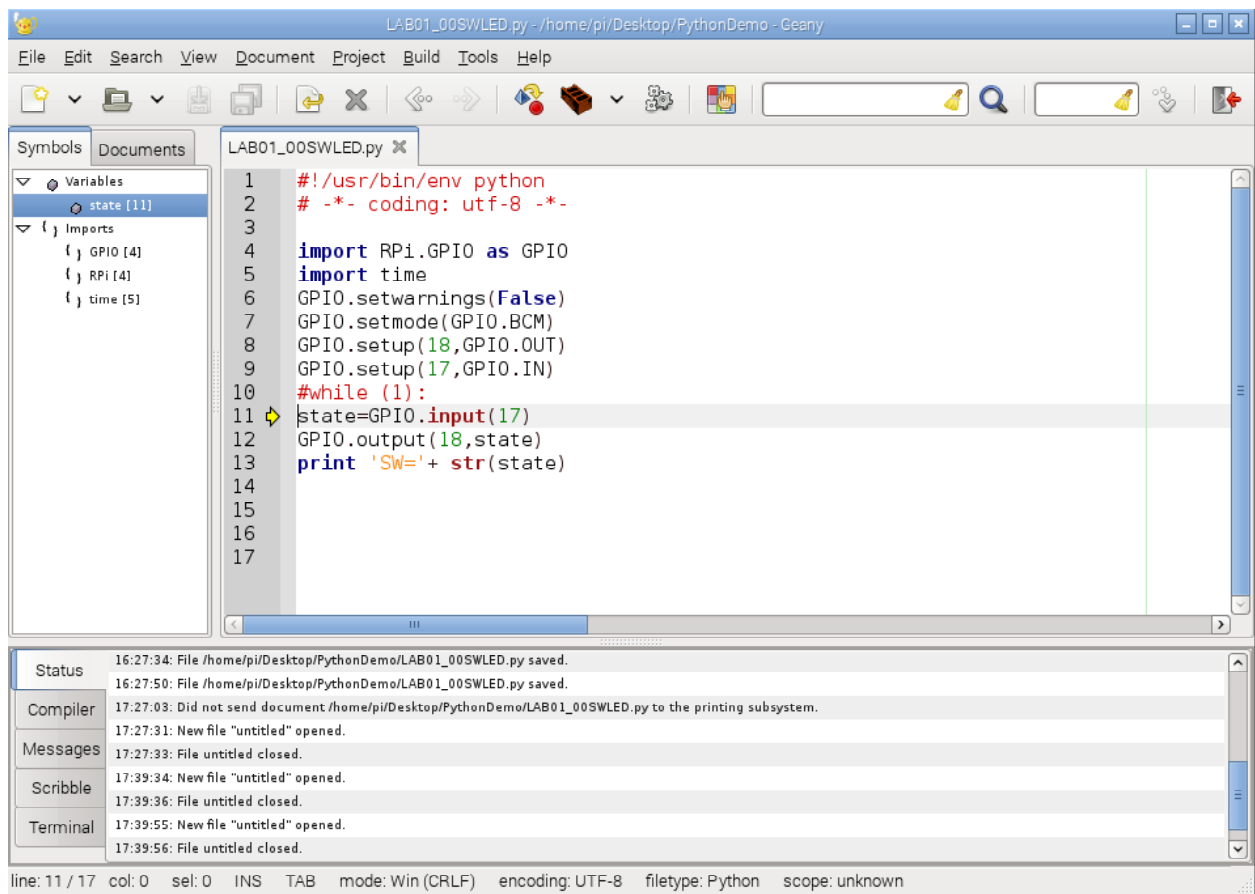
จะเห็นว่ามีคำสั่ง **Compile** และ **Execute** ให้เลือก โดยทั้งสองคำสั่งใช้สำหรับ Python3

(2) จากนั้นให้คลิกที่ **Set Build Commands** จะปรากฏตารางขึ้นมา ให้ทำการเพิ่ม **Label** และ **Command** ดังรูป แล้วคลิก **OK**

