Raspberry Pi 2

บอร์ดคอมพิวเตอร์ 32 บิตยอดนิยม

รู้จักและการใช้งานเบื้องต้น

ผศ.โอภาส ศิริครรชิตถาวร, สมเกียรติ กิจวงศ์วัฒนะ, นวพร เหล่าวัฒนธรรม,
กฤษดา ใจเย็น, วรพจน์ กรแก้ววัฒนกุล, ธีรวุฒิ จิตพรมมา

inex™
Innovative Experiment
www.inex.co.th

A4

Raspberry Pi 2

บอร์ดคอมพิวเตอร์ 32 บิต

รู้จักและการใช้งานเบื้องต้น

ผศ.โอภาส ศิริครรชิตถาวร

วิทยาลัยเทคโนโลยีอุตสาหกรรม

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

สมเกียรติ กิจวงศ์วัฒนะ

นวพร เหล่าวัฒนธรรม

กฤษดา ใจเย็น

วรพจน์ กรแก้ววัฒนกุล

ธีรวุธ จิตพรมมา

Raspberry Pi 2
บอร์ดคอมพิวเตอร์ 32 บิต : รู้จักและการใช้งานเบื้องต้น

ผศ.โอภาส ศิริครรชิตถาวร, สมเกียรติ กิจวงศ์วัฒนะ, นวพร เหล่าวัฒนธรรม, กฤษดา ใจเย็น, วรพจน์ กรแก้ววัฒนกุล, ธีรวุธ จิตพรมมา

ใครควรใช้หนังสือเล่มนี้

1. นักเรียน นิสิต นักศึกษา และบุคคลทั่วไปที่มีความสนใจใช้งานบอร์ดคอมพิวเตอร์ 32 บิตขนาดเล็กสมัยใหม่ที่ติดตั้งระบบปฏิบัติการที่เหมาะสมสำหรับการใช้งานเป็น Embedded Computer หรือ Embedded PC ที่มีประสิทธิภาพสูง

2. สถาบันการศึกษา โรงเรียน วิทยาลัย มหาวิทยาลัย ที่มีการเปิดการเรียนการสอนวิชาอิเล็กทรอนิกส์หรือภาควิชาวิศวกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์

3. คณาจารย์ที่มีความต้องการศึกษา และเตรียมการเรียนการสอนเกี่ยวกับระบบสมองกลฝังตัวทั้งในระดับเบื้องต้นจนถึงขั้นก้าวหน้าในระดับอาชีวศึกษา และปริญญาตรี

ดำเนินการจัดพิมพ์และจำหน่ายโดย
บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด
108 ซ.สุขุมวิท 101/2 ถ.สุขุมวิท แขวงบางนา เขตบางนา กรุงเทพฯ 10260
โทรศัพท์ 0-2747-7001-4
โทรสาร 0-2747-7005

คำนำ

Raspberry Pi คือบอร์ดไมโครคอมพิวเตอร์แบบแผ่นเดี่ยวที่โด่งดังที่สุดเท่าที่เคยมีมา มันคือบอร์ดคอมพิวเตอร์ซึ่งผลไม้ที่บรรจุความสามารถไว้อย่างมากมาย รองรับระบบปฏิบัติการ Linux สำหรับการพัฒนาไปสู่บอร์ด Embedded Linux ในราคาประหยัด แต่อุดมด้วยความสามารถขั้นเทพมาพร้อมจุดเชื่อมต่ออุปกรณ์อินพุตเอาต์พุตทั้งผ่านพอร์ต USB, LAN, HDMI, ช่องสัญญาณภาพ และ GPIO สำหรับต่อกับวงจรหรืออุปกรณ์อิเล็กทรอนิกส์

Raspberry Pi เป็นบอร์ดคอมพิวเตอร์ 32 บิตที่ได้รับความนิยมสูงเป็นประวัติการณ์ของวงการระบบสมองกลฝังตัวโลก ด้วยยอดจำหน่ายมากกว่า 1 ล้านบอร์ดในเวลาไม่ถึงหนึ่งปีหลังจากที่ออกจำหน่ายครั้งแรกในวันที่ 29 กุมภาพันธ์ ค.ศ. 2012 และนับถึงวันที่จัดทำหนังสือเล่มนี้มียอดจำหน่ายไปมากกว่า 5 ล้านชิ้น (นับถึงวันครบรอบ 3 ปีของการก่อกำเนิดบอร์ด Raspberry Pi ในวันที่ 28 กุมภาพันธ์ ค.ศ. 2015) Raspberry Pi มีดีอย่างไร อะไรเป็นปัจจัยหลักที่ทำให้บอร์ดคอมพิวเตอร์ 32 บิตขนาดเล็กนี้โด่งดังไปทั่วโลก

ประเด็นหลักคือ *คุณสมบัติทางเทคนิคที่ยอดเยี่ยมที่มาพร้อมกับราคาที่ไม่แพง* ราคาของ Raspberry Pi ที่เปิดจำหน่ายในอังกฤษคือ US$35 หรือ GBP25 (ราคาขายในไทยประมาณ 1,605 ถึง 2,140 บาท รวมภาษีและค่าส่งแล้ว) นั่นทำให้นักเล่นบอร์ดคอมพิวเตอร์แทบทุกรายให้ความสนใจในทันที **ด้านคุณสมบัติทางเทคนิค** ด้วยการใช้ชิปควบคุมหลักเป็นชิปของ Broadcom เบอร์ BCM2835 ใน Raspberry Pi รุ่นแรกที่รวมซีพียู, หน่วยประมวลกราฟิกหรือ GPU และหน่วยความจำ SDRAM ไว้ภายในตัวถังเดียวกัน ทำให้นักเล่นทั้งหลายถึงกับตะลึง เพราะใน BCM2835 บรรจุซีพียู ARM11 ความเร็ว 700MHz, หน่วยประมวลกราฟิกหรือ GPU รุ่น Broadcom VideoCore IV ที่รองรับการแสดงผลผ่านจอภาพแบบ HDMI ด้วยความละเอียดในระดับ HD 1080p และมีหน่วยความจำ SDRAM มากถึง 512MB (เพิ่มเป็น 1GB ในรุ่น Raspberry Pi 2) นั่นหมายความว่า ไม่มีบอร์ดคอมพิวเตอร์ขนาดเล็กรุ่นใดในโลกที่มีขีดความสามารถสูงเท่านี้ ภายใต้ราคาที่แสนถูกขนาดนี้

เท่านั้นยังไม่พอ Raspberry Pi ยังมีจุดต่อ USB 2.0 จำนวน 2 พอร์ต (เพิ่มเป็น 4 พอร์ตในรุ่น Raspberry Pi B+ และ 2) รองรับการต่อเมาส์ คีย์บอร์ด และอุปกรณ์ USB อื่นๆ, แจ๊ก AV และ HDMI เพื่อต่อกับโทรทัศน์หรือจอแสดงผลที่มีจุดต่อแบบ RCA ตัวเมียหรือ HDMI, จุดต่ออีเธอร์เน็ตหรือ LAN, คอนเน็กเตอร์หรือจุดต่อพอร์ตอินพุตเอาต์พุต (General Purpose Input/Output : GPIO) ที่มีขาต่อบัส SPI (Serial Peripheral Interface Bus), I2C, I2S, ขาสัญญาณรับส่งข้อมูลอนุกรมหรือ UART และซ็อกเก็ตของ SD การ์ดสำหรับเสียบ SD การ์ดที่ติดตั้งระบบปฏิบัติการไว้แล้ว

นั่นคือ Raspberry Pi สามารถใช้งานเป็นคอมพิวเตอร์ตัวน้อยเพื่อการเขียนโปรแกรมทั่วๆ ไปก็ได้ หรือจะใช้เป็นบอร์ดควบคุมตัวเก่งในระบบสมองกลฝังตัวที่มีการเชื่อมต่อกับวงจรหรืออุปกรณ์อิเล็กทรอนิกส์ก็ได้

ผู้ผลิต Raspberry Pi คือ **มูลนิธิ** Raspberry Pi *(Raspberry Pi Foundation – www.raspberrypi.org)* ยังพัฒนาบอร์ดอย่างต่อเนื่อง เป็นรุ่น Raspberry Pi 2 ที่เปลี่ยนซีพียูเป็นเบอร์ BCM2836 อันเป็นชิป ARM Cortex-A7 ที่มี 4 แกน ความเร็วในการประมวลผลสูงขึ้นถึง 900MHz และเพิ่มหน่วยความจำแรมเป็น 1GB ทำให้ Raspberry Pi 2 ทำงานเร็วขึ้นถึง 6 เท่า ด้วยราคาจำหน่ายเดิม

ระบบปฏิบัติการ Linux เป็นตัวช่วยสำคัญของ Raspberry Pi โดยติดตั้งลงใน SD การ์ดที่ทำหน้าที่เสมือนเป็นฮาร์ดดิสก์ของคอมพิวเตอร์ ทำให้ Raspberry Pi กลายเป็นคอมพิวเตอร์สมบูรณ์แบบขนาดเล็กที่ใครๆ ก็ใช้งาน นับเป็นการปฏิวัติวงการ Embedded PC ครั้งใหญ่ที่สุดในประวัติศาสตร์

เพราะราคาของ Raspberry Pi ถูกและมีขีดความสามารถสูง ทำให้มีโปรแกรมเมอร์มือสมัครเล่นและชั้นเซียนเข้าถึงได้มาก ต่างก็ช่วยกันพัฒนาโปรแกรมประยุกต์ออกมาอย่างมากมาย รวมถึงซอฟต์แวร์ที่ใช้ในการเรียนรู้การเขียนโปรแกรมอย่างง่ายอย่าง Scratch ส่งผลให้เยาวชนสามารถเข้าถึงการเขียนโปรแกรมและใช้งานคอมพิวเตอร์ได้อย่างกว้างขวาง ไม่เพียงเท่านั้น Raspberry Pi ยังเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตได้อย่างง่ายดาย มีเว็บบราวเซอร์ให้เข้าถึงเว็บไซต์ต่างๆ ได้ ทั้งยังทำงานร่วมกันบนเครือข่ายได้สะดวก ทั้งหมดนี้เราพบเห็นได้ทั่วไปกับคอมพิวเตอร์สมัยใหม่แทบทุกตัว แต่ไม่ใช่ที่ราคาต่ำกว่า 2,500 บาท

ดังนั้น จึงเชื่อได้อย่างสนิทใจว่า Raspberry Pi คือหนึ่งในบอร์ดคอมพิวเตอร์ที่ผู้สนใจในระบบสมองกลฝังตัวสมัยใหม่ส่วนใหญ่ต้องเรียนรู้และใช้งานให้เป็น หนังสือเล่มนี้จึงเกิดขึ้นเพื่อเป็นตัวช่วยสำหรับผู้สนใจเริ่มต้นใช้งานบอร์ดคอมพิวเตอร์ตัวเก่งและยอดนิยมตัวนี้

ชัยวัฒน์ ลิ้มพรจิตรวิไล

บรรณาธิการ

คำชี้แจงจากคณะผู้เขียน/เรียบเรียง

การนำเสนอข้อมูลเกี่ยวกับข้อมูลทางเทคนิคและเทคโนโลยีในหนังสือเล่มนี้ เกิดจากความต้องการที่จะอธิบายกระบวนการและหลักการทำงาน ของอุปกรณ์ในภาพรวมด้วยถ้อยคำที่ง่ายเพื่อสร้างความเข้าใจแก่ผู้อ่าน ดังนั้นการแปลคำศัพท์ทางเทคนิคหลายๆ คำอาจไม่ตรงตามข้อบัญญัติของราชบัณฑิตยสถาน และมีหลายๆ คำที่ยังไม่มีการบัญญัติอย่างเป็นทางการ คณะผู้เขียนจึงขออนุญาตบัญญัติศัพท์ขึ้นมาใช้ในการอธิบาย โดยมีข้อจำกัดเพื่ออ้างอิงในหนังสือเล่มนี้เท่านั้น

ทั้งนี้สาเหตุหลักของข้อชี้แจงนี้มาจาก การรวบรวมข้อมูลของอุปกรณ์ในระบบสมองกลฝังตัวและเทคโนโลยีหุ่นยนต์สำหรับการศึกษาเพื่อนำมาเรียบเรียงเป็นภาษาไทยนั้นทำได้ไม่ง่ายนัก ทางคณะผู้เขียนต้องทำการรวบรวมและทดลองเพื่อให้แน่ใจว่า ความเข้าใจในกระบวนการทำงานต่างๆ นั้นมีความคลาดเคลื่อนน้อยที่สุด

เมื่อต้องทำการเรียบเรียงออกมาเป็นภาษาไทย ศัพท์ทางเทคนิคหลายคำมีความหมายที่ทับซ้อนกันมาก การบัญญัติศัพท์จึงเกิดจากการปฏิบัติจริงร่วมกับความหมายทางภาษาศาสตร์ ดังนั้นหากมีความคลาดเคลื่อนหรือผิดพลาดเกิดขึ้น ทางคณะผู้เขียนขอน้อมรับและหากได้รับคำอธิบายหรือชี้แนะจากท่านผู้รู้จะได้ทำการชี้แจงและปรับปรุงข้อผิดพลาดที่อาจมีเหล่านั้นโดยเร็วที่สุด

ทั้งนี้เพื่อให้การพัฒนาสื่อทางวิชาการ โดยเฉพาะอย่างยิ่งกับความรู้ของเทคโนโลยีสมัยใหม่สามารถดำเนินไปได้อย่างต่อเนื่อง ภายใต้การมีส่วนร่วมของผู้รู้ในทุกภาคส่วน

บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

สารบัญ

บทที่ 1 Raspberry Pi รายงานตัว....7

บทที่ 2 เตรียมการใช้งาน Raspberry Pi 2....19

บทที่ 3 Raspberry Pi กับการเชื่อมต่อระบบเครือข่ายไร้สาย WiFi และการควบคุมผ่าน SSH....49

บทที่ 4 คำสั่งพื้นฐานบนเทอร์มินอลของระบบปฏิบัติการ Linux....69

บทที่ 5 เริ่มต้นรู้จักกับภาษา Python....91

บทที่ 6 Raspberry Pi 2 กับการพัฒนาโปรแกรมภาษา Python ด้วยซอฟต์แวร์ Geany....121

บทที่ 7 การติดต่อพอร์ตอินพุตเอาต์พุตอเนกประสงค์ หรือ GPIO ของ Raspberry Pi 2....129

บทที่ 8 Raspberry Pi 2 กับการขับโหลดกระแสไฟฟ้าสูงด้วยรีเลย์....165

บทที่ 9 เชื่อมต่อไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลผ่านบัส SPI....179

บทที่ 10 Raspberry Pi 2 กับการติดต่อตัวตรวจจับแบบดิจิตอล....199

- โมดูลตรวจจับความเคลื่อนไหว ZX-PIR
- โมดูลตรวจจับแสง BH1750
- โมดูลวัดความชื้นและอุณหภูมิ DHT11

บทที่ 11 Raspberry Pi 2 กับโมดูลกล้อง....221

บทที่ 1
Raspberrry Pi รายงานตัว

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

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

1.1 จุดกำเนิดของ Rasapberry Pi

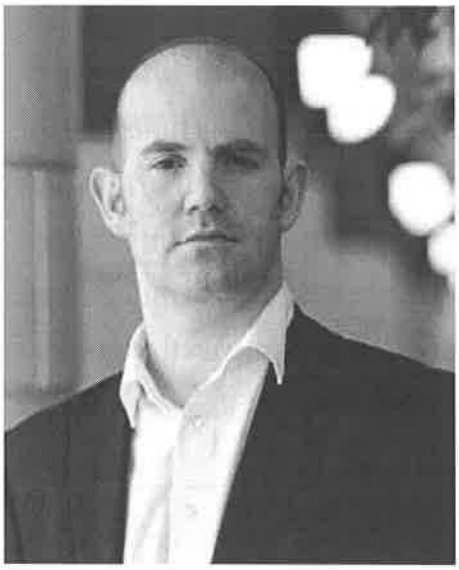
Raspberry Pi ถูกสร้างขึ้นภายใต้แนวคิดที่ไม่ซับซ้อนว่า ***"ต้องการให้เด็กๆ หัดเขียนโปรแกรมด้วยเครื่องมือที่ราคาถูก"*** แนวคิดนี้มาจาก**มูลนิธิ Raspberry Pi (www.raspberrypi.org)** โดยในตอนแรกไม่ได้คิดถึงการสร้างคอมพิวเตอร์ขนาดเล็กเลยด้วยซ้ำ ทางที่จะทำให้เด็กๆ สนใจและหัดเขียนโปรแกรมคือหาเครื่องมือให้พวกเขาในแบบฟรีหรือถูกที่สุดเท่าที่จะเป็นไปได้ Raspberry Pi จึงถูกสร้างขึ้นโดย ***Eben Upton*** และคณะทำงานในมูลนิธิ Raspberry Pi มีการแจกจ่ายแก่ผู้เรียนในสาขาวิทยาการคอมพิวเตอร์ของมหาวิทยาลัยเคมบริดจ์ สหราชอาณาจักร เพื่อเรียนรู้และระดมสมอง โดยมีเป้าหมายไปสู่การพัฒนาสื่อเพื่อช่วยให้เด็กๆ สนใจในการเขียนโปรแกรม นั่นเป็นจุดเริ่มต้นที่ทำให้มีคนรู้จัก Raspberry Pi แต่ก็ยังจำกัดอยู่ในแวดวงการศึกษาระดับมหาวิทยาลัยเท่านั้น

ย้อนหลังไปในปี ค.ศ. 2006 ต้นแบบแรกของ Raspberry Pi ใช้ไมโครคอนโทรลเลอร์ ATmega644 ของ Atmel โดยมีการเผยแพร่วงจรและแบบลายทองแดงของแผ่นวงจรพิมพ์สู่สาธารณะ ด้วยการดำเนินการในลักษณะนี้ Eben Upton ผู้ให้กำเนิด Raspberry Pi สามารถรวบรวมกลุ่มของครู อาจารย์ นักวิชาการและผู้เชี่ยวชาญด้านคอมพิวเตอร์ทั้งด้านซอฟต์แวร์และฮาร์ดแวร์ เพื่อแลกเปลี่ยนข้อมูล ข้อคิดเห็น จนนำไปสู่การเลือกชิปประมวลผลตัวใหม่โดยได้รับแรงบันดาลใจจากเครื่องคอมพิวเตอร์ BBC Micro ของ Acorn ซึ่งเป็นผู้ผลิตชิป ARM *เดิม Raspberry Pi ถูกตั้งชื่อว่า ABC computer* เนื่องจากต้องการสื่อสารทางการตลาดว่า *นี่คือคอมพิวเตอร์เพื่อการเริ่มต้นเทียบได้กับหนังสือเรียนภาษาอังกฤษ ABC นั่นเอง*

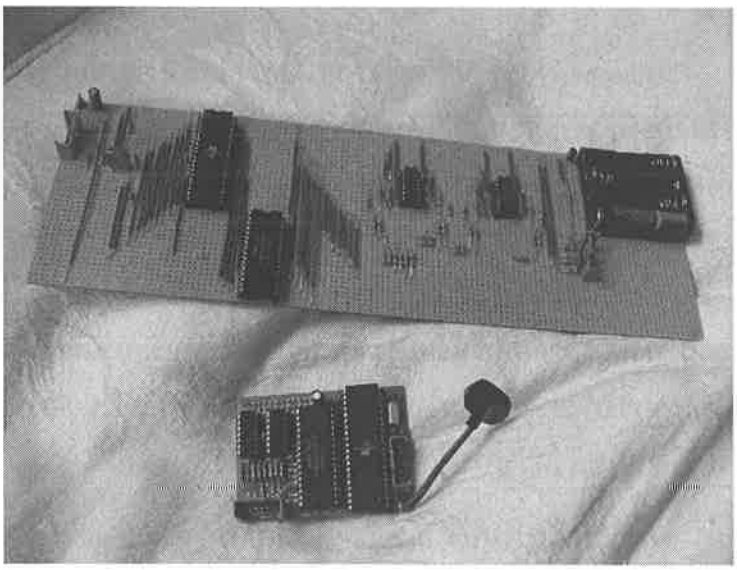
จากนั้น Upton ได้ผลิตบอร์ดคอมพิวเตอร์ต้นแบบตัวใหม่ออกมาโดยใช้ ARM โปรเซสเซอร์ มีขนาดเท่ากับ USB แฟลชไดรฟ โดยมีจุดต่อพอร์ต USB 1 ช่อง และพอร์ตต่อจอภาพ HDMI อีก 1 ช่อง นั่นแสดงให้เห็นถึงวิสัยทัศน์ของ Upton ที่เชื่อว่า จอแสดงผลความละเอียดสูงที่ใช้ช่องทางการสื่อสารแบบ HDMI (High-Definition Multimedia Interface) จะเป็นคำตอบสำหรับการใช้งานคอมพิวเตอร์ในยุคใหม่ และ Raspberry Pi จะต้องมีความสามารถนี้เป็นหลัก

Upton ต้องรออีกถึง 5 ปี จนถึงเดือนสิงหาคม ปี ค.ศ. 2011 บอร์ดเวอร์ชัน Alpha จำนวน 50 ชุดได้ถูกผลิตออกมาเพื่อใช้ในการทดสอบและนำเสนอต่อผู้สนใจ จนถึงเดือนธันวาคมในปีเดียวกัน Raspberry Pi Model B ในรุ่น Beta จำนวน 25 บอร์ดได้รับการทดสอบอีกครั้ง โดยคราวนี้รันด้วยระบบปฏิบัติการ Linux เล่นไฟล์วิดีโอความละเอียดในระดับ HD ด้วยความละเอียด 1080p ทำเอาโลกทั้งใบแทบจะหยุดหมุน สปอตไลต์ของวงการ Embedded System และคอมพิวเตอร์ขนาดเล็กหันไปส่องและโฟกัสไปที่ Raspberr Pi แทบจะในทันที และเฝ้ารอวันเปิดตัวอย่างเป็นทางการ

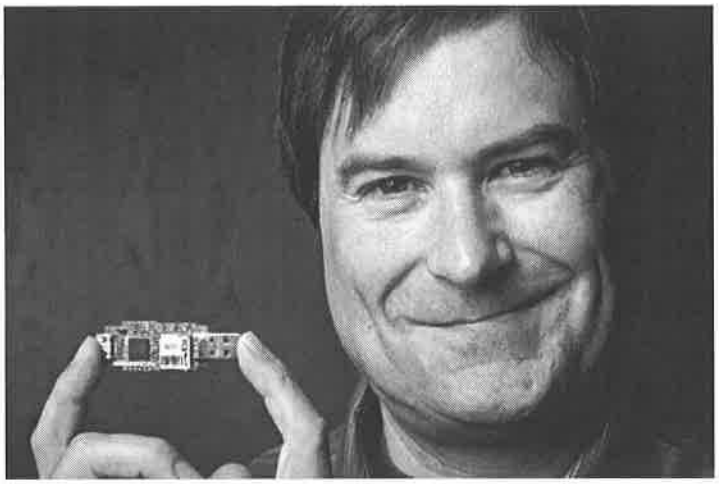
ในสัปดาห์แรกของปีค.ศ. 2012 บอร์ด Raspberry Pi จำนวน 10 บอร์ดถูกส่งไปประมูลในเว็บไซต์ E-Bay หนึ่งในนั้นได้รับการประมูลจากพิพิธภัณฑ์ The Centre for Computing History ใน Suffolk สหราชอาณาจักร บอร์ด Raspberry Pi ล็อตแรกทั้ง 10 บอร์ดสามารถสร้างมูลค่าจากการประมูลได้มากกว่า 16,000 ปอนด์ โดยบอร์ดที่มีเลขรหัสประจำตัว 01 มีมูลค่าสูงถึง 3,500 ปอนด์



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



รูปที่ 1-2 บอร์ดต้นแบบรุ่นแรกของ Raspberry Pi ที่พัฒนาขึ้นในปี ค.ศ. 2006 โดยใช้ ATmega644 ของ Atmel ทำงานด้วยสัญญาณนาฬิกาความถี่ 22.1MHz มีหน่วยความจำ SRAM 512 กิโลไบต์ สร้างสัญญาณภาพที่ความละเอียด 320x240 จุด (ภาพจาก http://www.raspberrypi.org)



รูปที่ 1-3 Eben Upton ในวัยหนุ่มกับต้นแบบของ Raspberry Pi ที่เริ่มต้นใช้ชิป ARM ขนาดเท่ากับ USB แฟลชไดรฟ มีจุดต่อพอร์ต USB และ HDMI อย่างละ 1 ช่อง (ภาพจาก http:/www.ixwebhosting.mobi/2011/12/29/12379.html)