#	Label	Command	Working directory	Reset
Python commands				
1.	<u>C</u> ompile	python3 -m py_compile "%f"		
2.	<u>C</u> ompile Python2	python -m py_compile "%f"		
3.				
Error regular expression:				
Independent commands				
1.	<u>M</u> ake	make		
2.	M <u>a</u> ke Custom <u>T</u> arget	make		
3.	M <u>a</u> ke <u>O</u> bject	make %e.o		
4.				
Error regular expression:				
<i>Note: Item 2 opens a dialog and appends the response to the command.</i>				
Execute commands				
1.	<u>E</u> xecute	sudo python3 "%f"		
2.	E <u>x</u> ecute Python2	sudo python "%f"		
%d, %e, %f, %p are substituted in command and directory fields, see manual for details.				
		Cancel	OK	

(3) จากนั้นไปที่ **Build** จะเห็นว่า มีเมนูเพิ่มขึ้นมาจากเดิมดังรูป

	<u>C</u> ompile	F8
	<u>C</u> ompile Python2	F9
	<u>M</u> ake	Shift+F9
	M <u>a</u> ke Custom <u>T</u> arget	Shift+Ctrl+F9
	M <u>a</u> ke <u>O</u> bject	Shift+F8
	<u>N</u> ext Error	
	<u>P</u> revious Error	
	<u>E</u> xecute	F5
	E <u>x</u> ecute Python2	
	<u>S</u> et Build Commands	



รูปที่ A1-2 แสดงตัวอย่างหน้าต่างของ Geany เมื่อมีการเปิดไฟล์ของโปรแกรมภาษา Python ขึ้นมาใช้งาน

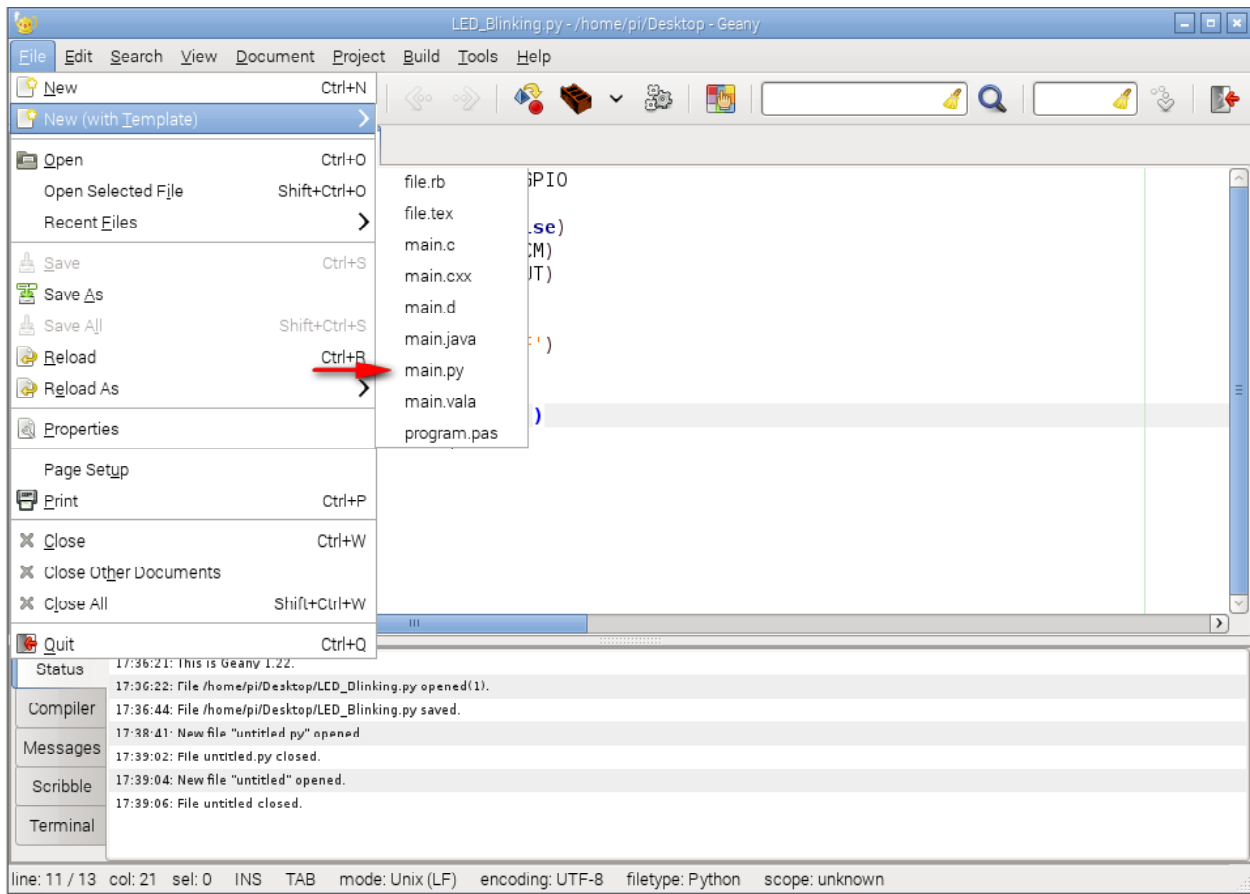
นับจากนี้ผู้พัฒนาจะสามารถเลือกคอมไพล์และเอ็กซิคิวต์หรือรันโปรแกรมภาษา Python ได้ทั้ง Python 2 และ Python 3 ผ่านทางเมนู Build ได้ตามต้องการ

อนึ่ง โปรแกรมตัวอย่างของภาษา Python ที่พัฒนาขึ้นเพื่อควบคุมการทำงานของ Raspberry Pi 3 ในหนังสือเล่มนี้นับจากบทนี้จะใช้ Geany เป็นเท็กซ์เอดิเตอร์หลักและใช้ Python3 ในการพัฒนาโปรแกรม

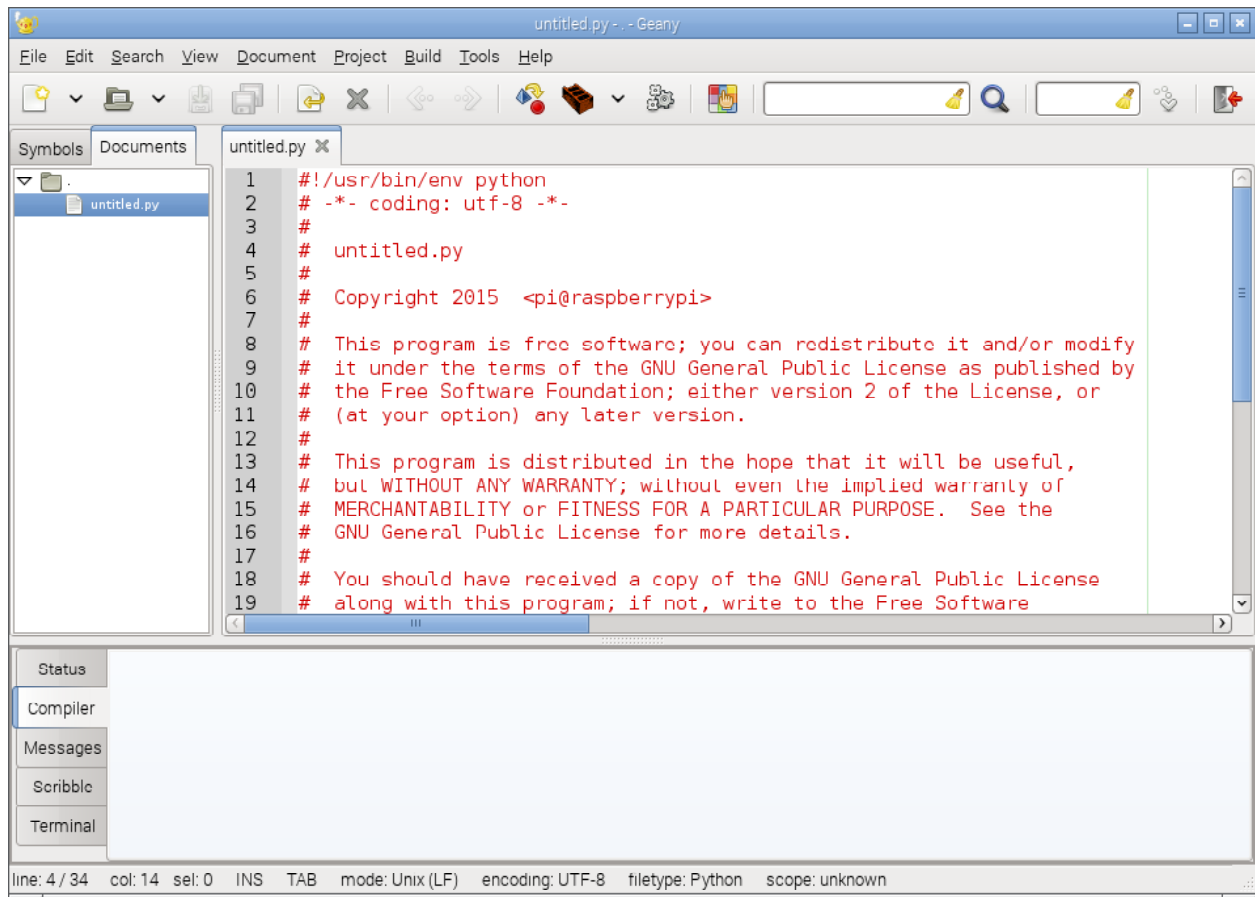
A1.3 ตัวอย่างการทดสอบโปรแกรมภาษา Python ด้วย Geany