ต่อมาโครงการ Raspberry Pi นี้ได้รับการขยายผลโดย BBC (British Broadcasting Corporation - www.bbc.com) อันเป็นเครือข่ายวิทยุและโทรทัศน์ที่ทรงอิทธิพลของสหราชอาณาจักรและของโลก ทันทีที่ทั่วโลกได้รับข่าวสารของ Raspberry Pi ยอดการสั่งซื้อก็หลั่งไหลมาอย่างมากมาย บริษัทที่ดูแลการผลิตและจำหน่ายทั้ง RS และ Element 14 ต้องวางแผนการผลิตใหม่ เพื่อรองรับกับกระแสความนิยมในตัว Raspberry Pi

Raspberry Pi ขายได้ 10,000 ชุดภายในชั่วโมงแรกที่มีการเปิดขายอย่างเป็นทางการ จากนั้นสินค้าก็ขาดตลาด ต้องมีการสั่งจองล่วงหน้า บางรายต้องรอนานกว่า 12 สัปดาห์ เมื่อ Raspberry Pi กลับเข้ามามีจำหน่ายอีกครั้งด้วยจำนวนกว่า 100,000 บอร์ด ผู้เกี่ยวข้องต้องพบกับความประหลาดใจอีกครั้งเมื่อมันถูกจำหน่ายหมดภายในวันแรกของการเปิดจำหน่าย จนถึงต้นปี ค.ศ. 2013 ยอดจำหน่ายของ Raspberry Pi ก็ผ่านหลัก 1 ล้านชิ้น นอกจากนั้น Google ยักษ์ใหญ่ของอุตสาหกรรมไอทีอีกรายก็ได้บริจาค Raspberry Pi กว่า 15,000 บอร์ดแก่โรงเรียนทั่วสหราชอาณาจักร เพื่อส่งเสริมให้เด็กๆ ได้เรียนรู้การเขียนโปรแกรมคอมพิวเตอร์ในแบบง่ายๆ อันเป็นการสร้างฐานในการพัฒนาบุคลากรด้านไอทีสำหรับอนาคต

รูปที่ 1-4 Eben Upton ในฐานะผู้ก่อตั้งมูลนิธิ Raspberry Pi มอบสิทธิ์ในการผลิตและจำหน่ายบอร์ด Raspberry Pi แก่ Element14 และ RS Components (ภาพจาก http://www.cambridge-news.co.uk)

1.2 Raspberry Pi ทำอะไรได้บ้าง

ทุกอย่างที่คอมพิวเตอร์ใหญ่ๆ ทำได้ Raspberry Pi ก็ทำได้ ! *แถมยังมีความยืดหยุ่นมากกว่าเมื่อนำไปใช้ในโครงงานระบบสมองกลฝังตัว* เพราะมีขนาดเล็กกว่ามาก มีพอร์ต GPIO สำหรับเชื่อมต่อกับอุปกรณ์อิเล็กทรอนิกส์ได้สะดวก ทำให้การส่งสัญญาณเพื่อควบคุมอุปกรณ์ภายนอกอื่นๆ ทำได้อย่างมีประสิทธิภาพและเป็นไปได้สำหรับคนในทุกระดับ เพียงมีความรู้ด้านการเขียนโปรแกรม

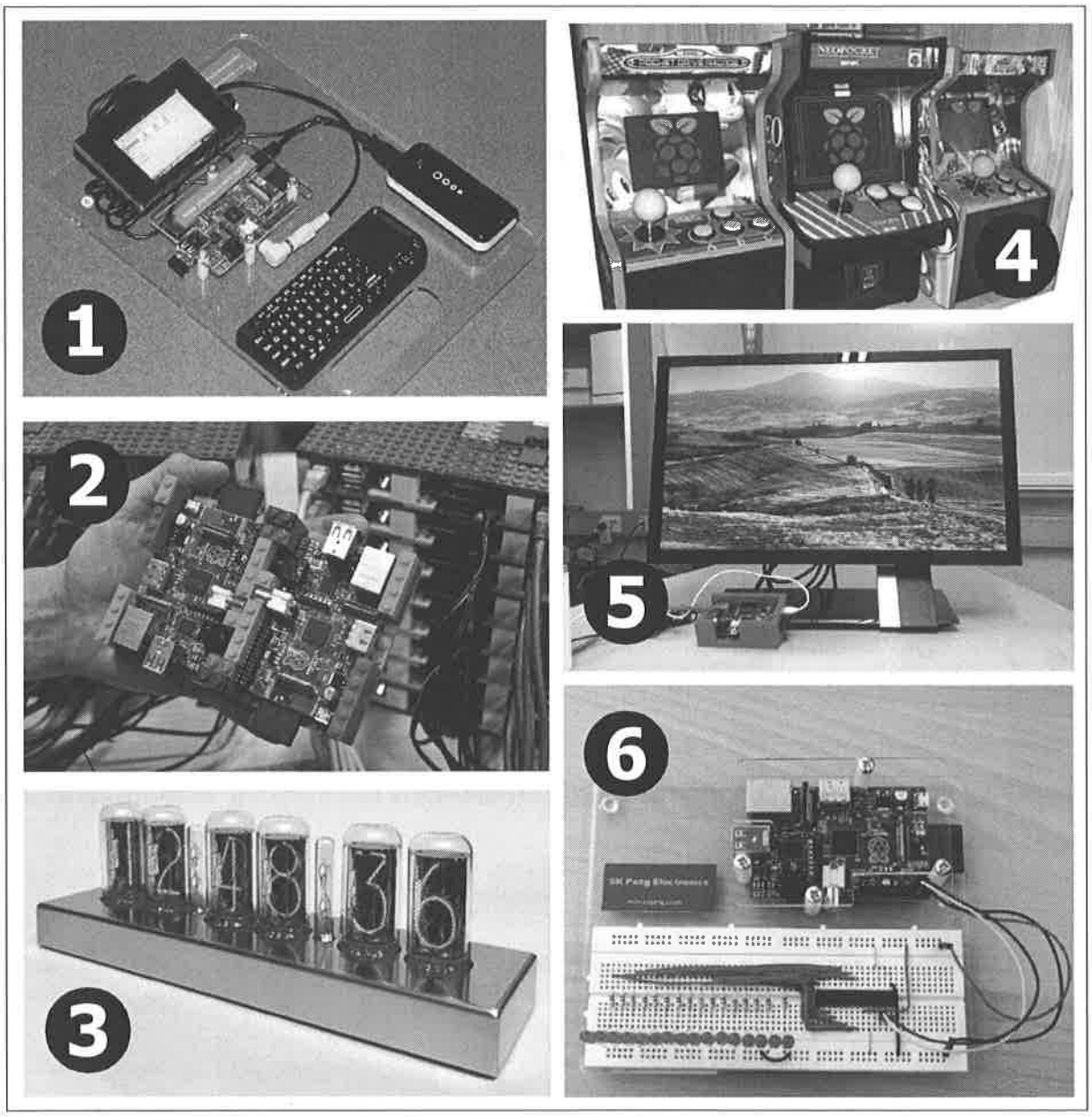
มีโครงงานจำนวนมากในเครือข่ายอินเทอร์เน็ตให้ดูเป็นตัวอย่าง ที่โดดเด่นมากคือ การนำไปใช้งานด้านความบันเทิง ทำเป็นเครื่องเล่นสื่อผสมหรือ Media player ราคาประหยัด โดยต่อ Raspberry Pi เข้ากับฮาร์ดดิสก์ที่บรรจุไฟล์เพลงหรือภาพยนต์ แล้วทำการเรียกออกมาเพื่อเล่นกลับภายใต้คุณภาพในระดับ HD เมื่อใช้กับจอภาพแบบ HDMI หรือการนำไปสร้างเป็นสมาร์ตทีวีหรือโทรทัศน์อัจฉริยะที่รับชมรายการผ่านเครือข่ายอินเทอร์เน็ต สามารถท่องเว็บไซต์ เล่นเกมออนไลน์ โดยทั้งหมดนี้ไม่ต้องใช้คอมพิวเตอร์ขนาดใหญ่อีกต่อไป เพียงมี Raspberry Pi, แหล่งจ่ายไฟที่มีเสถียรภาพสูง และจอ HDMI ร่วมกับเครือข่ายอินเทอร์เน็ตความเร็วสูง ทุกอย่างทั้งความบันเทิงและสื่อความรู้ก็จะอยู่ในกำมือของผู้ใช้งาน

นอกจากนั้นยังนำ Raspberry Pi ไปเชื่อมต่อกับตัวตรวจจับและระบบสื่อสารไร้สาย เพื่อสร้างระบบดาต้าล็อกเกอร์หรือระบบมอนิเตอร์ที่ต้องการงานประมวลผลข้อมูลที่สูงทั้งจำนวนและความเร็ว นำไปสร้างหุ่นยนต์ที่มีขีดความสามารถสูงทั้งเพื่อการแข่งขันและใช้ในอุตสาหกรรม ทั้งหมดนี้ทำได้ไม่ยากและเป็นไปได้ด้วยบอร์ด Raspberry Pi

1.2 รุ่นของบอร์ด Raspberry Pi

มูลนิธิ Raspberry Pi ได้พัฒนาและผลิตบอร์ด Raspberry Pi (นับถึงเมษายน ค.ศ. 2015) ออกมาจำหน่าย รวมแล้ว 6 รุ่นตามลำดับเวลาดังนี้

1. Raspberry Pi Model B
2. Raspberry Pi Model A
3. Raspberry Pi Model B+
4. Raspberry Pi Model A+
5. Raspberry Pi Compute module
6. Raspberry Pi 2



รูปที่ 1-5 ตัวอย่างการนำไป Raspberry Pi ไปใช้ในการสร้างสิ่งประดิษฐ์

(1) คอมพิวเตอร์เด็กแนวในสไตล์ R-Pi

(2) สร้างซูเปอร์คอมพิวเตอร์ โดยใช้ตัวต่อ Lego มาสร้างเป็นกล่องบรรจุ

(3) นาฬิกาย้อนยุคจากหลอด Nixxie ควบคุมด้วย Raspberry

(4) ตู้เกมแบบ Retro

(5) Rasperry Pi media center ดูหนัง ฟังเพลง ท่องเน็ต โดยไม่ต้องใช้คอมพิวเตอร์พีซี

(6) การทดลองใช้งาน GPIO เพื่อเชื่อมต่อวงจรอิเล็กทรอนิกส์ อีกหนึ่งความสามารถที่หาได้ไม่ง่ายจากบอร์ด Embedded Linux รุ่นอื่น ๆ แต่ Raspberry Pi จัดให้ !

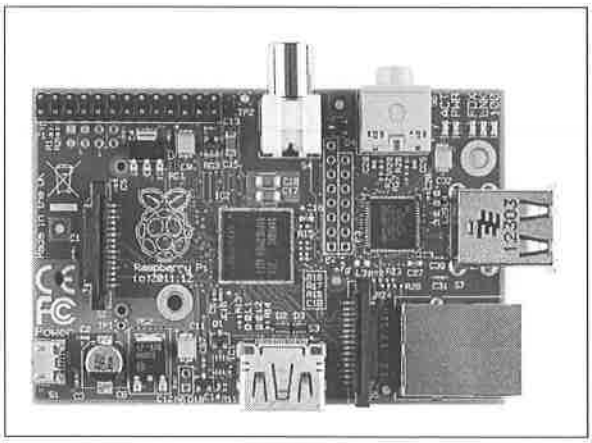
1.2.1 Raspberry Pi Model B

เป็นบอร์ด Raspberry Pi ที่ออกวางจำหน่ายในเชิงพาณิชย์รุ่นแรก โดยใช้ซีพียู BCM2835 ซึ่งเป็นชิป ARM11 คอร์ ARM1176JZF-S ความเร็ว 700MHz ที่รวมหน่วยประมวลกราฟิกหรือ GPU ไว้ภายในตัวถังเดียวกัน มีหน่วยความจำแรม 256MB ต่อมาเพิ่มเป็น 512MB

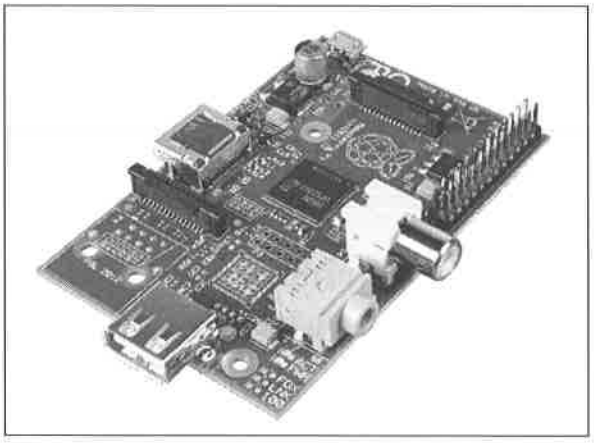
ด้านพอร์ตเชื่อมต่ออุปกรณ์ภายนอก ประกอบด้วย พอร์ต USB 2.0 (2 พอร์ต), แจ๊ก RCA และ HDMI เอาต์พุตสัญญาณวิดีโอสำหรับต่อกับโทรทัศน์หรือจอแสดงผลที่มีจุดต่อแบบ RCA ตัวเมียหรือ HDMI, จุดต่อเอาต์พุตเสียงเป็นแจ๊กหูฟัง 3.5 มม., จุดต่ออีเธอร์เน็ตหรือจุดต่อระบบ LAN, คอนเน็กเตอร์หรือจุดต่อพอร์ตอินพุตเอาต์พุต (General Purpose Input/Output : GPIO) ที่มีขาต่อบัส SPI (Serial Peripheral Interface Bus), I?C, I?S, ขาสัญญาณรับส่งข้อมูลอนุกรมหรือ UART และซ็อกเก็ตของ SD การ์ดสำหรับเสียบ SD การ์ดที่ติดตั้งระบบปฏิบัติการเรียบร้อยแล้ว

1.2.2 Raspberry Pi Model A

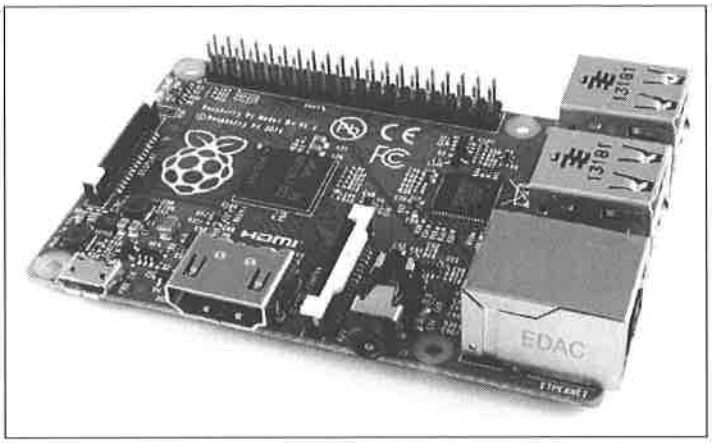
เป็นบอร์ด Raspberry Pi ที่ออกตามหลังมา โดยมีการตัดอุปกรณ์ที่ไม่ได้ใช้งานออกไป เพื่อทำให้ต้นทุนลดลง สำหรับนำไปผลิตเป็นสินค้าเพื่อให้แข่งขันได้ โดยยังคงใช้ซีพียู BCM2835 หน่วยความจำแรม 512MB มีพอร์ต USB 2.0 เหลือ 1 ช่อง, แจ๊ก RCA และ HDMI, จุดต่อเอาต์พุตเสียงเป็นแจ๊กหูฟัง 3.5 มม., จุดต่อพอร์ตอินพุตเอาต์พุต GPIO และซ็อกเก็ตของ SD การ์ด โดยตัดอุปกรณ์ในส่วนของการเชื่อมต่อ LAN ออกไป



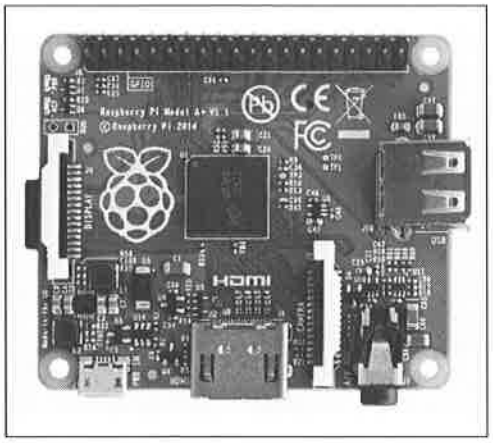
รูปที่ 1-6 บอร์ด Raspberry Pi Model B ที่ออกวางจำหน่ายเป็นรุ่นแรก



รูปที่ 1-7 บอร์ด Raspberry Pi Model A ที่ใช้แผ่นวงจรพิมพ์แบบเดียวกับ Model B แต่ไม่ติดตั้งหรือตัดอุปกรณ์บางตัวออกไป



รูปที่ 1-8 บอร์ด Raspberry Pi Model B+ ที่มีการปรับอย่างมากด้านรูปลักษณ์และจุดต่ออุปกรณ์ภายนอก



รูปที่ 1-9 บอร์ด Raspberry Pi Model A+ ที่ออกแบบแผ่นวงจรพิมพ์ใหม่ทั้งหมดเพื่อให้มีขนาดเล็กลง

1.2.3 Raspberry Pi Model B+

หลังจากได้รับคำแนะนำอย่างมากมาย ทางมูลนิธิ Raspberry Pi จึงตัดสินใจปรับรูปแบบของบอร์ด Raspberry Pi ใหม่ โดยยังคงใช้ซีพียู BCM2835 หน่วยความจำแรม 512MB มีจุดต่ออีเธอร์เน็ตและพอร์ต HDMI

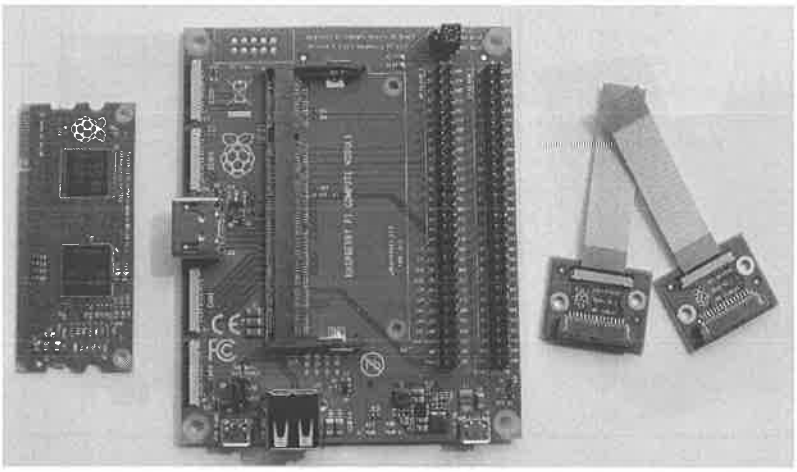
จุดที่ปรับคือ เพิ่มพอร์ต USB 2.0 เป็น 4 พอร์ต แล้วรวมแจ็ก RCA แจ็กหูฟังเข้าไว้ด้วยกันกลายเป็นแจ็ก AV ด้านคอนเน็กเตอร์หรือจุดต่อพอร์ตอินพุตเอาต์พุต GPIO เพิ่มเป็น 40 ขา และเปลี่ยนซ็อกเก็ตของ SD การ์ดเป็นแบบ micro SD การ์ด เพื่อรองรับการ์ดที่มีความจุสูงได้

1.2.4 Raspberry Pi Model A+

รุ่น A+ ได้รับการออกแบบแผงวงจรใหม่ทั้งหมด โดยไม่ใช้วิธีเดียวกับรุ่น A ทำให้บอร์ดมีขนาดเล็กลง นำไปติดตั้งในผลิตภัณฑ์ได้ง่ายขึ้น และต้นทุนรวมลดลงอย่างมีนัยสำคัญ โดยส่วนที่ตัดออกคือ จุดต่ออีเธอร์เน็ต และลดจำนวนพอร์ต USB ลงเหลือ 1 พอร์ตเหมือนกับรุ่น A

1.2.5 Raspberry Pi compute module

สำหรับบอร์ด Raspberry Pi รุ่นนี้ได้รับการพัฒนาขึ้นเพื่อนำไปสร้างระบบคอมพิวเตอร์ในแบบเฉพาะตัว (custom system) โดยตัดส่วนของซีพียูและอุปกรณ์สำคัญออกมาเป็นโมดูลคล้ายโมดูลหน่วยความจำ ดังรูปที่ 1-10 (ซ้ายสุด) ในโมดูลหลักประกอบด้วย โปรเซสเซอร์ BCM2835, แรม 512MB และหน่วยความจำแฟลช 4GB บรรจุลงในแผ่นวงจรพิมพ์ที่มีขนาดเพียง 67.6 x 30 มม. ตามขนาดมาตรฐานของโมดูล DDR2 SODIMM



รูปที่ 1-10 หน้าตาของ Rasberry Pi Compute Module Development kit (ภาพจาก http://www.cnx-software.com)

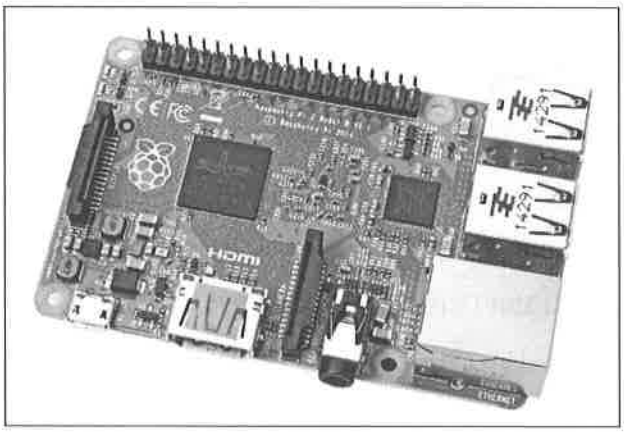
เพื่ออำนวยความสะดวกในการใช้งานโมดูล Raspberry Pi แบบนี้ บอร์ด Compute Module IO จึงเกิดขึ้น (รูปที่ 1-10 ลำดับที่ 2 จากซ้าย) สำหรับนำโมดูล Raspberry Pi compute ไปติดตั้ง โดยบอร์ด IO นี้จะมีจุดต่อไฟเลี้ยง, จุดต่อพอร์ต USB ทั้งแบบ A และ micro USB สำหรับบูตระบบจากโฮสต์ที่ต่อเข้ามาร่วมใช้งานด้วย, จุดต่อพอร์ต GPIO และจุดต่อ HDMI ส่วนการติดต่อกับโมดูลกล้องและจอแสดงผลผ่านพอร์ต CSI และ DSI ก็จะมีบอร์ดอะแดปเตอร์มาช่วยดังในรูปที่ 1-10 (ขวามือ)

1.2.6 Raspberry Pi 2

หลังจากบอร์ด Raspberry Pi ออกวางจำหน่าย ก็มีผู้ผลิตอื่นๆ หลายรายผลิตบอร์ดคอมพิวเตอร์ที่ทำงานในลักษณะเดียวกันออกมาจำนวนมาก ทางมูลนิธิ Raspberry Pi จึงได้ผลิตบอร์ดรุ่นใหม่ออกมา โดยเปลี่ยนซีพียูเป็น BCM2836 ซึ่งเป็นชิป ARM Cortex-A7 แบบ 4 แกน และเพิ่มหน่วยความจำเป็น 1GB กำหนดชื่อเป็น **Raspberry Pi 2** มีความเร็วในการทำงานสูงขึ้นถึง 6 เท่า

การเปลี่ยนแปลงส่งผลต่อตลาดในทันที บอร์ด Raspberry Pi 2 ได้รับการตอบรับดีมาก Microsoft ประกาศพัฒนาระบบปฏิบัติการวินโดวส์ 10 ในรุ่นพิเศษ (Windows 10 IoT) ที่รองรับ Raspberry Pi 2 (เปิดให้ดาวน์โหลดมาใช้งานได้ฟรีตั้งแต่ 29 กรกฎาคม พ.ศ. 2558) จึงทำให้บอร์ด Raspberry Pi 2 กลายเป็นบอร์ดที่ผู้สนใจงานด้านนี้ต้องมี

รูปที่ 1-11 บอร์ด Raspberry Pi 2 ที่มีการปรับหน้าตาเล็กน้อยเนื่องจากมีการเปลี่ยนซีพียูและเพิ่มความจุของหน่วยความจำเป็น 1GB



1.3 คุณสมบัติทางเทคนิคของ Raspberry Pi 2

สรุปได้ดังนี้

● ชิปประมวลผลหลัก : Broadcom BCM2836 ARM Cortex-A7 quad core ความเร็ว 900MHz

● หน่วยประมวลกราฟิกหรือ GPU : Broadcom VideoCore IV dual-core GPU หรือเทียบเท่า รองรับการแสดงผลผ่านจอ HDMI

● หน่วยความจำ SDRAM : 1GB LPDDR2

● จุดต่อ :

USB 2.0 จำนวน 4 พอร์ต รองรับการขยายผ่าน USB ฮับ ใช้เชื่อมต่อกับอุปกรณ์ USB ได้หลายแบบ อาทิ เมาส์, คีย์บอร์ด, WiFi ดองเกิล, บลูทูธดองเกิล, ฮาร์ดดิสก์ภายนอก, USB แฟลชไดรฟ เป็นต้น

แจ็ก AV ที่รวมเอาต์พุตเสียงและสัญญาณภาพหรือวิดีโอไว้ด้วยกันเป็นแจ็กหูฟัง 3.5 มม.

จุดต่ออีเธอร์เน็ตหรือจุดต่อระบบ LAN

คอนเน็กเตอร์หรือจุดต่อพอร์ตอินพุตเอาต์พุต 40 ขา (General Purpose Input/Output : GPIO) ที่มีขาต่อบัส SPI (Serial Peripheral Interface Bus), I?C, I?S, ขาสัญญาณรับส่งข้อมูลอนุกรม หรือ UART

*ซ็อกเก็ตของ micro SD การ์ด*สำหรับเสียบ micro SD การ์ดที่ติดตั้งระบบปฏิบัติการแล้ว

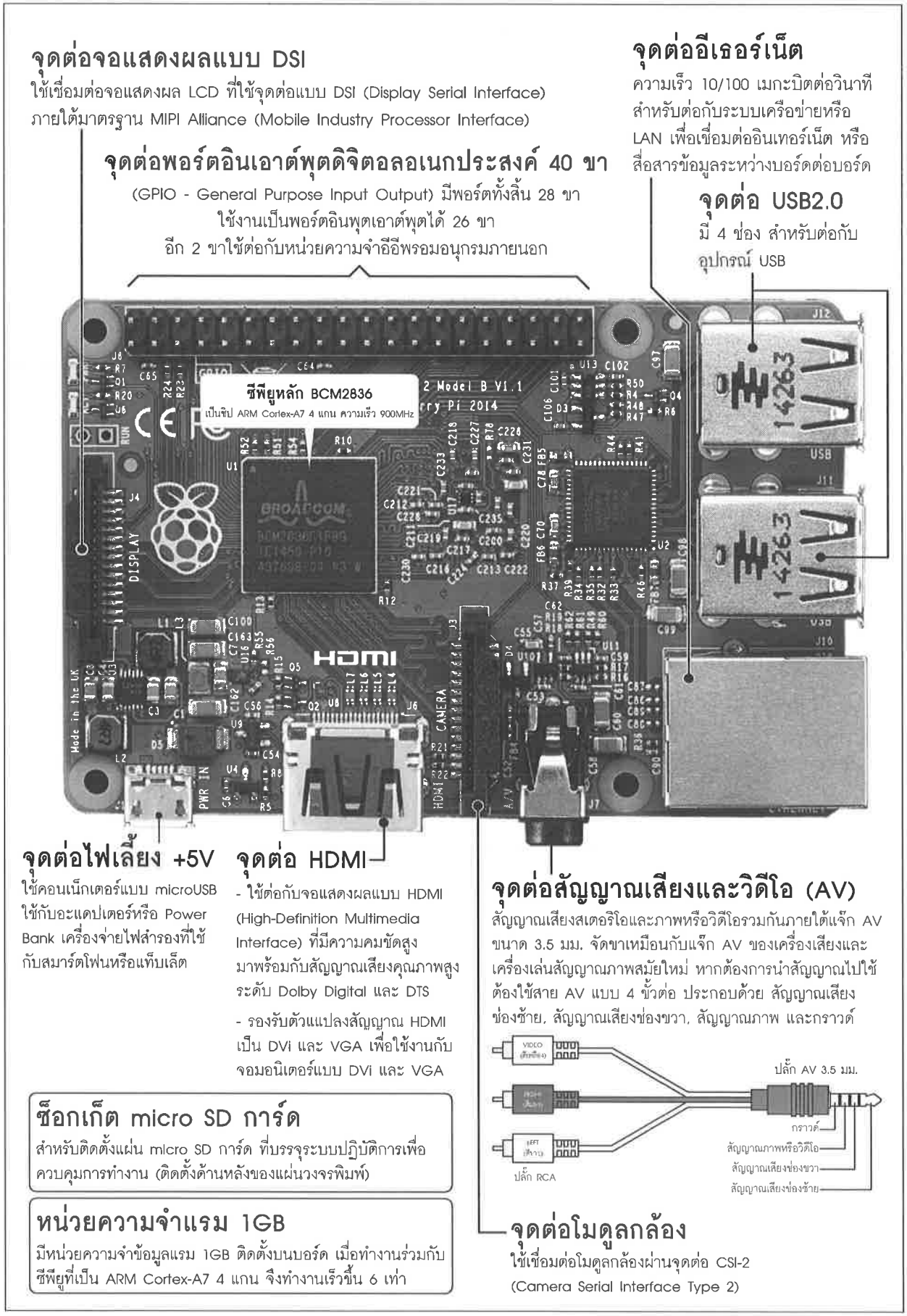
จุดต่อ CSI สำหรับต่อโมดูลกล้อง Raspberry Pi

จุดต่อ DSI สำหรับจอแสดงผล LCD

จุดต่อไฟเลี้ยง +5V ผ่านคอนเน็กเตอร์ microUSB

● ความต้องการไฟเลี้ยง : +5V 1.5A เป็นอย่างน้อย แนะนำใช้งานกับอะแดปเตอร์ไฟตรง +5V 2A หรือเครื่องจ่ายไฟสำรองหรือเพาเวอร์แบงก์ +5V ขนาดตั้งแต่ 3,000mA ขึ้นไป

● ขนาด : 85 x 56 มม.



รูปที่ 1-12 ส่วนประกอบที่ควรทราบของบอร์ด Raspberry Pi 2

1.4 ส่วนประกอบต่างๆ ของบอร์ด Raspberry Pi 2

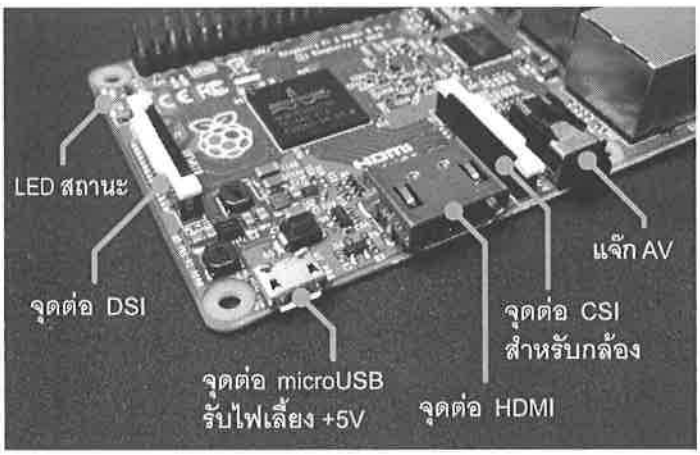
ในรูปที่ 1-12 แสดงภาพรวมส่วนประกอบต่างๆ ที่สำคัญของบอร์ด Raspberry Pi 2

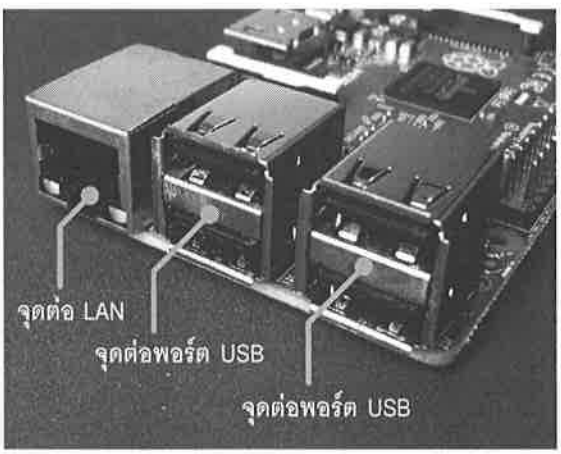
1.4.1 ส่วนประมวลผล

บอร์ด Raspberry Pi 2 ใช้ชิปประมวลผลจาก Broadcom เบอร์ BCM2836 เป็นซีพียู 4 แกนที่ ใช้สถาปัตยกรรม ARM Cortex-A7 ทำงานด้วยความถี่ 900MHz และมีชิปประมวลผลกราฟิกเป็น VideoCore IV แบบ 2 แกนที่รองรับ OpenGL ES 2.0, OpenVG และรองรับ Decode H.264 ระดับ 1080p30 มีหน่วยความจำข้อมูลแรมแบบ LPDDR2 ความจุ 1GB

1.4.2 จุดต่ออุปกรณ์ภายนอก

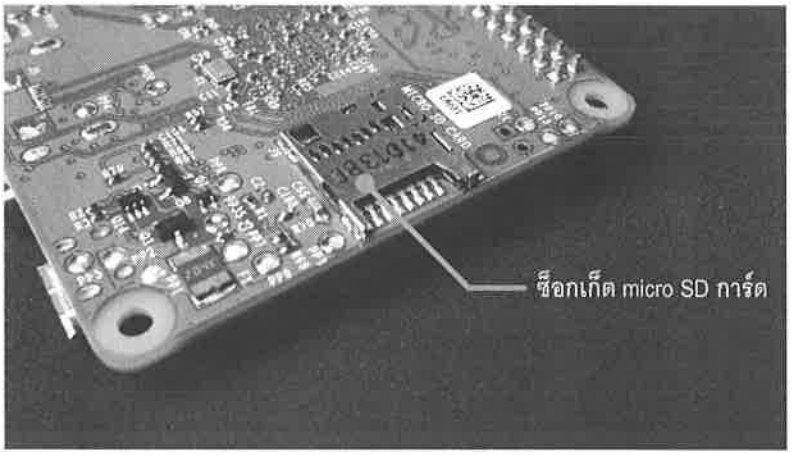
มีจุดต่อ HDMI สำหรับต่อกับจอแสดงผลความละเอียดสูง รองรับ HDMI 1.3 และ 1.4, มีแจ๊ก AV 3.5 มม. สำหรับเอาต์พุตเสียงสเตอริโอและสัญญาณภาพแบบ Component Video, มีจุดต่อแหล่งจ่ายไฟ +5V ผ่านทางคอนเน็กเตอร์ microUSB, รองรับการเชื่อมต่อโมดูลกล้องผ่านคอนเน็กเตอร์ CSI (Camera Serial Interface), เชื่อมต่อกับจอแสดงผลผ่านคอนเน็กเตอร์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





1.4.3 ส่วนติดต่อการ์ดหน่วยความจำ

หน่วยความจำข้อมูลและระบบปฏิบัติการของบอร์ด Raspberry Pi 2 จะถูกเก็บไว้อยู่ในแผ่นหน่วยความจำ micro SD การ์ด โดยมีช่องเสียบ micro SD การ์ดอยู่ที่ใต้บอร์ด



1.4.4 จุดต่อพอร์ต GPIO

ที่ขาดไม่ได้เลยสำหรับบอร์ด Raspberry Pi 2 นั่นคือ พอร์ตอินพุตเอาต์พุตดิจิตอลอเนกประสงค์ หรือ GPIO 40 ขาเพื่อนำไปต่อเข้ากับวงจรอิเล็กทรอนิกส์ต่างๆ มีขาพอร์ตที่รองรับระดับสัญญาณลอจิก 3.3V ให้ใช้งานรวม 26 ขา รองรับ UART, I^2C และ SPI แต่ไม่รองรับอินพุตเอาต์พุตอะนาลอกโดยตรง

ทั้งหมดนี้คือ ปฐมบทของการแนะนำให้รู้จักกับบอร์ดไมโครคอมพิวเตอร์แผ่นเดี่ยวที่โด่งดังที่สุด พัฒนาโปรแกรมได้หลากหลายบนระบบปฏิบัติการ Linux บอร์ดไมโครฯ ที่ใช้ชื่อผลไม้ที่อุดมไปด้วยสารอาหารที่เป็นประโยชน์ต่อร่างกาย และนี่คือบอร์ดสำหรับนักพัฒนา Embedded Computer ยุคใหม่ที่มาพร้อมกับความโดดเด่นทั้งคุณสมบัติและราคาที่ใครๆ ก็เข้าถึงได้

บทที่ 2
เตรียมการใช้งาน Raspberry Pi 2

เนื่องจากบอร์ด Raspberry Pi 2 เป็นบอร์ดคอมพิวเตอร์ขนาดเล็กแบบหนึ่ง จึงต้องมีการติดตั้งหน่วยความจำหลักของระบบ ติดตั้งระบบปฏิบัติการ และเชื่อมต่ออุปกรณ์พื้นฐานเพื่อตั้งค่าการทำงาน

2.1 เตรียมอุปกรณ์และเครื่องมือ

เตรียมอุปกรณ์ดังนี้

1. บอร์ด Raspberry Pi 2

2. แผ่นหน่วยความจำ microSD การ์ด สำหรับเก็บระบบปฏิบัติการและข้อมูล ควรมีความจุตั้งแต่ 4GB คลาส 4 ขึ้นไป ซึ่งก็คือ SDHC การ์ด แนะนำ Class 10 จะเป็นแบบ FAT32 หรือ NTFS ก็ได้

3. สาย microUSB ใช้สำหรับต่อกับแหล่งจ่ายไฟ +5V 2A ที่มีจุดต่อแบบ USB เช่นอะแดปเตอร์ของแท็บเล็ตแอนดรอยด์หรือวินโดวส์, เครื่องจ่ายไฟสำรองหรือ เพาเวอร์แบงก์ (Power Bank) ขนาด 3,000mA ขึ้นไป ไม่ควรใช้แหล่งจ่ายไฟจากพอร์ต USB ของคอมพิวเตอร์ เนื่องจากมีความสามารถในการจ่ายกระแสไฟฟ้าไม่เพียงพอ

4. อะแดปเตอร์ไฟตรง +5V 2A ที่มีจุดต่อแบบ USB หรือเครื่องจ่ายไฟสำรองหรือ เพาเวอร์แบงก์ (Power Bank) ขนาด 3,000mA ขึ้นไป หรืออะแดปเตอร์ +5V 2A ที่มีหัวต่อแบบ microUSB หากใช้แบบนี้ สาย microUSB ในข้อ 3. ไม่ต้องใช้

5. สาย HDMI สำหรับต่อกับจอแสดงผล สำหรับผู้ใช้งานที่มีจอแสดงผลแบบ DVi จะต้องใช้ตัวแปลงหรือสายแปลงสัญญาณ HDMI เป็น DVi มาต่อแทน

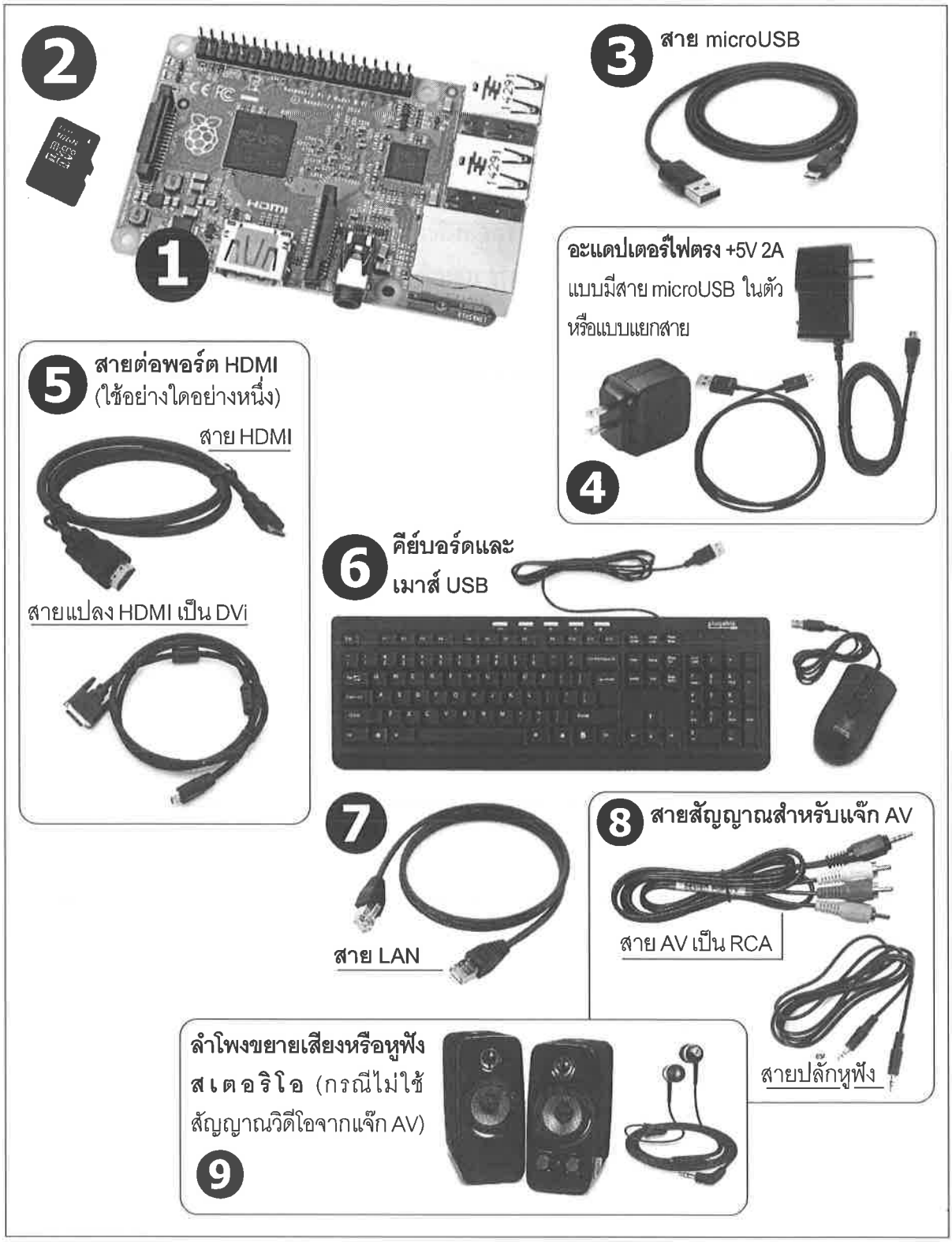
6. คีย์บอร์ดและเมาส์แบบ USB สำหรับควบคุมการทำงานของบอร์ด Raspberry Pi 2

7. สาย LAN (มีหรือไม่มีก็ได้) สำหรับเชื่อมต่อกับอินเทอร์เน็ต

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

9. ลำโพงขยายเสียงที่มีจุดต่อสัญญาณอินพุตเป็นแจ๊ก RCA หรือแจ๊กหูฟังสเตอริโอ 3.5 มม. เพื่อต่อกับแจ๊ก AV ของบอร์ด Raspberry Pi 2 (มีหรือไม่มีก็ได้)

รูปที่ 2-1 แสดงอุปกรณ์ทั้งหมดที่ควรมีสำหรับใช้งานบอร์ด Raspberry Pi 2 และ Model B ทุกรุ่น



รูปที่ 2-1 อุปกรณ์ที่ต้องเตรียมสำหรับการเริ่มต้นใช้งานบอร์ดคอมพิวเตอร์ Raspberry Pi ทุกรุ่น (ในที่นี้เลือกใช้กับ Raspberry Pi 2)

2.2 ติดตั้งระบบปฏิบัติการลงใน microSD การ์ด

(ในกรณีจัดซื้อ Raspberry Pi 2 Starter kit จาก inex ให้ข้ามหัวข้อนี้เพราะได้จัดการไว้ให้พร้อมใช้งานแล้ว)

สำหรับ Raspberry Pi 2 ใช้หน่วยความจำ microSD การ์ดในการทำงานและติดตั้งระบบปฏิบัติ มีข้อดีที่ผู้ใช้งานสามารถสลับเปลี่ยน microSD การ์ดได้ตามต้องการ โดยติดตั้งระบบปฏิบัติการต่างกันในแต่ละแผ่น

สำหรับไฟล์ติดตั้งของระบบปฏิบัติการจะเรียกกันว่า ไฟล์อิมเมจ (image file) สำหรับบอร์ด Raspberry Pi 2 มีหลายแบบให้เลือกใช้งาน ไม่ว่าจะเป็น Raspbian, Pidora หรือ Raspbmc เป็นต้น สำหรับในหนังสือเล่มนี้เลือกใช้ Raspbian ดาวน์โหลดได้ฟรีจาก***http://www.raspberrypi.org/downloads*** เลือกหัวข้อ ***RASPBIAN*** จากนั้นทำการแตกไฟล์ .zip ออกมาจะได้เป็นไฟล์อิมเมจ .img

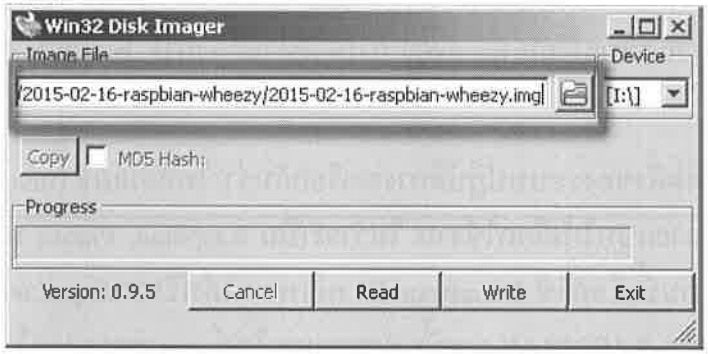
ในการติดตั้งไฟล์อิมเมจลงใน microSD การ์ดจะต้องทำผ่านคอมพิวเตอร์ด้วยการใช้ตัวอ่านเขียนการ์ดหน่วยความจำอาจเป็นแบบติดตั้งมาตัวคอมพิวเตอร์ หรือแบบต่อภายนอกผ่านพอร์ต USB ก็ได้ หากตัวอ่านเขียนการ์ดหน่วยความจำไม่สามารถใช้งานกับ microSD การ์ด จะต้องใช้ตัวแปลงขนาดเป็น SD การ์ด ซึ่งปกติจะมาพร้อมกับตัว micro SD การ์ด

สำหรับระบบปฏิบัติการวินโดวส์แนะนำให้ใช้ซอฟต์แวร์ที่ชื่อว่า **Win32 Disk Imager** โดยดาวน์โหลดได้จาก ***http://www.softpedia.com/get/CD-DVD-Tools/Data-CD-DVD-Burning/Win32-Disk-Imager.shtml*** จากนั้นทำการติดตั้งให้เรียบร้อย

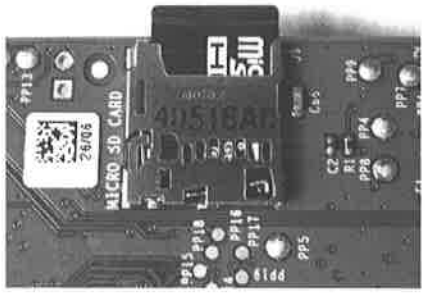
เมื่อติดตั้งซอฟต์แวร์ Win32 Disk Imager แล้ว จะเข้าสู่ขั้นตอนการติดตั้งไฟล์อิมเมจลงใน microSD การ์ด ดังนี้

(2.2.1) เสียบ microSD การ์ดเข้าที่ตัวอ่านเขียนการ์ดหน่วยความจำที่คอมพิวเตอร์ให้รันซอฟต์แวร์ Win32 Disk Imager ขึ้นมา สังเกตที่ช่อง Device ว่าเป็นไดรฟของ SD การ์ดหรือไม่ จากตัวอย่างจะเป็นไดรฟ I ต้องตรวจสอบให้แน่ใจว่าไม่ผิด เพราะถ้าเลือกผิดอาจทำให้ข้อมูลในไดรฟนั้นๆ หายไปได้ จากนั้นคลิกที่ปุ่มรูปแฟ้มที่อยู่ในช่อง Image File เพื่อเลือกไฟล์อิมเมจที่ต้องการติดตั้ง

(2.2.2) เลือกไฟล์อิมเมจของระบบปฏิบัติการ Raspbian ทำการตรวจสอบไดรฟที่จะติดตั้งไฟล์อิมเมจอีกครั้ง เพื่อป้องกันความผิดพลาด เมื่อตรวจสอบจนแน่ใจแล้ว ให้คลิกปุ่ม Write เพื่อเริ่มเขียนข้อมูลของไฟล์อิมเมจลงใน SD การ์ด ในขั้นตอนการเขียนข้อมูลจะใช้เวลาค่อนข้างนาน เพราะข้อมูลมีขนาดใหญ่ ทั้งนี้ยังขึ้นอยู่กับความเร็วในการเขียนข้อมูลของ SD การ์ดด้วย