ในการเขียนโปรแกรมภาษา Python สำหรับ Raspberry Pi 3 ในที่นี้ขอแนะนำให้ใช้ Python 3 เป็นหลัก สำหรับการพัฒนาโปรแกรมบน Geany เบื้องต้น มีขั้นตอนดังนี้

(1) สร้างไฟล์สคริปต์ใหม่ โดยไปที่ **File > New(With_Template)** แล้วคลิกที่ **main.py** ดังรูป



(2) จะปรากฏไฟล์สคริปต์ที่มีชื่อว่า **untitled.py** ดังรูป



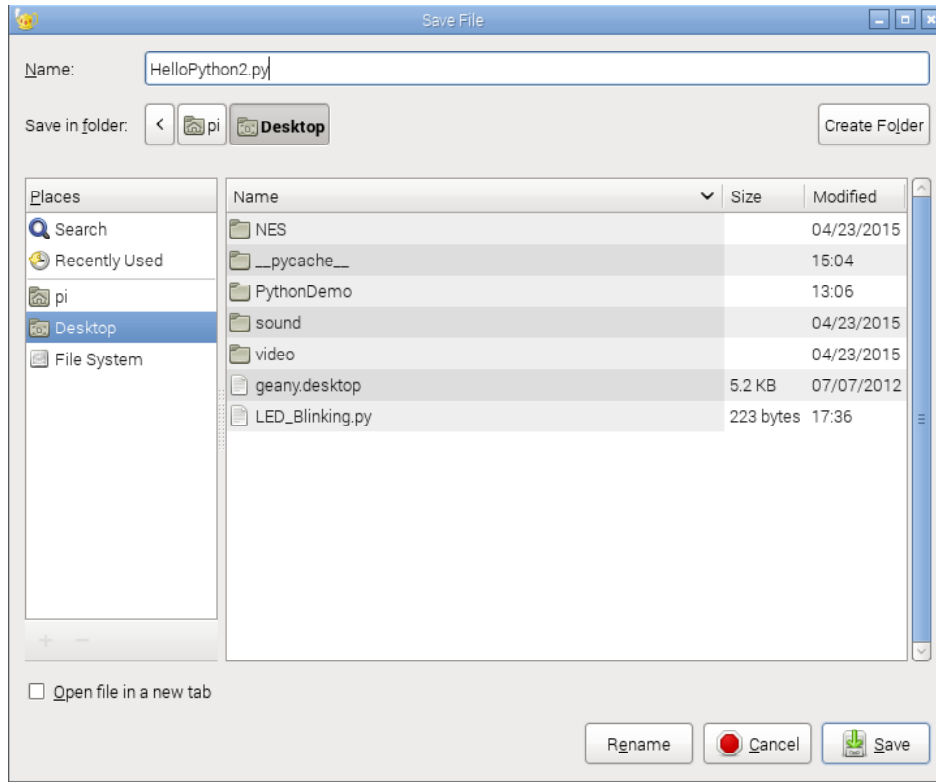
(3) พิมพ์คำสั่งต่อไปนี้

```

print ('Python',3)
print ('Hello','World!')
print ("Hello","World!")

```

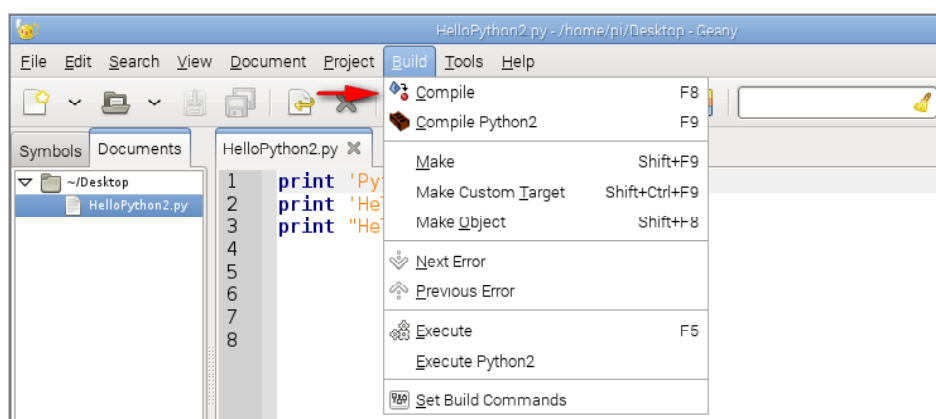
(4) บันทึกไฟล์สคริปต์ โดยไปที่ **File > Save** เลือกที่อยู่ของ File แล้วตั้งชื่อว่า **HelloPython3.py** ในที่นี้จะเก็บไฟล์ไว้ที่ **Desktop** แล้วคลิก **Save**



(5) ทดสอบคอมไพล์และเอ็กซิกิวต์ คำสั่ง **print** ที่พิมพ์ลงในไฟล์สคริปต์เป็นคำสั่งแสดงข้อความออกทางหน้าจอภาพ รูปแบบคำสั่ง **print** ของ Python3 และ 2 แตกต่างกัน ในที่นี้จะใช้ Python3 เป็นหลัก ดังนั้นหากผู้พัฒนาโปรแกรมมีตัวอย่างโปรแกรมเดิมของ Python2 เมื่อนำมาคอมไพล์หรือเอ็กซิกิวต์ด้วย Python3 จะต้องทำการแก้ไขคำสั่งให้ถูกต้องก่อน โดยศึกษาได้จาก help ของ Python3

(5.1) เริ่มต้นด้วยการคอมไพล์

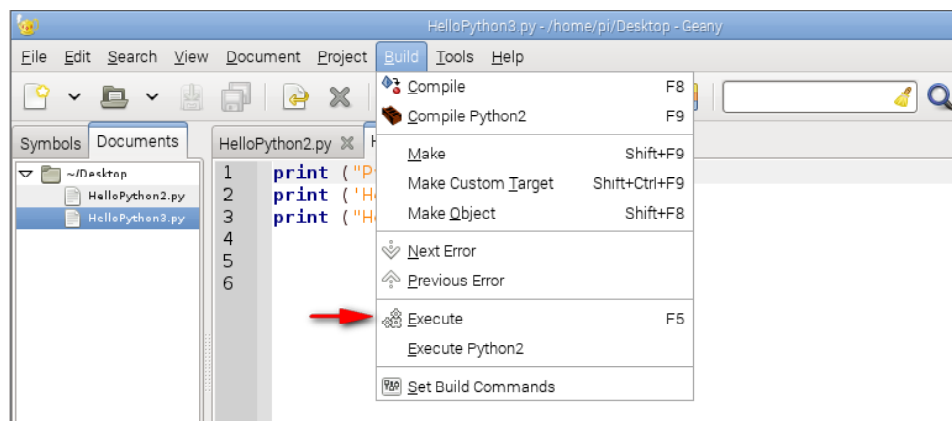
(5.1.1) ไปที่ **Build** จะปรากฏเมนูดังรูป