(2.2.3) เมื่อเสร็จแล้ว ถอดแผ่น microSD การ์ดออกจากตัวอ่านเขียนการ์ดหน่วยความจำนำไปติดตั้งใช้งานกับบอร์ด Raspberry Pi 2 ต่อไป



2.3 ทดสอบการทำงานเบื้องต้น

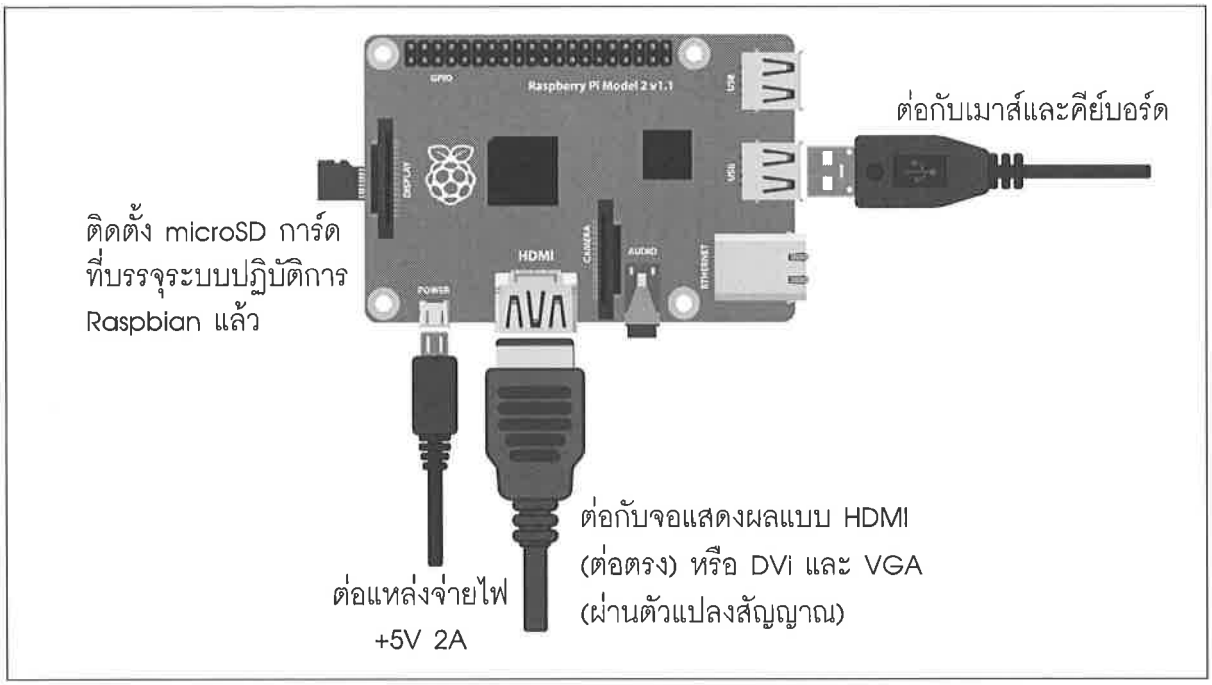
มีขั้นตอนโดยสรุปดังนี้

(2.3.1) ต่อสายคีย์บอร์ด, เมาส์, จอภาพ และแหล่งจ่ายไฟให้เรียบร้อย จากนั้นจ่ายไฟให้กับบอร์ด Raspberry Pi 2 ดังรูปที่ 2-2

(2.32) จากนั้นบอร์ด Raspberry Pi 2 จะเริ่มทำงาน แสดงรายการบูตระบบต่างๆ ดังรูปที่ 2-3 รอจนกว่าจะขึ้นหน้าล็อกอิน ป้อนข้อมูลและรหัสผ่านต่อไปนี้เพื่อทำการล็อกอิน

```
Username: pi
Password: raspberry
```

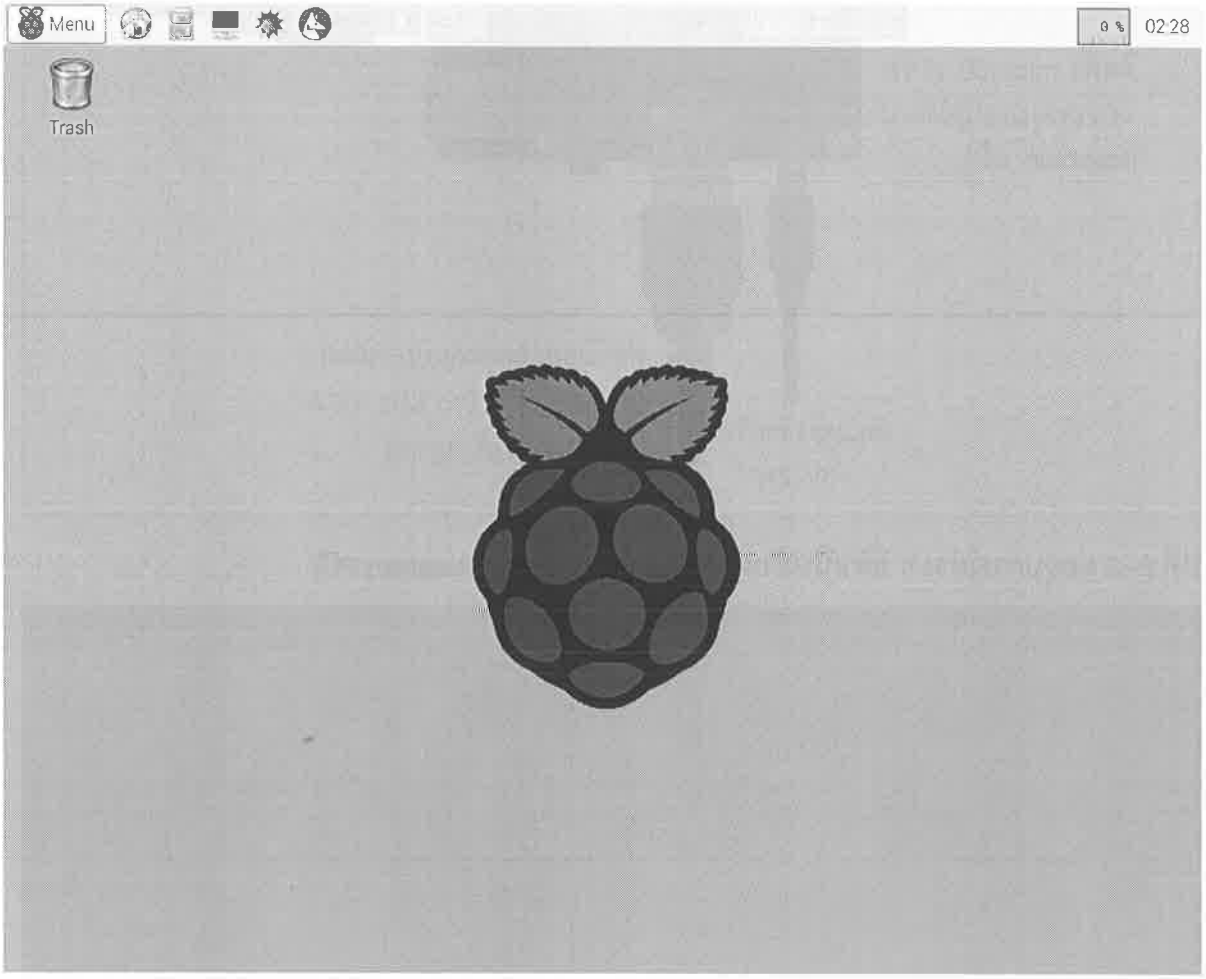
หลังจากล็อกอินเรียบร้อยแล้ว ใช้งานบอร์ด Raspberry Pi 2 ได้ทันที



รูปที่ 2-2 ต่ออุปกรณ์ต่าง ๆ ที่จำเป็นสำหรับเริ่มต้นใช้งานบอร์ด Raspberry Pi 2

รูปที่ 2-3 หน้าต่างแสดงการบูตระบบของ Raspberry Pi 2

(2.3.3) กรณีที่ต้องการใช้งานแบบกราฟิกให้พิมพ์คำสั่ง `startx` จากนั้นระบบจะเข้าสู่หน้าต่างที่มีรูปพื้นหลังเป็นราสเบอร์รี เพื่อเริ่มการใช้งานในโหมดกราฟิก ดังรูปที่ 2-4

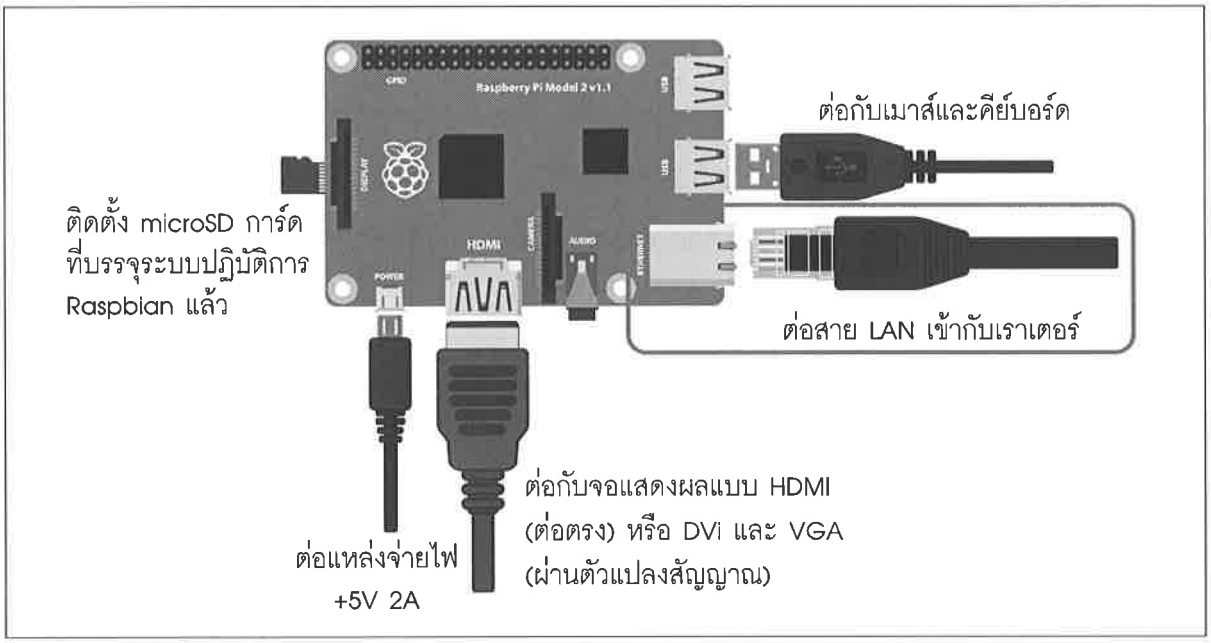


รูปที่ 2-4 หน้าต่าง Desktop ของระบบปฏิบัติการ Raspbian ในโหมดกราฟิก

2.4 การใช้งานกับระบบเครือข่ายและเชื่อมต่ออินเทอร์เน็ต

มีขั้นตอนดังต่อไปนี้

(2.4.1) เมื่อต้องการเชื่อมต่ออินเทอร์เน็ตให้ต่อสาย LAN เข้าที่จุดต่อ LAN ซึ่งเป็นคอนเน็กเตอร์ RJ-45 จะใช้งานได้ทันที โดยไม่ต้องทำการตั้งค่าใดๆ (ทั้งนี้ ผู้ใช้งานต้องมีการเปิดใช้บริการอินเทอร์เน็ตจากผู้ให้บริการด้วย) ดังรูปที่ 2-5



รูปที่ 2-5 ต่อสาย LAN เข้ากับบอร์ด Raspberry Pi 2

(2.4.2) เมื่อเชื่อมต่ออินเทอร์เน็ตแล้ว เปิดโปรแกรม **LX Terminal** ขึ้นมา แล้วใช้คำสั่ง **`sudo apt-get update`** เพื่อทำการอัปเดตโปรแกรมล่าสุด

(2.4.3) ใช้คำสั่ง **`sudo apt-get upgrade`** กดปุ่ม **Y** ตามด้วยกดปุ่ม **Enter** เพื่อยืนยัน แล้วรอดาวน์โหลดไฟล์ จากนั้นระบบจะถามว่า *ต้องการทำอย่างไรกับแพ็กเกจ* กดปุ่ม **Y** แล้ว **Enter** เพื่อติดตั้งทับไฟล์เดิม แล้วรอจนกระทั่งการอัปเดตเสร็จสิ้น

2.5 การติดตั้งโปรแกรมอื่นๆ ลงใน microSD การ์ดเพื่อใช้งานกับบอร์ด Raspberry Pi 2

(2.5.1) ในกรณีที่ต้องการติดตั้งโปรแกรมอื่นๆ เพิ่มเติม ให้ใช้คำสั่ง

sudo apt-get install (ชื่อโปรแกรม)

ผู้ใช้งานจะต้องตรวจสอบโปรแกรมหรือซอฟต์แวร์ที่นำมาติดตั้งก่อนว่า โปรแกรมนั้นๆ ทำงานบนระบบปฏิบัติการที่ติดตั้งให้กับบอร์ด Raspberry Pi 2 ได้หรือไม่

(2.5.2) ตัวอย่างการติดตั้งโปรแกรม **Chromium Browser** ให้กับบอร์ด Raspberry Pi 2 พิมพ์คำสั่ง

sudo apt-get install chromium-browser

จากนั้นกดปุ่ม **Y** ตามด้วย **Enter** เพื่อยืนยันการติดตั้ง

(2.5.3) เมื่อติดตั้งเสร็จแล้ว เมื่อต้องการเรียกใช้งาน ให้ไปที่ **Start > Internet > Chromium Web Browser**

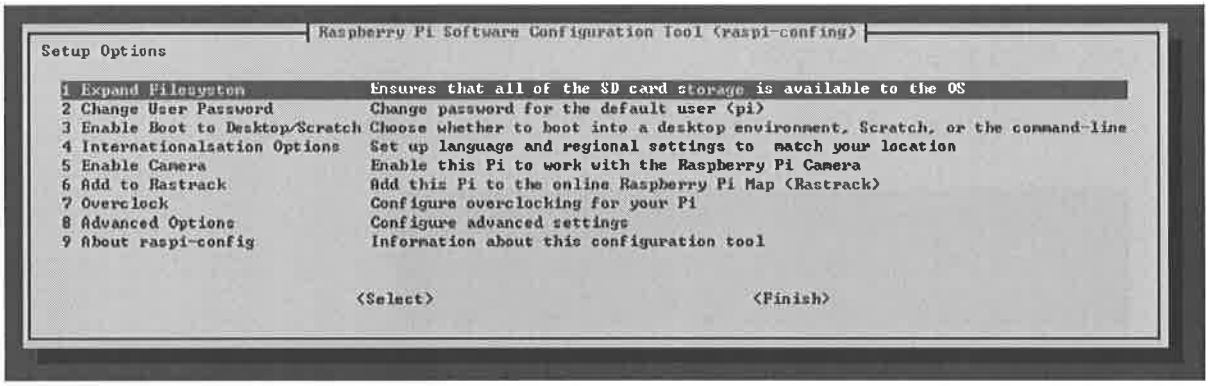
2.6 การเพิ่มพื้นที่หน่วยความจำข้อมูลบน Raspberry Pi 2 ให้มากที่สุด

โดยปกติแล้วเมื่อติดตั้งระบบปฏิบัติการณ์ใดๆลงใน microSD การ์ด ตัวระบบจะจองพื้นที่เล็กน้อยเท่านั้น โดยไม่สนใจว่าพื้นที่ของ microSD การ์ดจะมีความจุเท่าใด จึงต้องมีการตั้งค่าบน Raspberry Pi 2 ก่อน เพื่อทำให้ Raspberry Pi 2 มีพื้นที่หน่วยความจำข้อมูลสูงสุดจาก microSD การ์ดไว้ใช้งาน

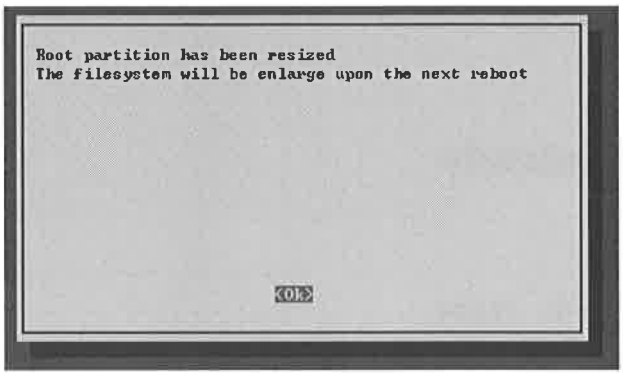
(2.6.1) ทำการตั้งค่าโดยเปิดหน้าต่าง **Terminal** เรียกคำสั่ง

```
sudo raspi-config
```

แล้วเลือกที่ **Expand Filesystem**



(2.6.2) รอจนกระทั่งมีหน้าต่างแจ้งว่า *เสร็จเรียบร้อย* ทำการรีบูตระบบใหม่ ก็จะเห็นพื้นที่ของหน่วยความจำที่ปรับใหม่แล้ว



(2.6.3) เมื่อรีบูตระบบเสร็จแล้วให้ตรวจสอบหน่วยความจำข้อมูลด้วยคำสั่ง **df -h** จะเห็นว่า *พื้นที่หน่วยความจำถูกขยายเพิ่มขึ้นตามขนาดของ SD การ์ดแล้ว*

```
pi@raspberrypi ~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          3.6G  1.8G  1.6G  53% /
/dev/root       3.6G  1.8G  1.6G  53% /
devtmpfs        211M     0  211M   0% /dev
tmpfs            44M  228K   44M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            88M     0   88M   0% /run/shm
/dev/mmcblk0p1   56M   19M   38M  34% /boot
pi@raspberrypi ~ $
```

2.7 คำสั่งเบื้องต้นใน Terminal

2.7.1 ls

แสดงรายชื่อไฟล์และโฟลเดอร์ในตำแหน่งที่อยู่ปัจจุบัน

```
pi@raspberrypi ~ $ ls
Desktop  ocr_pi.png  python_games
pi@raspberrypi ~ $
```

2.7.2 cd <directory>

ย้ายตำแหน่งปัจจุบันไปยังตำแหน่งที่กำหนดไว้ใน **<directory>**

```
pi@raspberrypi ~ $ cd /home/pi/Desktop
pi@raspberrypi ~/Desktop $
```

2.7.3 sudo apt-get install <program>

เป็นคำสั่งติดตั้งโปรแกรมจากเซิร์ฟเวอร์

2.7.4 sudo apt-get update

อัปเดตระบบจากเซิร์ฟเวอร์

2.7.5 sudo apt-get upgrade

อัปเกรดระบบจากเซิร์ฟเวอร์

2.7.6 sudo shutdown -h now

ปิดระบบ

2.7.7 sudo reboot

รีบูตระบบ

2.7.8 sudo nano <file directory>

เปิดเท็กซ์เอดิเตอร์เพื่อแก้ไขไฟล์ตามที่กำหนดไว้ใน **<file directory>**

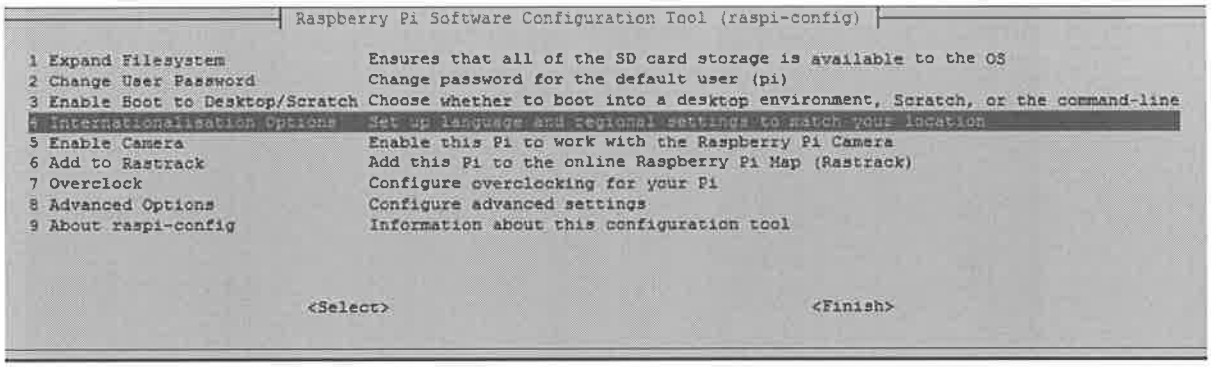
2.8 เปลี่ยนรูปแบบคีย์บอร์ด

รูปแบบเริ่มต้นของคีย์บอร์ดที่ระบบปฏิบัติการ Raspbian กำหนดไว้เป็นแบบ **English (UK)** ทำให้การพิมพ์สัญลักษณ์พิเศษบางตัวต่างจากแบบ US หากต้องการเปลี่ยนค่ารูปแบบของคีย์บอร์ดใหม่ ให้ดำเนินการดังนี้

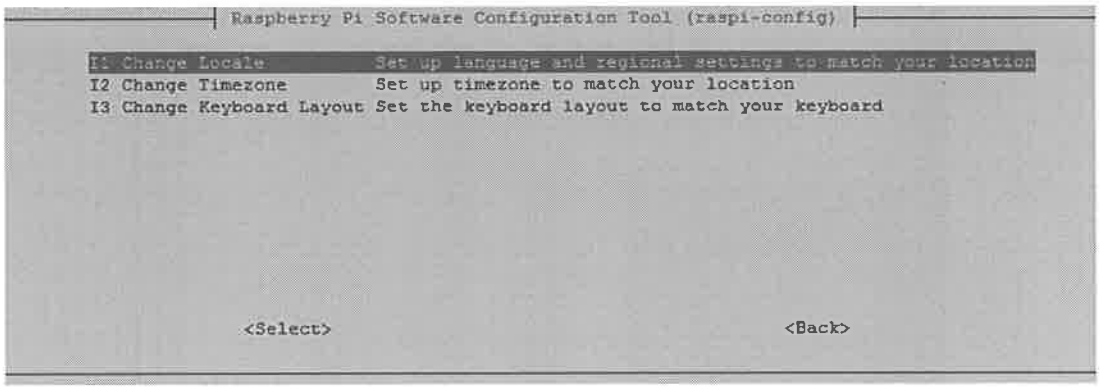
(2.8.1) พิมพ์คำสั่ง

```
sudo raspi-config
```

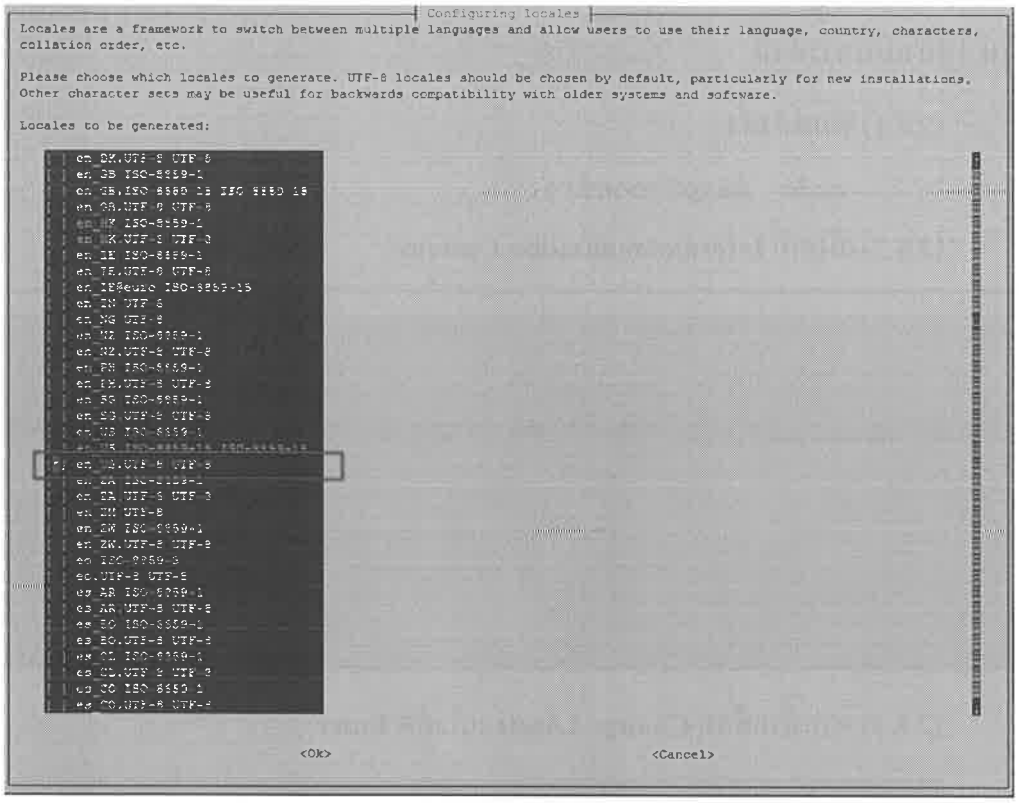
(2.8.2) เลือกที่ **Internationalisation Options**