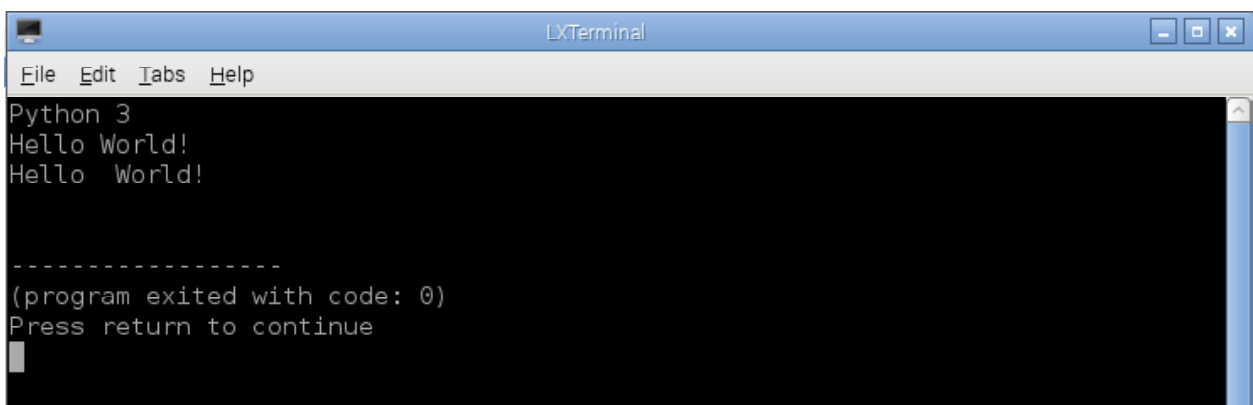
(5.1.2) คลิกที่ **Compile** เพื่อใช้ Python3 คอมไพเลอร์ในการคอมไพล์ จากนั้นผลลัพธ์ของการคอมไพล์จะปรากฏที่กรอบด้านล่างของหน้าต่างโปรแกรม โดยแสดงข้อความว่า **Compilation finished successfully** แสดงว่ารูปแบบคำสั่งถูกต้องพร้อมที่จะทำการเอ็กซิกิวต์หรือรันต่อไป

(5.2) การเอ็กซิกิวต์หรือสั่งให้รันโปรแกรม

(5.2.1) ไปที่ **Build** ทำการเลือกเอ็กซิกิวต์ด้วย Python3 โดยคลิกเลือกที่ **Execute** ดังรูป



(5.2.2) ผลลัพธ์ของการเอ็กซิกิวต์แสดงดังรูป



นี่คือตัวอย่างการใช้งาน Geany ในการพัฒนาโปรแกรมภาษา Python ให้แก่บอร์ด Raspberry Pi โดยตัว Geany เองรองรับและทำงานร่วมกับคอมไพเลอร์ได้หลายตัว ทั้ง Python2, Python3 สำหรับภาษา Python หากเป็นภาษา C คอมไพเลอร์ที่นิยมสำหรับผู้เริ่มต้นคือ WiringPi

สำหรับการเรียนรู้เพื่อเขียนโปรแกรมควบคุมการทำงานของบอร์ด Raspberry Pi 3 ในหนังสือเล่มนี้เลือกใช้ภาษา Python และเป็นรุ่น Python3 โดยได้ทำการผนวกคอมไพเลอร์ไว้ใน Geany ตั้งแต่ขั้นตอนการสร้างอิมเมจไฟล์สำหรับระบบปฏิบัติการ Raspbian ที่ทาง INEX จัดทำขึ้น ทั้งนี้เพื่ออำนวยความสะดวกในการเรียนรู้และพัฒนาโปรแกรม