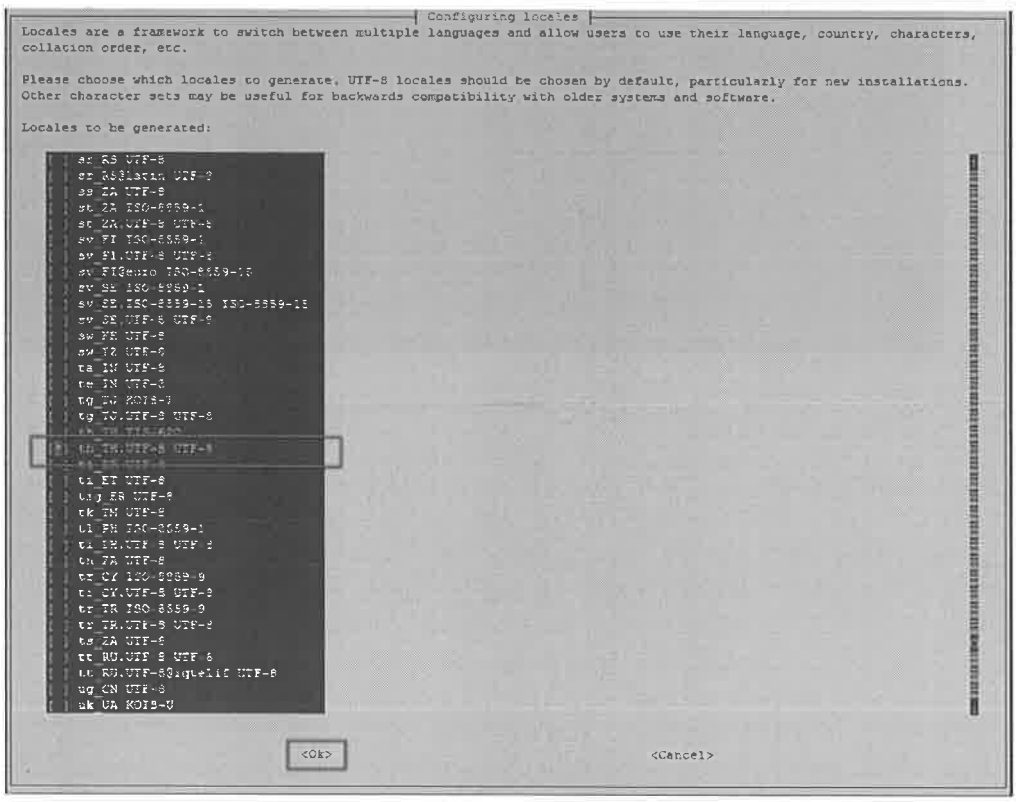


(2.8.3) จากนั้นเลือก **Change Locale** แล้วกด **Enter**



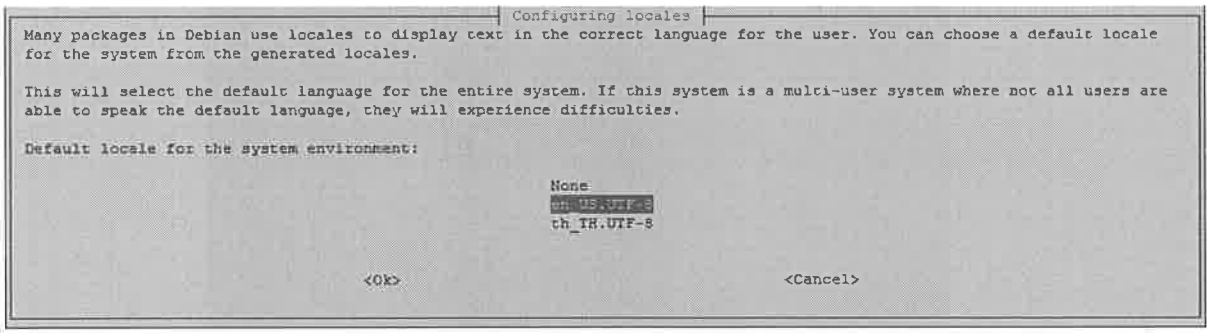
(2.8.3.1) เข้าสู่หัวข้อ **Locales to be generated:** ใช้ปุ่มลูกศรเลื่อนรายการลงมาเรื่อยๆ เลือกโดยกด **spacebar** ที่หัวข้อย่อย **en_US.UTF-8 UTF-8** และ **th_TH.UTF-8 UTF-8** แล้วกดที่ปุ่ม **OK**





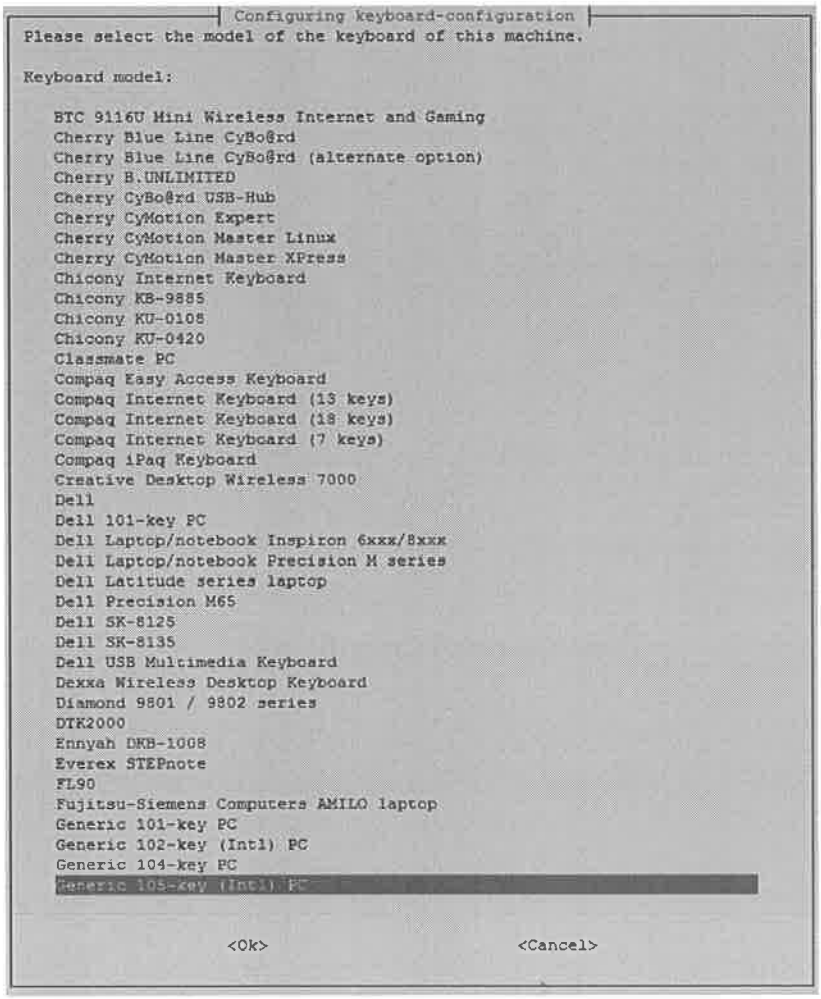
(2.8.3.2) ที่หัวข้อ **Default locale for the system environment :**

เลือก **en_US.UTF-8** เพื่อเลือกคีย์บอร์ดภาษาอังกฤษเป็นค่าเริ่มต้น

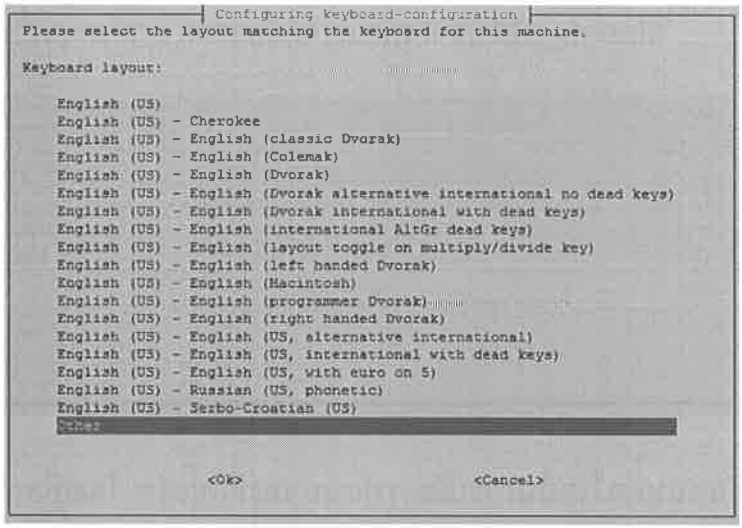


(2.8.4) ขั้นตอนต่อไปเป็นการเลือกรูปแบบของคีย์บอร์ด โดยเลือก **Change Keyboard Layout** แล้วกด **Enter**

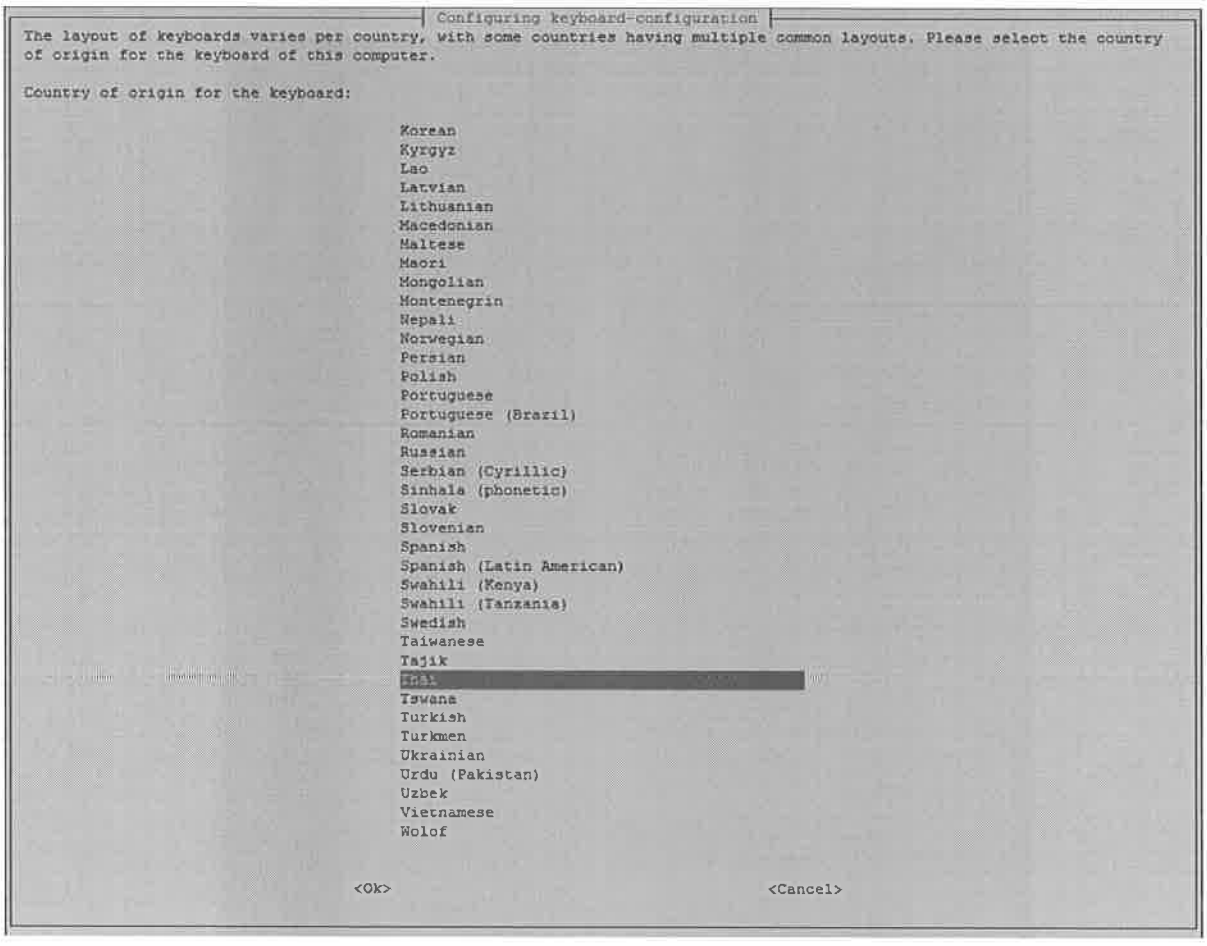
(2.8.4.1) ที่หัวข้อ **Keyboard model** : เลือก **Generic 105-key (Intl) PC** แล้วกด **Enter**



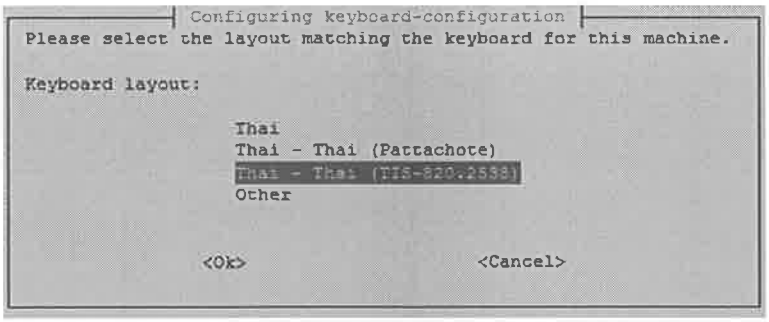
(2.8.4.2) ที่หัวข้อ **Keyboard layout :** เลือก **Other** แล้วกด **Enter**



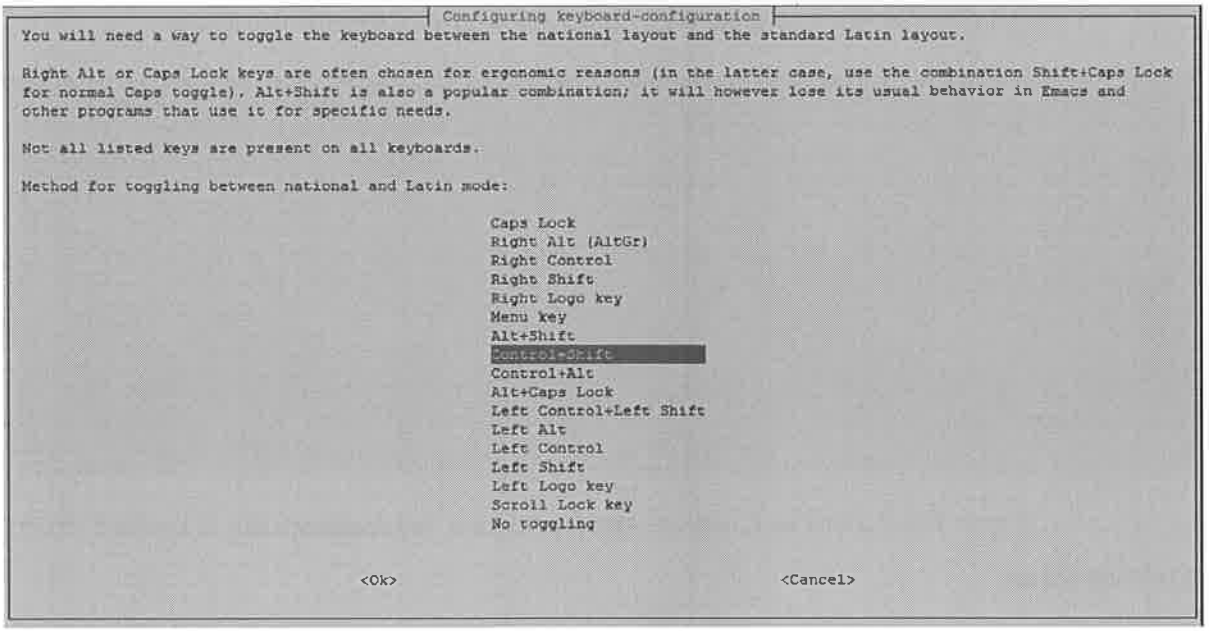
(2.8.4.3) ที่หัวข้อ **Country of origin for the keyboard :** เลือก **Thai** แล้วกด **Enter**



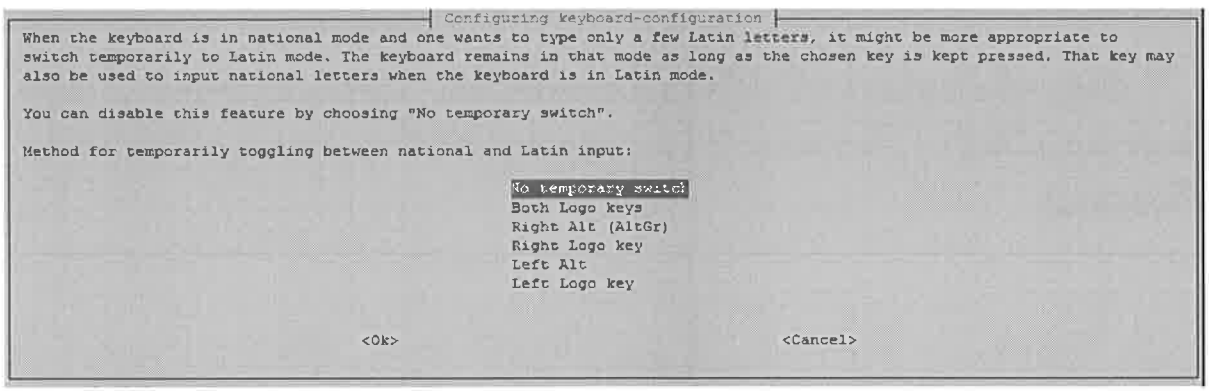
(2.8.4.4) ที่หัวข้อ **Keyboard layout :** เลือก **Thai-Thai (TIS-820.2538)** แล้วกด **Enter**



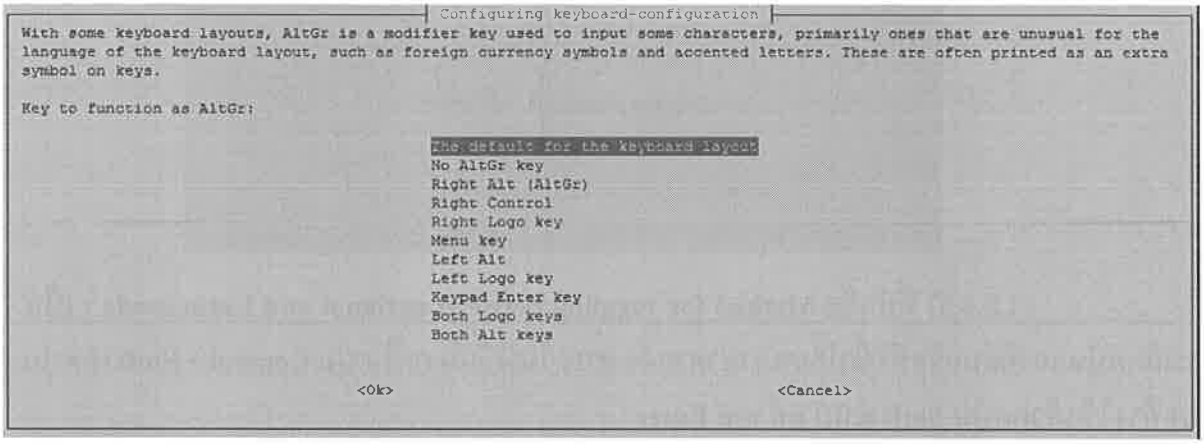
(2.8.4.5) ที่หัวข้อ **Method for toggling between national and Latin mode :** เป็นการเลือกปุ่มของคีย์บอร์ดที่ใช้เปลี่ยนภาษาตามต้องการ ในที่นี้เลือกเป็น ปุ่ม **Control + Shift** (กดปุ่ม Ctrl ค้างไว้แล้วกดปุ่ม Shift ตาม) แล้วกด **Enter**



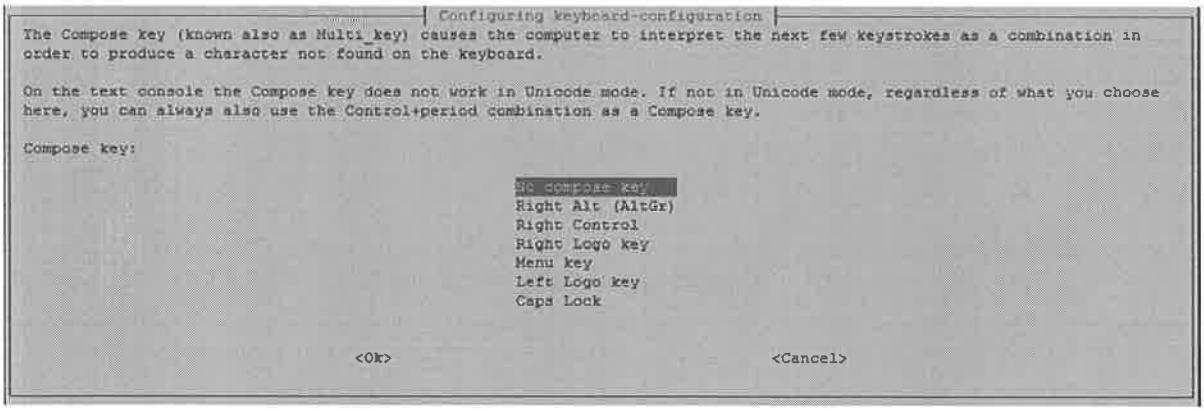
(2.8.4.6) ที่หัวข้อ **Method for temporarily toggling between nation and Latin input :** เลือก **No temporary switch** แล้วกด **Enter**



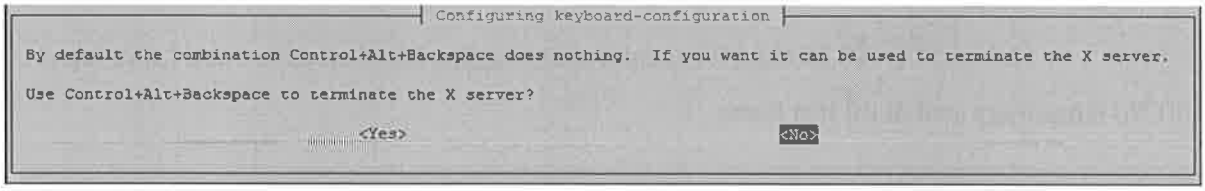
(2.8.4.7) ที่หัวข้อ **Key to function as AltGr :** เลือก **The default for the keyboard layout** แล้วกด Enter



(2.8.4.8) ที่หัวข้อ **Compose key :** เลือก **No compose key** แล้วกด **Enter**



(2.8.5) ที่หน้าต่าง **Use Control+Alt+Backspace to terminate the X server?** เลือก **No** แล้วกด **Enter**



เป็นอันเสร็จเรียบร้อย จากนี้ไปผู้ใช้งานสามารถใช้งานคีย์บอร์ดในรูปแบบที่รองรับทั้งภาษาไทยและอังกฤษได้แล้ว *(จากการทดสอบในขณะจัดทำหนังสือเล่มนี้ การแสดงผลภาษาไทยยังทำงานได้ไม่สมบูรณ์)*

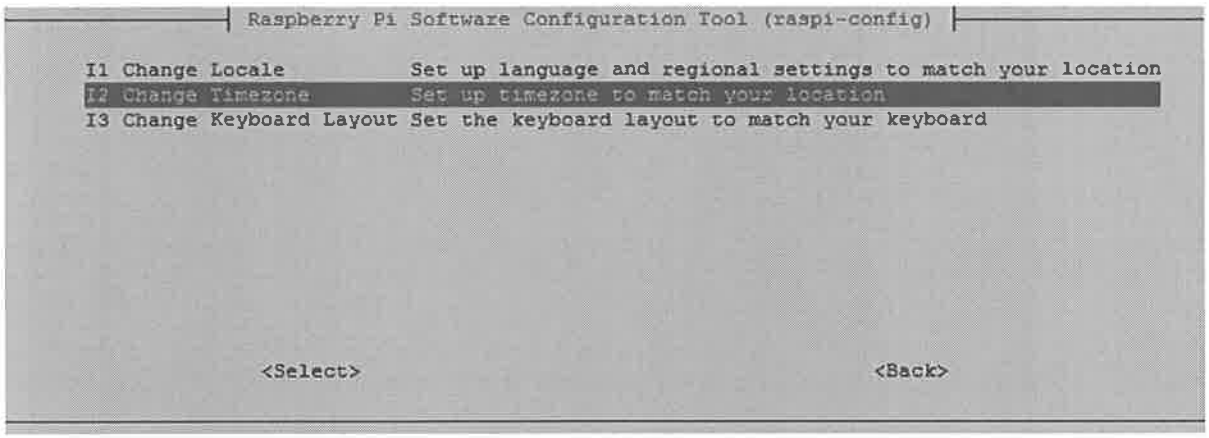
2.9 การตั้งค่าเวลา

บนบอร์ด Raspberry Pi 2 ไม่มีวงจรฐานเวลานาฬิกาจริงหรือรีลไทม์คล็อกในตัว แต่จะอาศัยการตั้งค่าเวลาจากเครือข่ายอินเทอร์เน็ตแทน โดยอ่านจาก **NTP เซิร์ฟเวอร์ (Network Time Protocol server)** เพื่อช่วยให้ระบบมีค่าเวลาที่เที่ยงตรงตามมาตรฐาน อย่างไรก็ตาม บอร์ด Raspberry Pi 2 ยังติดต่อกับวงจรรีลไทม์คล็อกภายนอกเพื่ออ่านค่าเวลาได้ หรือติดต่อกับโมดูล GPS เพื่ออ่านค่าเวลาจากดาวเทียมมากำหนดเป็นค่าเวลาหลักของระบบก็ได้

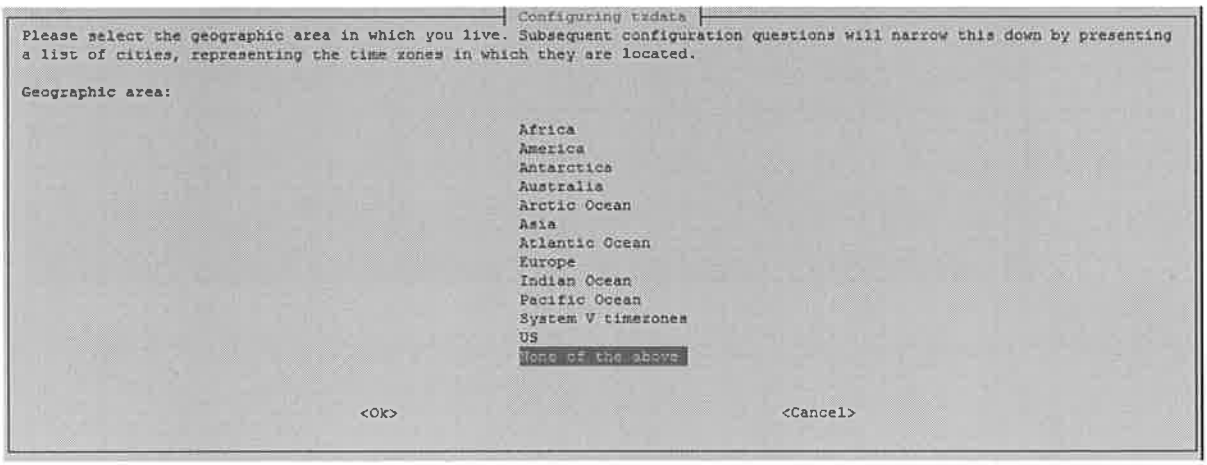
สำหรับในหัวข้อนี้เป็นการตั้งค่าเวลาด้วยการอ่านค่าจากเครือข่ายอินเทอร์เน็ต มีขั้นตอนดังนี้

(2.9.1) พิมพ์คำสั่ง **`sudo raspi-config`**

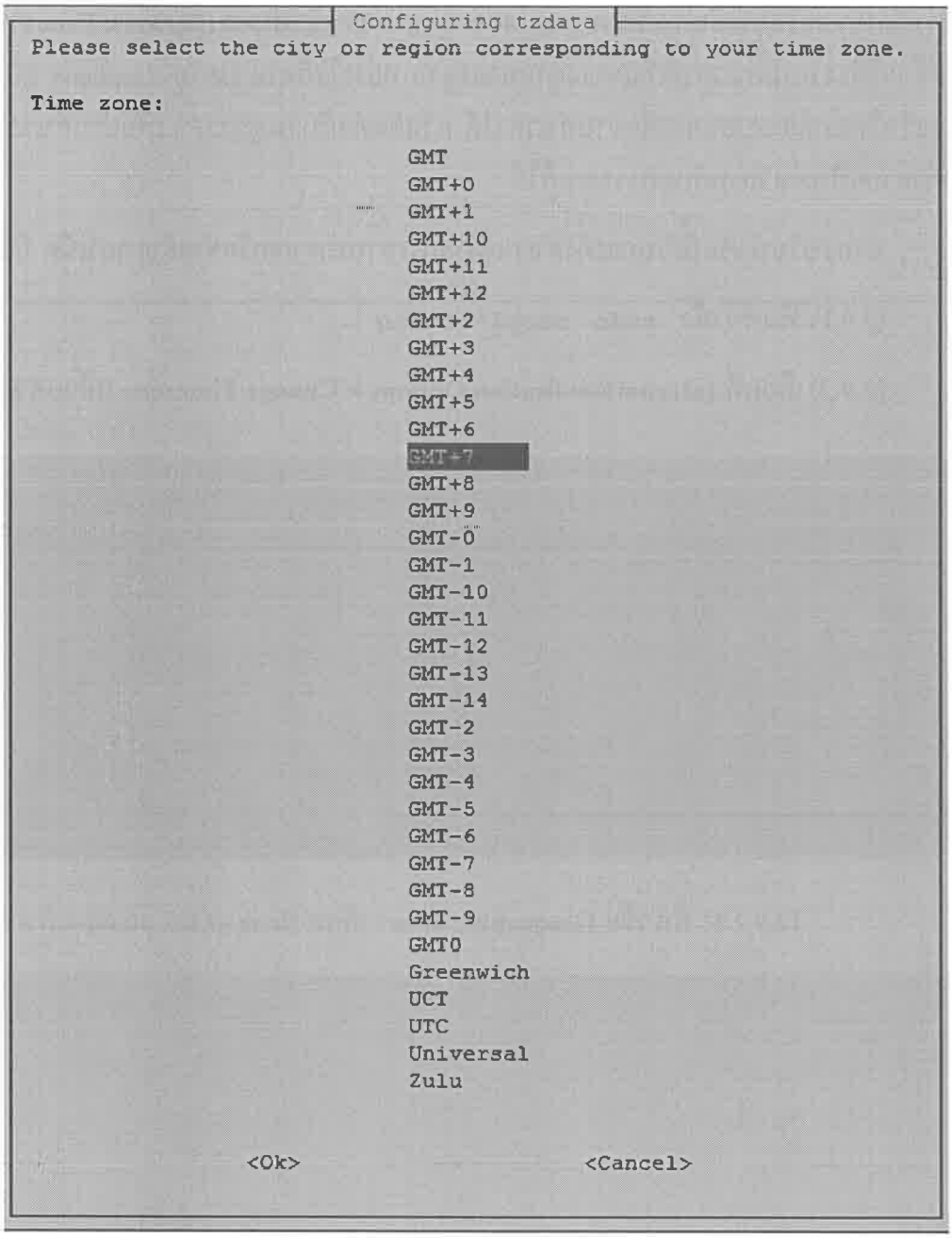
(2.9.2) เลือกที่ **Internationalisation Options > Change Timezone** แล้วกด **Enter**



(2.9.2.1) ที่หัวข้อ **Geographic area :** เลือก **None of the above** แล้วกด **Enter**



(2.9.2.2) ที่หัวข้อ **Time zone :** เลือก **GMT+7** แล้วกด **Enter** เนื่องจากเวลาของประเทศไทยจะเร็วกว่าเวลามาตรฐานกรีนิช (GMT : Greenwich Mean Time) 7 ชั่วโมง

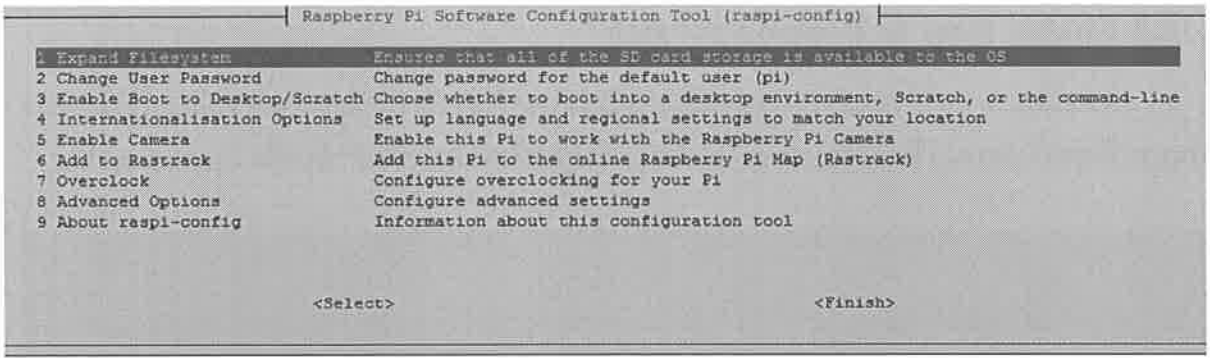


2.10 เกี่ยวกับ Setup Option เพื่อตั้งค่าการทำงานของ Raspberry Pi

เมื่อต้องการตั้งค่าการทำงานใดๆ ก็ตามที่เกี่ยวข้องกับการทำงานหลักของบอร์ด Raspberry Pi 2 จะต้องเข้าสู่หน้าต่าง **Raspberry Pi Software Configuration Tools** โดยพิมพ์คำสั่ง

```
sudo raspi-config
```

จะปรากฏหน้าต่าง **Raspberry Pi Software Configuration Tools** ดังรูปที่ 2-6



รูปที่ 2-6 หน้าต่างหลักของ Raspberry Pi Software Configuration Tools สำหรับตั้งค่าการทำงานของบอร์ด Raspberry Pi ทุกรุ่น

มีรายการสำหรับตั้งค่าการทำงานของ Raspberry Pi ทั้งสิ้น 9 รายการดังนี้

2.10.1. Expand Filesystem

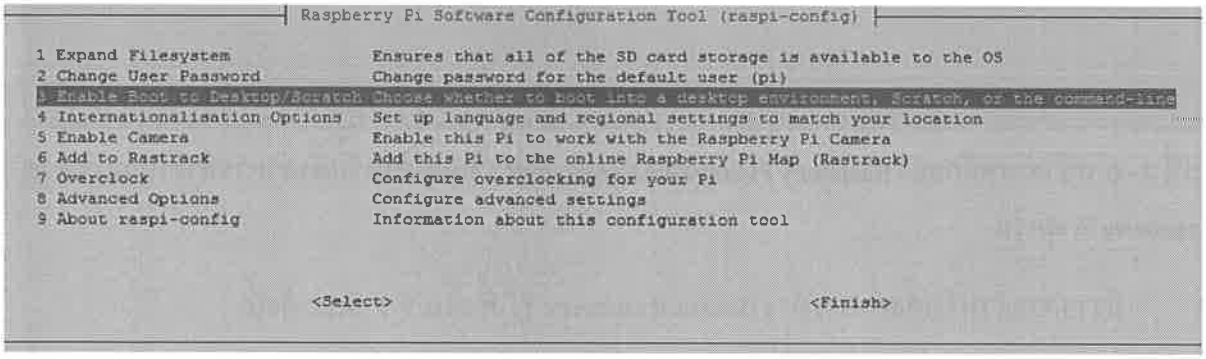
ตั้งค่าเพื่อขยายพื้นที่หน่วยความจำใน SD การ์ดหรือ microSD การ์ดให้บอร์ด Raspberry Pi เข้าถึงและใช้งานได้อย่างเต็มประสิทธิภาพ

2.10.2 Change User Password

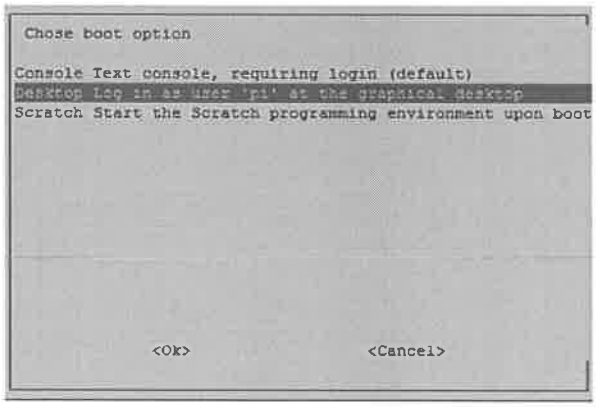
เปลี่ยนรหัสผ่านเพื่อเข้าสู่ระบบ ปกติรหัสผ่านคือ Pi

2.10.3 Enable Boot to Desktop/Scratch

ตั้งค่าเพื่อกำหนดให้ระบบทำการบูตหรือเตรียมระบบเข้าสู่หน้า Desktop ในโหมดกราฟิกโดยตรง หรือบูตเข้าสู่หน้าต่างของโปรแกรม Scratch หรือบูตเข้าสู่หน้าต่างบรรทัดคำสั่งหรือ Comnmand line



เมื่อเลือกเข้าไปที่หัวข้อนี้ หากต้องการให้ Raspberry Pi เริ่มต้นการใช้งานในแบบอัตโนมัติโดยไม่ต้องล็อกอิน ให้เลือกหัวข้อ **Desktop Log in as user 'pi' at the graphical desktop** แล้วกด **Enter**



2.10.4 Internationalisation Options

ตั้งค่าเกี่ยวกับภาษา, รูปแบบของคีย์บอร์ด, เวลา และข้อมูลอื่นๆ ที่เกี่ยวกับพื้นที่ใช้งาน

2.10.5 Enable Camera

เลือกเปิดการทำงานเพื่อติดต่อกับโมดูลกล้องที่ต่อกับจุดต่อ CSI โดยระบบจะทำการสงวนพื้นที่ของหน่วยความจำแรม 128MB เป็นอย่างน้อยสำหรับการประมวลผลของ GPU ชิป ประมวลผลภาพ

2.10.6 Add to Rastrack

เลือกให้บอร์ด Raspberry Pi ทำงานกับระบบแผนที่ออนไลน์ของ Google เพื่อระบุตำแหน่งปัจจุบันของบอร์ด Raspberry Pi

2.10.7 Overclock

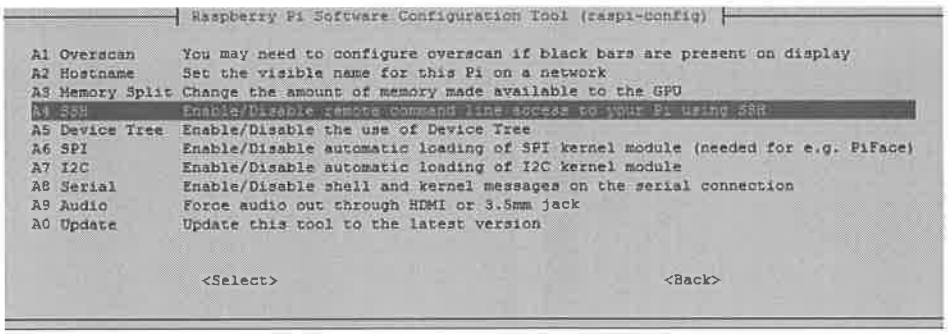
ตั้งค่าการเพิ่มความถี่ของสัญญาณนาฬิกาหลักของ Raspberry Pi เพื่อให้ทำงานได้เร็วขึ้น

2.10.8 Advance Options

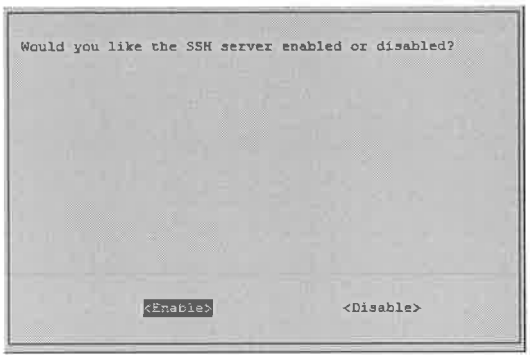
ตั้งค่าการทำงานขั้นสูงเพื่อเปิดการทำงานพิเศษ อาทิ เปิดการทำงานของบัส I^2C, SPI และ SERIAL ของพอร์ต GPIO เพื่อติดต่ออุปกรณ์อิเล็กทรอนิกส์ภายนอก, เปิดการทำงาน SSH เพื่อให้คอมพิวเตอร์ภายนอกต่อเข้ามารีโมตการทำงานของบอร์ด Raspberry Pi ได้, กำหนดการขับสัญญาณเสียง เป็นต้น

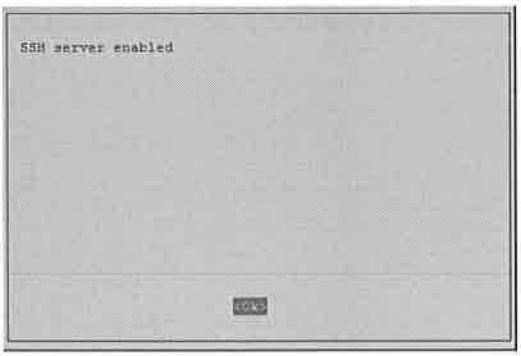
2.10.8.1 SSH

เปิดความสามารถการรีโมตผ่านช่องทาง SSH



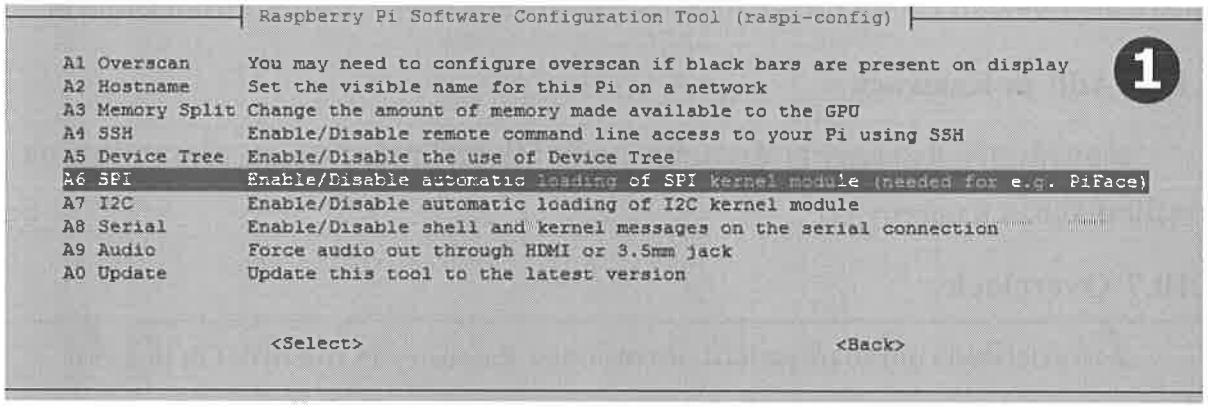
เลือก **Enable** แล้วกด **Enter** จะแสดงหน้าต่างยืนยันการเปิดใช้งาน SSH เลือกคลิกปุ่ม **OK**

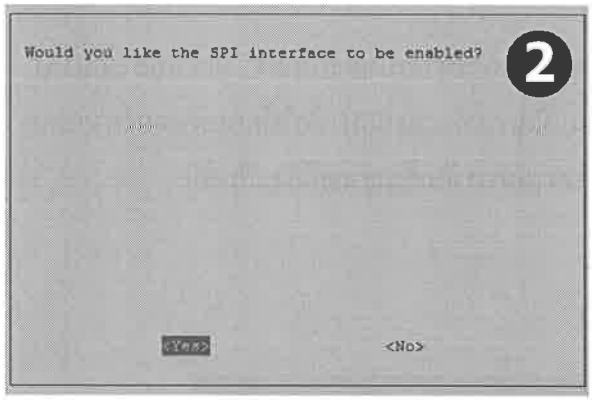


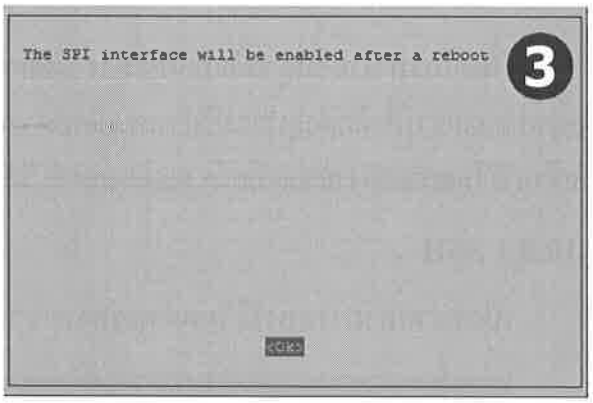


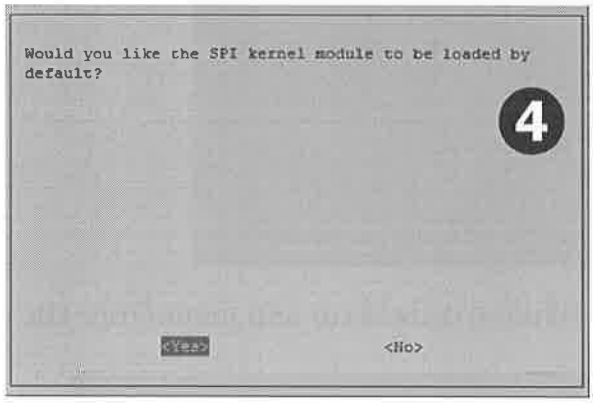
2.10.8.2 SPI

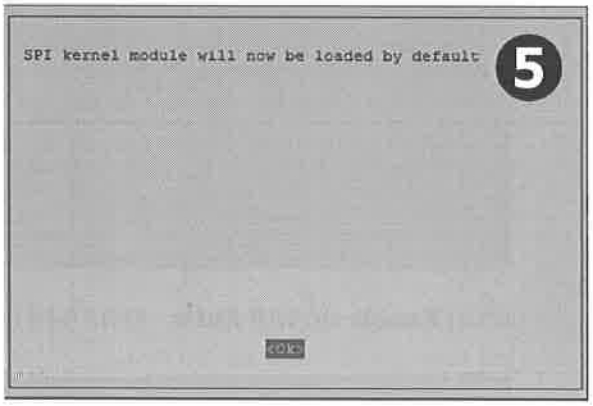
เปิดความสามารถการเชื่อมต่อกับอุปกรณ์บัส SPI โดยเลือก **Yes**





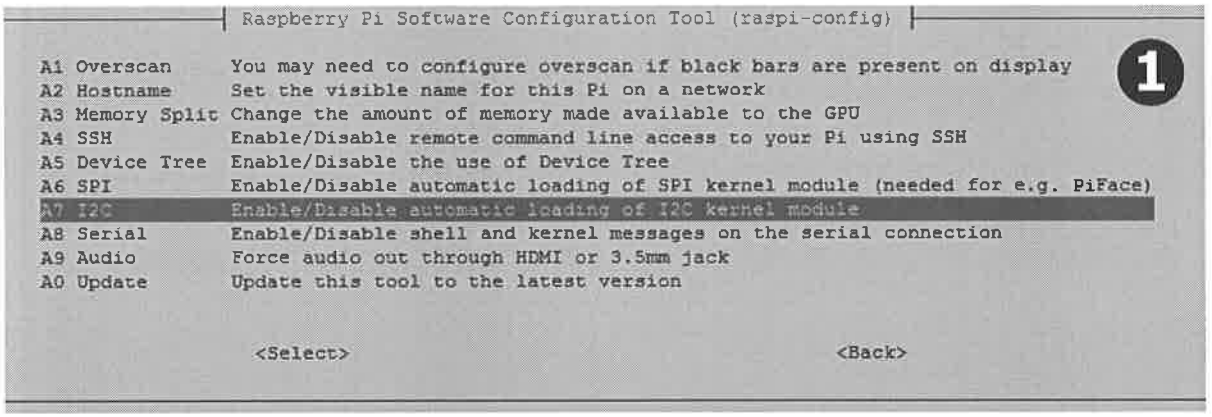


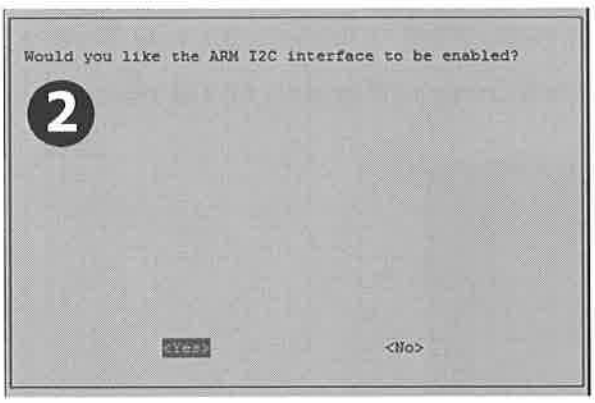


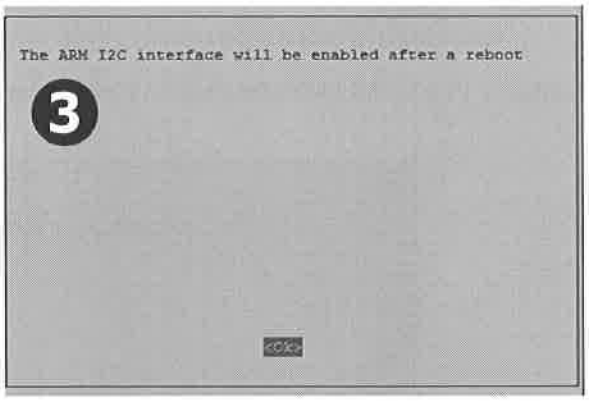


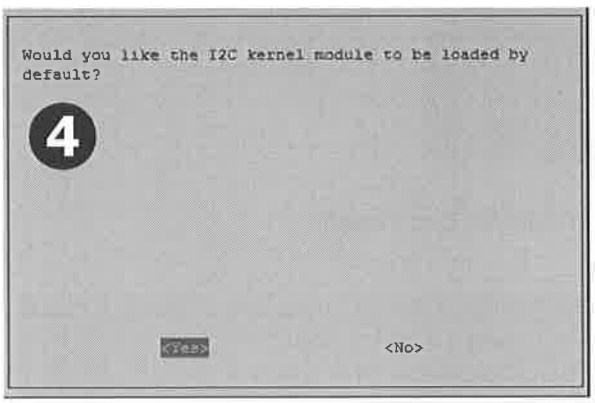
2.10.8.3 I2C

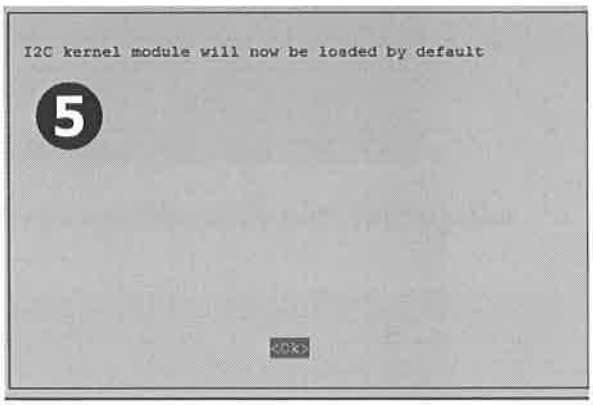
เปิดความสามารถการเชื่อมต่อกับอุปกรณ์บัส I²C โดยกดปุ่ม **Select** และ **Yes** เพื่อตอบรับ





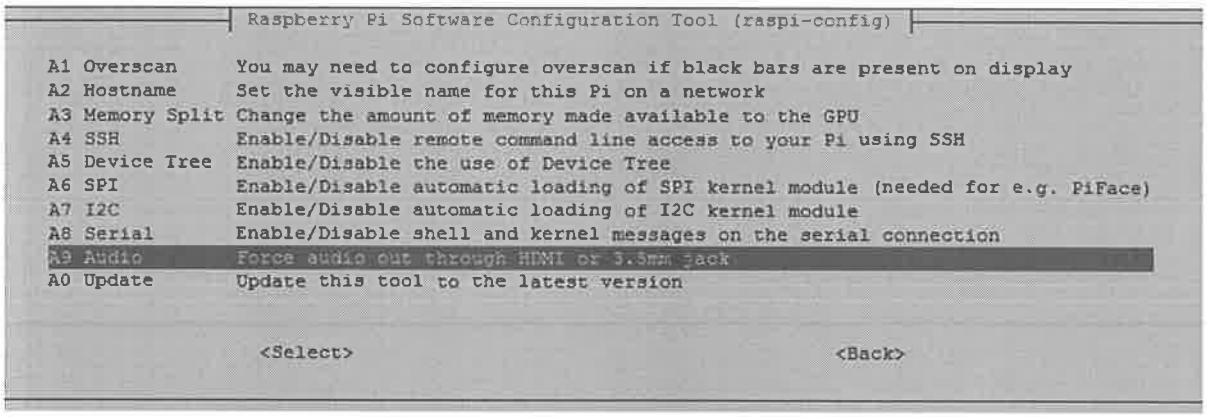




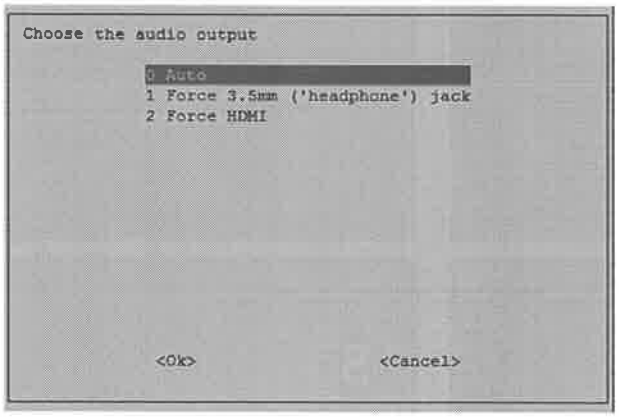


2.10.8.4 Audio

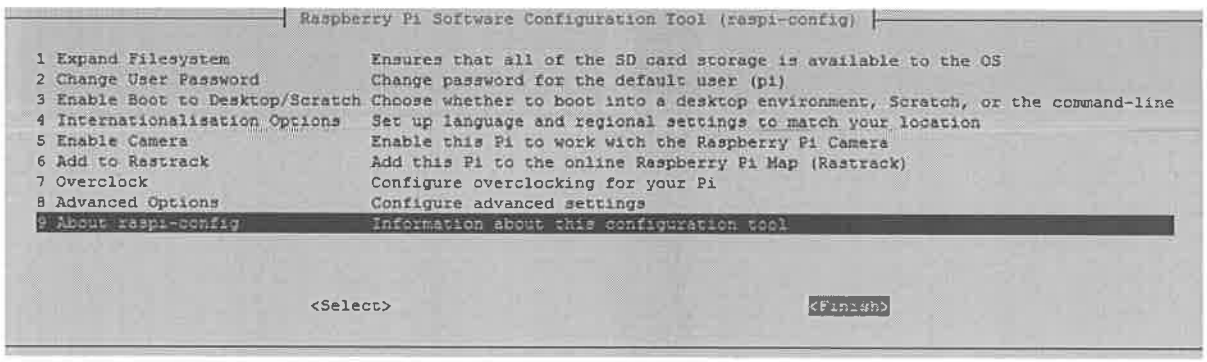
เปิดช่องทางการส่งสัญญาณเสียงออกจากบอร์ด Raspberry Pi 2



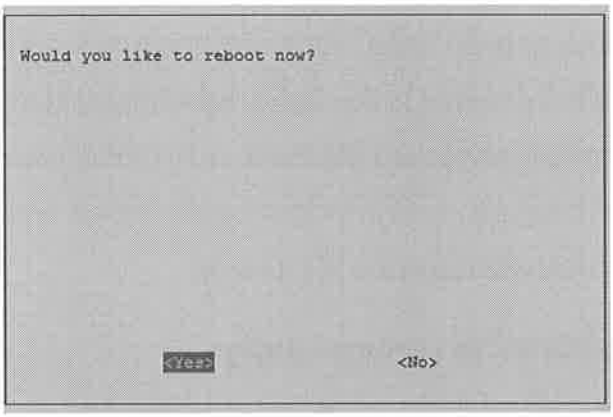
โดยเลือกให้สัญญาณเสียงออกทางแจ๊ก AV หรือจุดต่อ HDMI ในที่นี้เลือกแบบ **Auto** ซึ่งบอร์ด Raspberry Pi 2 จะตรวจสอบเองเมื่อมีการเชื่อมต่ออุปกรณ์เข้ากับช่องเอาต์พุตนั้นๆ กด **OK** เพื่อตอบรับ



กลับสู่หน้าต่างหลักหลังจากปรับแต่งตั้งค่า จากนั้นเลือกไปที่ **Finish**



แล้วกดคีย์ **Enter** หน้าต่างแจ้งให้รีบูตระบบปรากฏขึ้นมา เลือก **Yes** แล้วกด **Enter** ตัวบอร์ดจะทำการรีบูต



หลังจากการรีบูตแล้ว จะพบว่า ระบบจะทำการล็อกอินเข้ายูสเซอร์ Pi ให้อัตโนมัติ เป็นอันเสร็จสิ้นการตั้งค่าพื้นฐาน

2.10.9 About raspi-config

เลือกเพื่อแสดงข้อมูลเกี่ยวกับหน้าต่าง **Raspberry Pi Software Configuration Tools** ที่ใช้ในการตั้งค่าการทำงานของ Raspberry Pi

2.11 การตั้งค่าช่องเอาต์พุตเสียงของ Raspberry Pi

บอร์ด Raspberry Pi 2 มีเอาต์พุตสัญญาณเสียง 2 ช่องคือ **ผ่านทางจุดต่อพอร์ต HDMI** และ **แจ๊ค AV 3.5 มม.** ที่เป็นทั้งเอาต์พุตสัญญาณเสียงสเตอริโอและสัญญาณภาพ การขับเสียงออกของบอร์ด Raspberry Pi 2 โดยปกติจะเป็นไปอย่างอัตโนมัติ นั่นคือ ชิปหลักจะตรวจสอบว่า มีการต่ออุปกรณ์เข้ามาที่เอาต์พุตใด ก็จะให้ส่งสัญญาณเสียงออกไปยังช่องทางนั้น แต่ถ้าหากต่อพร้อมกัน โดยปกติแล้ว บอร์ด Raspberry Pi จะเลือกขับสัญญาณเสียงออกไปทางพอร์ต HDMI อย่างไรก็ตาม ผู้ใช้งานสามารถตั้งค่าเพื่อกำหนดช่องเอาต์พุตเสียงที่ต้องการใช้งานเองได้

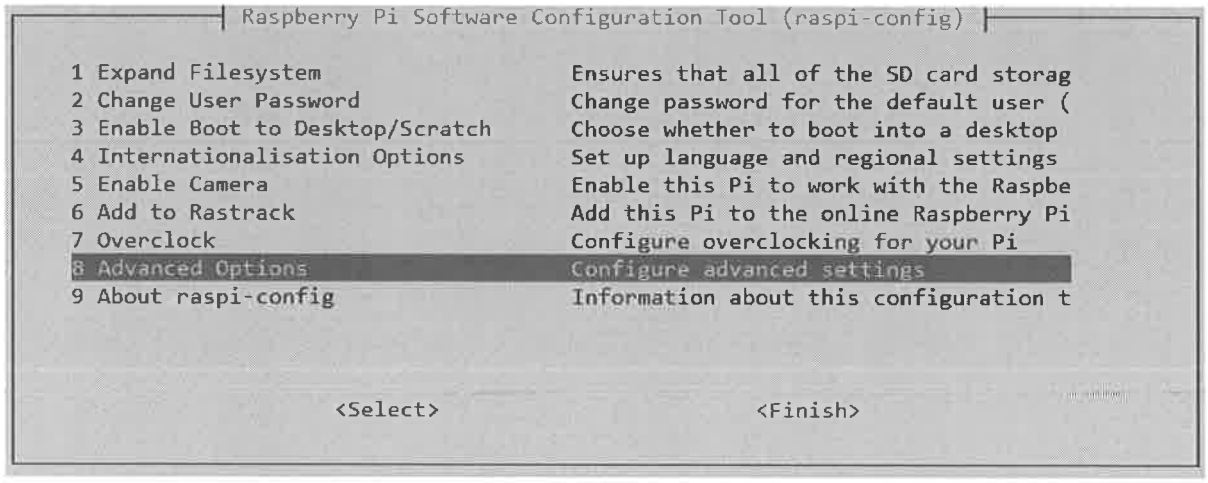
2.11.1 ตั้งค่าช่องเอาต์พุตเสียงผ่านหน้าต่าง raspi-config

อย่างไรก็ตาม ผู้ใช้งานสามารถตั้งค่าเพื่อกำหนดช่องเอาต์พุตเสียงที่ต้องการใช้งานเองได้ โดยเข้าสู่หน้าต่างเทอร์มินอล แล้วพิมพ์คำสั่งเพื่อเรียกหน้าต่าง **Raspberry Pi Software Configuration** ซึ่งเป็นหน้าต่างสำหรับตั้งค่าการทำงานของ Raspberry Pi ขึ้นมา ดังนี้

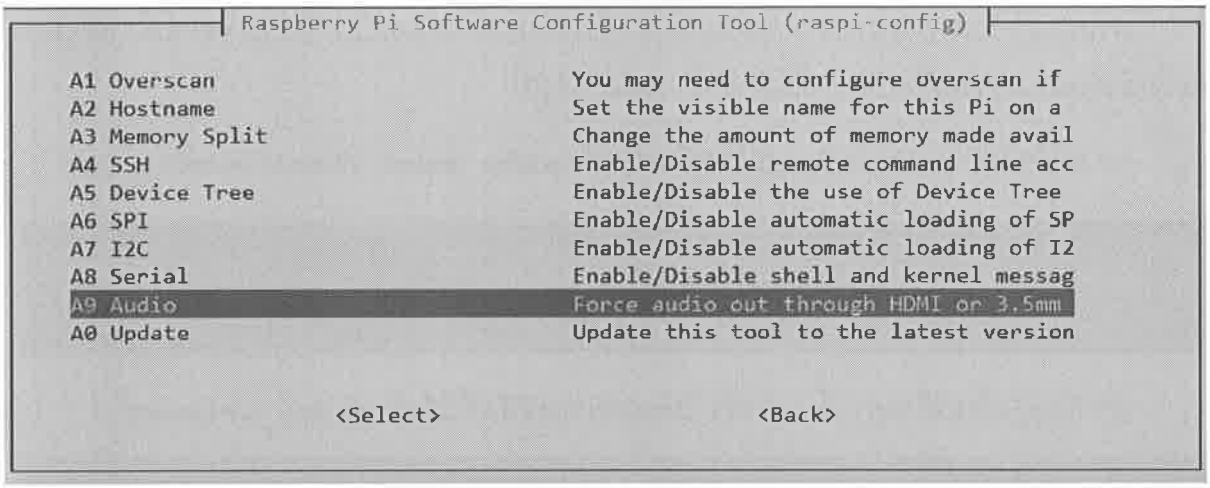
(1) พิมพ์คำสั่ง `sudo raspi-config`

```
pi@raspberrypi ~ $ sudo raspi-config
```

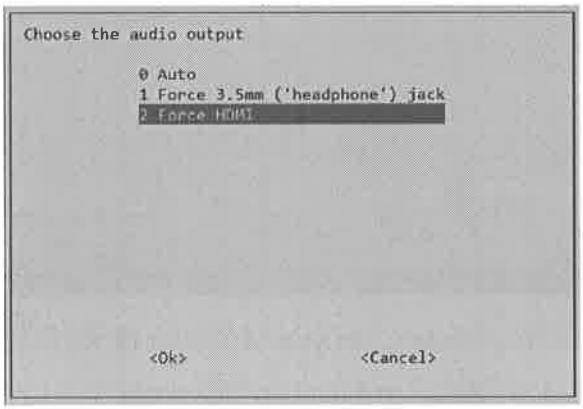
(2) เลือกไปที่ **Advanced options**



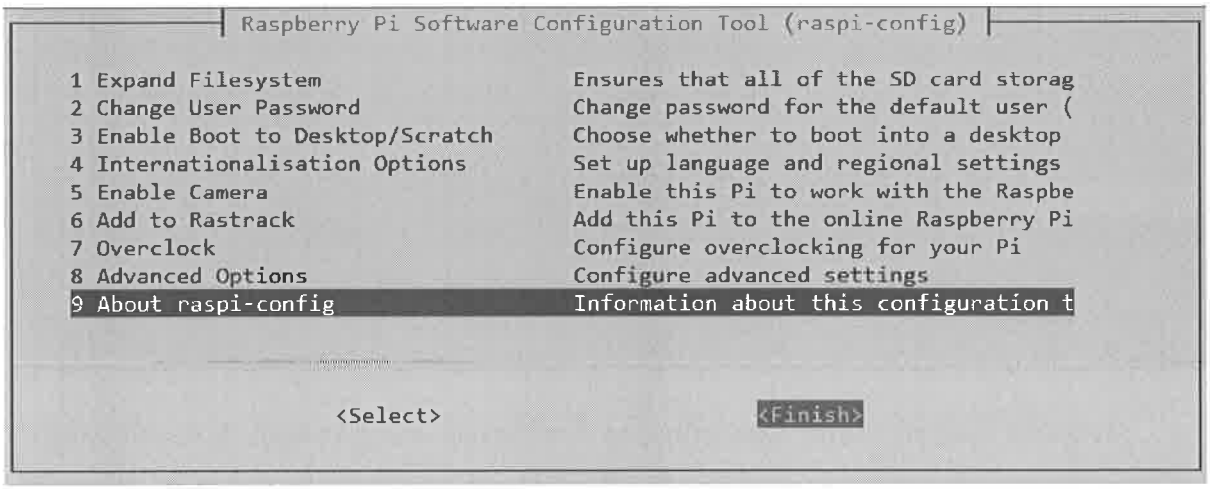
(3) เลือก **A9 Audio**



(4) เลือกช่องเอาต์พุตเสียงหรือ **Audio Ouput** ตามที่ต้องการ จากตัวอย่างจะเลือกรายการที่ 2 **Force HDMI** เป็นการเลือกให้เสียงออกเฉพาะพอร์ต HDMI แล้วกดปุ่ม **Enter**



(5) จะกลับมายังหน้าต่างหลัก เลือก **Finish** เพื่อปิดหน้าต่าง **raspi-config** เพียงเท่านี้ก็เสร็จสิ้นการตั้งค่าเอาต์พุตเสียงของบอร์ด Raspberry Pi แล้ว ทำการรีบูตอีกครั้ง



2.11.2 การแก้ไขปัญหาเกี่ยวกับช่องเอาต์พุตเสียง

สำหรับผู้ใช้งานบางท่านอาจพบปัญหาว่า *เมื่อกำหนดเอาต์พุตเสียงเป็น HDMI แล้ว แต่เสียงก็ยังไม่ดังออกมา* อาจแก้ไขปัญหาได้ด้วยวิธีการดังต่อไปนี้

(1) แก้ไขไฟล์ **/boot/config.txt** โดยพิมพ์คำสั่ง **sudo nano /boot/config.txt**

```
pi@raspberrypi ~/Desktop $ sudo nano /boot/config.txt
```

(2) เมื่อเปิดไฟล์ดังกล่าวขึ้นมาแล้ว ให้มองหาบรรทัดที่มีคำว่า **#hdmi_drive=2**

```
# uncomment if hdmi display is not detected and composite is being output
#hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1

# uncomment to force a HDMI mode rather than DVI. This can make audio work in
# DMT (computer monitor) modes
#hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4
```

(3) ลบเครื่องหมาย **#** ออก เพื่อให้คำสั่ง **hdmi_drive=2** ทำงาน (**#** คือทำให้ข้อความที่อยู่ต่อท้ายกลายเป็นหมายเหตุหรือ comment ทำให้คำสั่งหลังจากนั้นไม่ทำงาน)

```
# uncomment if hdmi display is not detected and composite is being output
#hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1

# uncomment to force a HDMI mode rather than DVI. This can make audio work in
# DMT (computer monitor) modes
hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4
```

(4) บันทึกไฟล์ให้เรียบร้อย โดยกดปุ่ม **Ctrl** ค้างไว้ตามด้วย **X** แล้วกดปุ่ม **Y** ตามด้วยกดปุ่ม **Enter** จากนั้นทำการรีบูตระบบ การตั้งค่าเอาต์พุตเสียงให้ออกทางจุดต่อพอร์ต HDMI ก็จะใช้งานได้

2.12 ข้อควรทราบเกี่ยวกับความจุของ microSD การ์ด หลังจากติดตั้งระบบปฏิบัติการ

หลังจากติดตั้งระบบปฏิบัติการบนบอร์ด Raspberry Pi 2 จะมีการแบ่งพื้นที่ของ microSD การ์ดออกเป็นหลายส่วน *หากนำ microSD การ์ดไปใช้งานปกติเหมือนเดิมหรือนำไปตรวจสอบพื้นที่ว่างด้วย Windows Explorer บนระบบปฏิบัติการวินโดวส์* จะพบว่า **ความจุของ microSD การ์ดลดลงมาก** ทั้งนี้เนื่องจากระบบปฏิบัติการณ์วินโดวส์ไม่สามารถมองเห็นพื้นที่ดังกล่าวได้

สิ่งที่เกิดขึ้นจึงไม่ใช่เรื่องผิดปกติอย่างใด เมื่อนำกลับมาทำงานบนระบบปฏิบัติการ Raspbian บนตัวบอร์ด Raspberry Pi 2 จะเห็นพื้นที่ของ microSD การ์ดทั้งหมดเองตามปกติ

2.13 การเปลี่ยนรหัสผ่านผู้ใช้งาน

เป็นที่ทราบกันอยู่แล้วว่า เมื่อเริ่มต้นใช้งานบอร์ด Raspberry Pi นั้น จะมีการกำหนดค่าเริ่มต้นของชื่อผู้ใช้คือ **pi** และรหัสผ่านคือ **raspberry**

หากมีการนำ Raspberry Pi ไปใช้ในการผลิตเป็นสินค้า ถ้าไม่เปลี่ยนรหัสผ่าน น่าจะไม่ปลอดภัย เพราะอาจมีผู้ไม่ประสงค์ดีแอบเข้ามาทำอะไรกับ Raspberry Pi ได้ ดังนั้นทางที่ดีจึงควรเปลี่ยนรหัสผ่านเพื่อความปลอดภัย

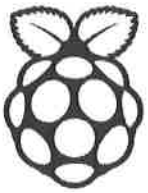
การเปลี่ยนรหัสผ่านทำได้ด้วยการพิมพ์คำสั่ง **passwd**

```
pi@raspberrypi ~ $ passwd
```

ระบบจะให้กรอกรหัสผ่านปัจจุบันก่อน แล้วตามด้วยรหัสผ่านใหม่ แล้วทวนรหัสผ่านใหม่อีกครั้ง โดยในการกำหนดรหัสผ่านจะไม่มีการแสดงข้อความให้เห็นเพื่อความปลอดภัย เมื่อกำหนดครบแล้ว เป็นอันเสร็จสิ้นการกำหนดรหัสผ่าน

```
pi@raspberrypi ~ $ passwd
Changing password for pi.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

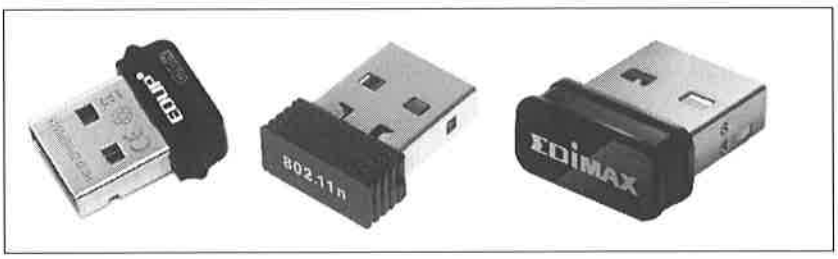
บทที่ 3
Raspberry Pi กับการเชื่อมต่อระบบเครือข่ายไร้สาย WiFi และการควบคุมผ่าน SSH



บอร์ด Raspberry Pi 2 มีจุดต่อระบบเครือข่ายหรือ LAN ผ่านจุดต่ออีเธอร์เน็ตที่เป็นคอนเน็กเตอร์แบบ RJ-45 และด้วยการใช้ระบบปฏิบัติการ Raspbian ในเวอร์ชันใหม่ๆ ได้บรรจุความสามารถนี้ไว้เรียบร้อย ทันทีที่บูตระบบเสร็จสิ้น บอร์ด Raspberry Pi 2 จะสามารถติดต่อกับระบบเครือข่ายแบบใช้สายได้ทันที รวมถึงการเชื่อมต่อกับอินเทอร์เน็ตด้วย หากเครือข่ายที่ทำการเชื่อมต่อนั้นเปิดการติดต่อเครือข่ายอินเทอร์เน็ตไว้แล้ว

ในการใช้งานสมัยใหม่ การเชื่อมต่อเครือข่ายมุ่งสู่การเชื่อมต่อแบบไร้สายผ่าน WiFi *(ถ้าเรียกให้ถูกต้องคือ WLAN หรือ Wireless Local Area Network ตามมาตรฐาน IEEE802.11– คำว่า WiFi เป็นชื่อเครื่องหมายการค้าของกลุ่ม Wi-Fi Alliance อันเป็นการรวมตัวกันของบริษัทผู้ผลิตสินค้าและบริการที่เกี่ยวข้องกับอุปกรณ์ระบบเครือข่ายไร้สายหรือ WLAN อาทิ Nokia, Intersil, 3COM, Motorola ภายใต้การสนับสนุนจาก Apple, Samsung, LG, Sony, Microsoft, Texas Instrument และ T-mobile)* บอร์ด Raspberry Pi 2 ไม่ได้มีวงจรหรือโมดูล WiFi (WLAN) ติดตั้งอยู่ในตัว หากต้องการเชื่อมต่อระบบเครือข่ายแบบไร้สาย จะต้องใช้อุปกรณ์ที่เรียกว่า **USB WiFi ดองเกิล** ต่อเข้าทางพอร์ต USB จากนั้นทำการตั้งค่าเล็กน้อย ก็จะทำให้บอร์ด Raspberty Pi 2 (หรือรุ่นอื่นๆ) เชื่อมต่อระบบเครือข่ายในแบบไร้สายได้ หรือใช้งานอินเทอร์เน็ตผ่าน WiFi ได้ด้วยเช่นกัน

อย่างไรก็ตาม ไม่ใช่อุปกรณ์ USB WiFi ดองเกิลทุกตัวที่มีขายในท้องตลาดจะสามารถใช้งานกับบอร์ด Raspberry Pi ได้ จึงต้องเลือกให้ดี เพราะมีบางรุ่นที่ไม่รองรับหรือไม่อาจติดต่อกับชิปควบคุมหลักบนบอร์ด Raspberry Pi ได้ จึงใช้งานไม่ได้ **รุ่นที่ใช้งานได้ส่วนใหญ่จะใช้ชิป Realtek RTL8188CU สังเกตจากบรรจุภัณฑ์หรือกล่องบรรจุที่มักจะระบุไว้อย่างชัดเจน**



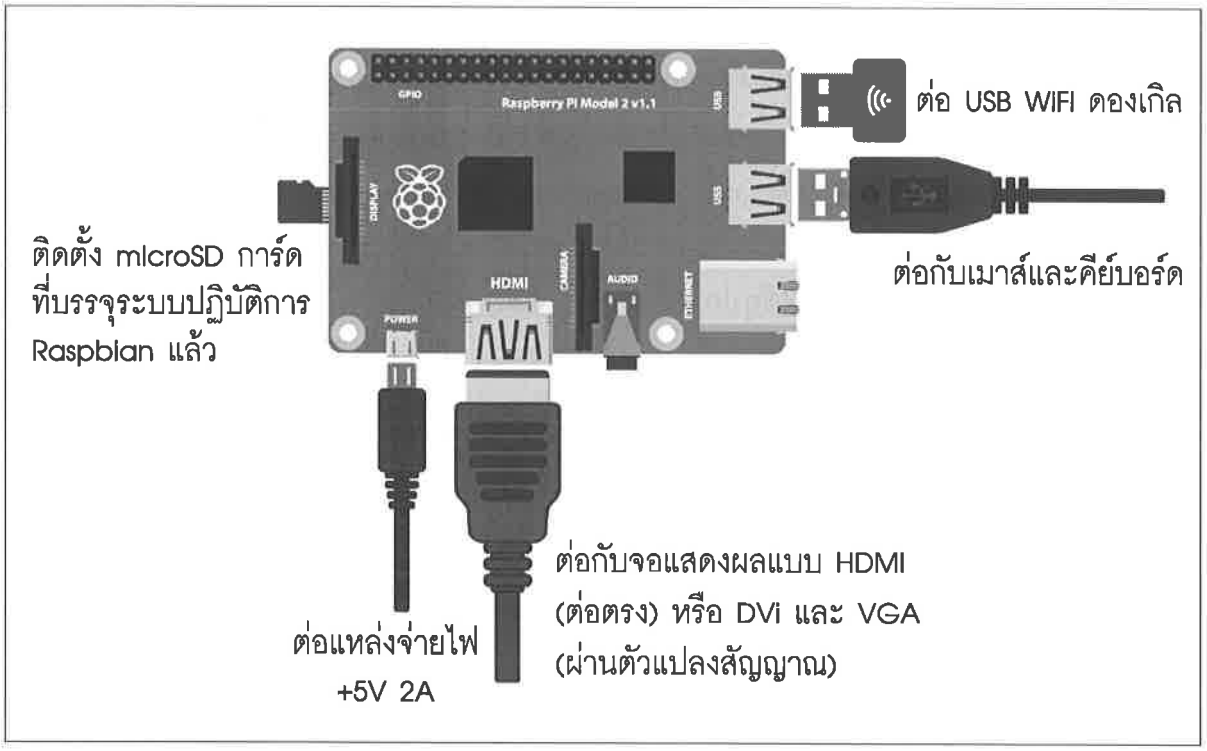
รูปที่ 3–1 ตัวอย่างของ USB WiFi ดองเกิลที่ใช้งานกับบอร์ด Raspberry Pi 2 ได้ (ใช้ได้กับ Raspberry Pi ทุกรุ่น)

3.1 การตั้งค่าการทำงานของ Raspberry Pi 2 เพื่อเชื่อมต่อ WiFi ผ่าน Terminal

ในระบบปฏิบัติการ Raspbian เวอร์ชันใหม่ๆ มีการตั้งค่าเพื่อให้บอร์ด Raspberry Pi ติดต่อกับ USB WiFi ดองเกิลไว้แล้ว จึงต้องไม่ต้องตั้งค่าระบบเพื่อให้มองเห็น USB WiFi ดองเกิล แต่อย่างใด *สิ่งที่จะต้องตั้งค่าจึงมีเพียงการกำหนดเครือข่ายที่ต้องการเชื่อมต่อเท่านั้น*

สำหรับการใช้งาน Raspbian ในโหมดกราฟิกหรือ GUI มีโปรแกรมสำหรับเชื่อมต่อ WiFi ไว้อยู่แล้ว แต่การใช้งานผ่านเทอร์มินอล (Terminal) หรือสั่งงานด้วยคอมมานด์ไลน์ (Command Line) มีวิธีการเชื่อมต่อที่ต่างกันไปดังนี้

(3.1.1) ตรวจสอบการเชื่อมต่อบอร์ด Raspberry Pi 2 กับอุปกรณ์ต่างๆ ทั้งสายไฟเลี้ยง, สายต่อจอภาพ HDMI, คีย์บอร์ด, เมาส์, Wi-Fi ดองเกิล และ SD การ์ดที่ติดตั้งระบบปฏิบัติการ Raspbian เวอร์ชันล่าสุด ดังรูปที่ 3-2



รูปที่ 3-2 การเชื่อมต่ออุปกรณ์ต่างเข้ากับบอร์ด Raspberry Pi 2 เพื่อเชื่อมต่อเครือข่ายแบบไร้สายผ่าน WiFi

(3.1.2) จ่ายไฟ แล้วเข้าสู่หน้าต่างเทอร์มินอล แล้วพิมพ์คำสั่ง **sudo nano /etc/network/interfaces** เพื่อตรวจสอบการกำหนดค่า Network Connection ของระบบ

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces
```

จากคำสั่งข้างต้น **nano** ในที่นี้ก็คือโปรแกรมเท็กซ์เอดิเตอร์ตัวหนึ่ง ไฟล์ที่เปิดคือ ไฟล์ระบบสำหรับตั้งค่าเครือข่าย เมื่อรันคำสั่ง โปรแกรมจะแสดงข้อมูลดังรูป

```
  GNU nano 2.2.6     File: /etc/network/interfaces

auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf

iface default inet dhcp
```

wlan0 คือ Wireless LAN หรือ WiFi ที่บอร์ด Raspberry Pi 2 ใช้เชื่อมต่อกับเครือข่ายและอินเทอร์เน็ต

(3.1.3) ถ้าต้องการปิดไฟล์ดังกล่าว ให้กด **Ctrl** และ **X** ในกรณีที่มีการแก้ไขไฟล์ข้อมูล โปรแกรมจะสอบถามว่า ต้องการให้บันทึกด้วยหรือไม่ ให้กดปุ่ม **Y** เพื่อทำการบันทึกไฟล์

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
 Y Yes
 N No           ^C Cancel
```

(3.1.4) จากนั้นโปรแกรมจะถามชื่อไฟล์ที่ต้องการบันทึก ให้กดปุ่ม **Enter** เพื่อบันทึกทับไฟล์เดิม

```
File Name to Write: /etc/network/interfaces
^G Get Help    M-D DOS Format  M-A Append      M-B Backup File
^C Cancel      M-M Mac Format  M-P Prepend
```

(3.1.5) ลำดับต่อไปเป็นการตั้งค่า WiFi ที่ต้องการเชื่อมต่อ ให้พิมพ์คำสั่ง **sudo nano /etc/wpa_supplicant/wpa_supplicant.conf**

```
pi@raspberrypi ~ $ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

(3.1.6) ทำการตั้งค่า WiFi ที่จะเชื่อมต่อ โดยกำหนดชื่อ WiFi ที่ต้องการเชื่อมต่อใน **ssid** และกำหนดรหัสผ่านที่บรรทัด **psk** ส่วนพารามิเตอร์อื่นๆ ให้กำหนดตามรูปต่อไปนี้

```
 GNU nano 2.2.6    File: /etc/wpa_supplicant/wpa_supplicant.conf

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
        ssid="WIFI_SSID"
        psk="WIFI_PASSWORD"
        proto=RSN
        key_mgmt=WPA-PSK
        pairwise=TKIP
        auth_alg=OPEN
}
```

(3.1.7) เมื่อกำหนดเสร็จแล้ว บันทึกแล้วปิดไฟล์ จากนั้นรีบูตระบบด้วยคำสั่ง **sudo reboot**

```
pi@raspberrypi ~ $ sudo reboot
```

(3.1.8) เมื่อรีบูตระบบเสร็จสิ้น พิมพ์คำสั่ง **ifconfig** เพื่อตรวจสอบสถานะของการเชื่อมต่อเครือข่ายหรือ Network Connection

```
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:ee:d2:29
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1104 (1.0 KiB)  TX bytes:1104 (1.0 KiB)

wlan0     Link encap:Ethernet  HWaddr 48:02:2a:cd:49:e1
          inet addr:192.168.1.37  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8577 errors:0 dropped:91 overruns:0 frame:0
          TX packets:852 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:865284 (845.0 KiB)  TX bytes:148687 (145.2 KiB)
```

ให้สังเกตที่ **wlan0** จะเห็นว่า บรรทัดที่ 2 ตรงคำว่า **inet addr** มีเลข IP แอดเดรสอยู่ นั่นหมายความว่า เชื่อมต่อ WiFi ได้แล้ว และ IP แอดเดรสนั้นก็คือ IP แอดเดรสของบอร์ด Raspberry Pi

(3.1.9) **ในกรณีที่เชื่อมต่อไม่ได้ จะไม่มี `inet addr` แสดงให้เห็น** ทำการตรวจสอบการตั้งค่า **ssid** และ**รหัสผ่าน**จากขั้นตอน (3.1.5) ใหม่อีกครั้ง รวมถึงตรวจสอบด้วยว่า ตัวปล่อยสัญญาณ WiFi หรือเราเตอร์อยู่ในระยะที่เชื่อมต่อกันได้หรือไม่

(3.1.10) การตรวจสอบการเชื่อมต่ออินเทอร์เน็ตทำได้ง่ายๆ ด้วยการ Ping สัญญาณไปที่ Google โดยพิมพ์คำสั่ง **`ping www.google.com`**

```
pi@raspberrypi ~ $ ping www.google.com
PING www.google.com (173.194.120.147) 56(84) bytes of data.
64 bytes from kul06s08-in-f19.1e100.net (173.194.120.147): icmp_req=1 ttl=52 time=34.9 ms
64 bytes from kul06s08-in-f19.1e100.net (173.194.120.147): icmp_req=2 ttl=52 time=32.4 ms
```

จะเห็นข้อความที่ส่งกลับมารวมไปถึงระยะเวลาของการ Ping สัญญาณด้วย

นั่นหมายความว่า บอร์ด Raspberry Pi 2 เชื่อมต่อกับเครือข่ายอินเทอร์เน็ตได้แล้ว

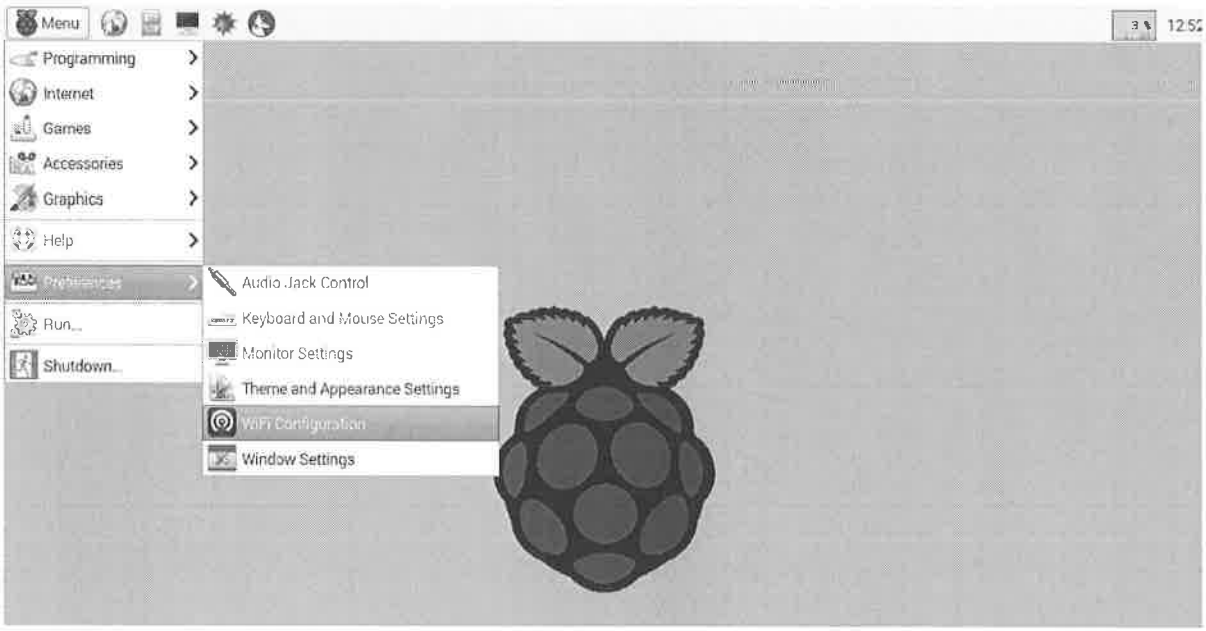
3.2 การตั้งค่าการทำงานของ Raspberry Pi 2 เพื่อเชื่อมต่อ WiFi ผ่านโปรแกรม WiFi Configuration

การเชื่อมต่อเครือข่ายอินเทอร์เน็ตกับบอร์ด Raspberry Pi 2 ในแบบไร้สายผ่าน WiFi ในโหมดกราฟิกหรือ GUI ต้องกระทำผ่านโปรแกรม WiFi Configuration ซึ่งมีขั้นตอนดังนี้

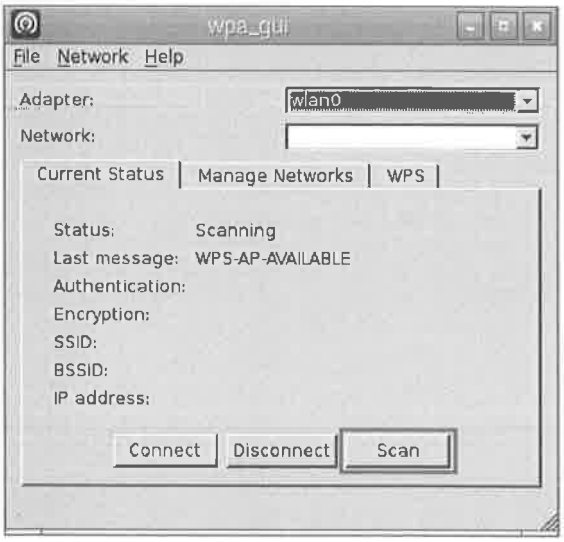
(3.2.1) ตรวจสอบการเชื่อมต่อบอร์ด Raspberry Pi 2 กับอุปกรณ์ต่างๆ ทั้งสายไฟเลี้ยง, สายต่อจอภาพ HDMI, คีย์บอร์ด, เมาส์, Wi-Fi ดองเกิล และ SD การ์ดที่ติดตั้งระบบปฏิบัติการ Raspbian เวอร์ชันล่าสุด เหมือนกับการตั้งค่าผ่านเทอร์มินอล

(3.2.2) จ่ายไฟแล้วบูตระบบเข้าสู่โหมดกราฟิก ด้วยการพิมพ์ **`startx`** หลังจากบูตเสร็จแล้ว

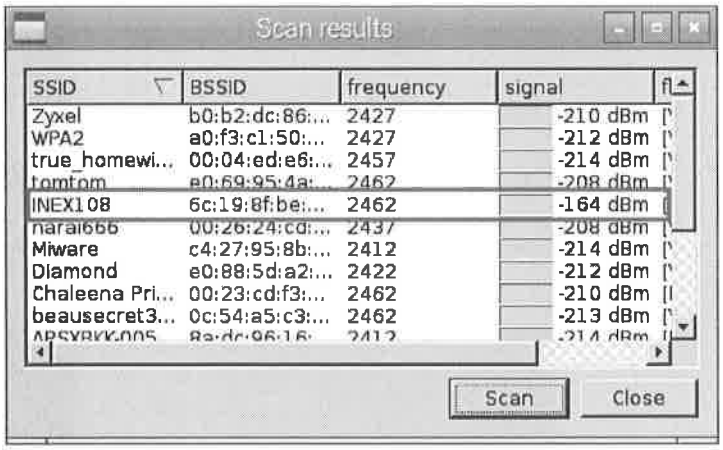
(3.2.3) เลือกที่หัวข้อ **Menu > Preferences > WiFi Configuration**



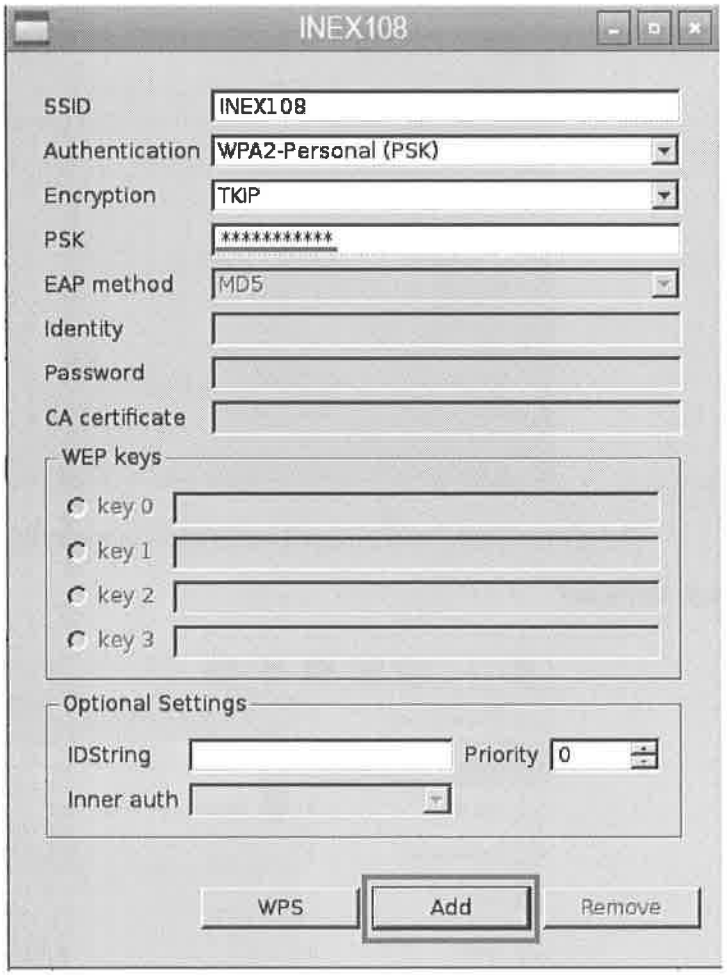
(3.2.4) หน้าต่างโปรแกรม **wpa_gui** ที่ใช้ในการตั้งค่าการเชื่อมต่อเครือข่ายไร้สายปรากฏขึ้นมา ที่ช่อง **Adapter :** จะแสดง **wlan0** เนื่องจากได้เชื่อมต่อกับ WiFi ดองเกิลแล้ว จากนั้นกดปุ่ม **Scan**



(3.2.5) หน้าต่างแจ้งผลการค้นหาเครือข่ายไร้สายปรากฏขึ้นมา ถ้าหากไม่พบ ให้กดปุ่ม **Scan** อีกครั้ง ทั้งนี้อาจเกิดจากช่องสัญญาณไม่ว่าง หรือเราเตอร์อาจยังไม่พร้อมทำงาน เมื่อค้นหาเครือข่ายพบ ทำการดับเบิ้ลคลิกเลือกเครือข่ายที่ต้องการเชื่อมต่อ ในที่นี้เลือกเครือข่าย **INEX108**



(3.2.6) จากนั้นหน้าต่างที่แจ้งรายละเอียดการการเชื่อมต่อกับเครือข่ายนี้ปรากฏขึ้นมา ที่ช่อง **PSK** จะเป็นรหัสรักษาความปลอดภัยเพื่อเข้าใช้งานเครือข่าย คลิกปุ่ม **Add** เพื่อเพิ่มเข้าไปในการเชื่อมต่อ

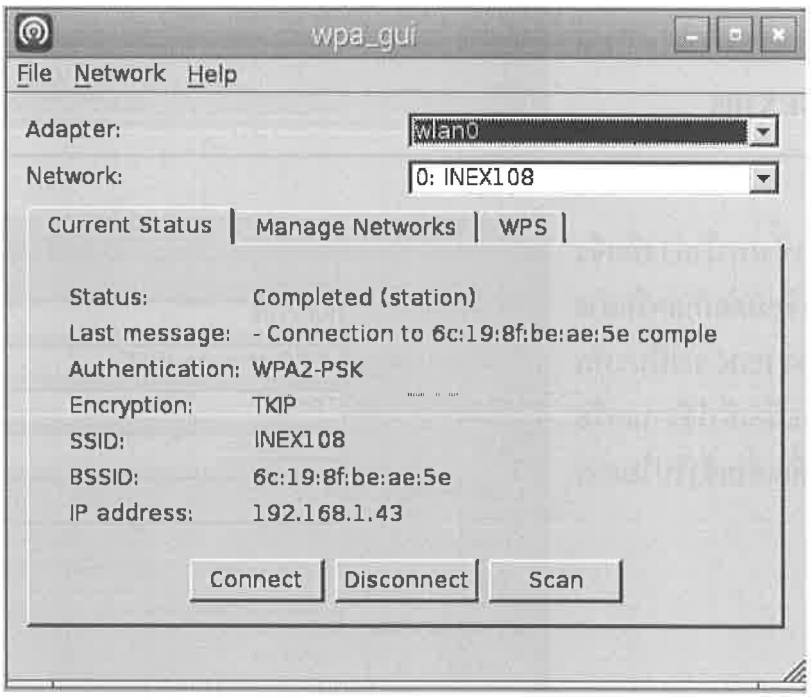


(3.2.7) กลับมายังหน้าต่างโปรแกรม **wpa_gui** หากทุกอย่างถูกต้อง จะแจ้งสถานะการเชื่อมต่อดังนี้

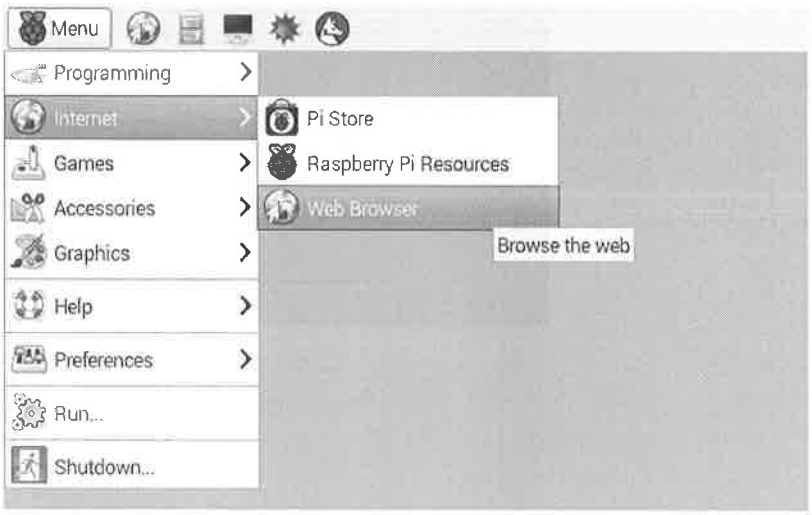
Status : Completed

ชื่อ SSID และ IP address ใช้ในการเชื่อมต่อในช่องทาง SSH

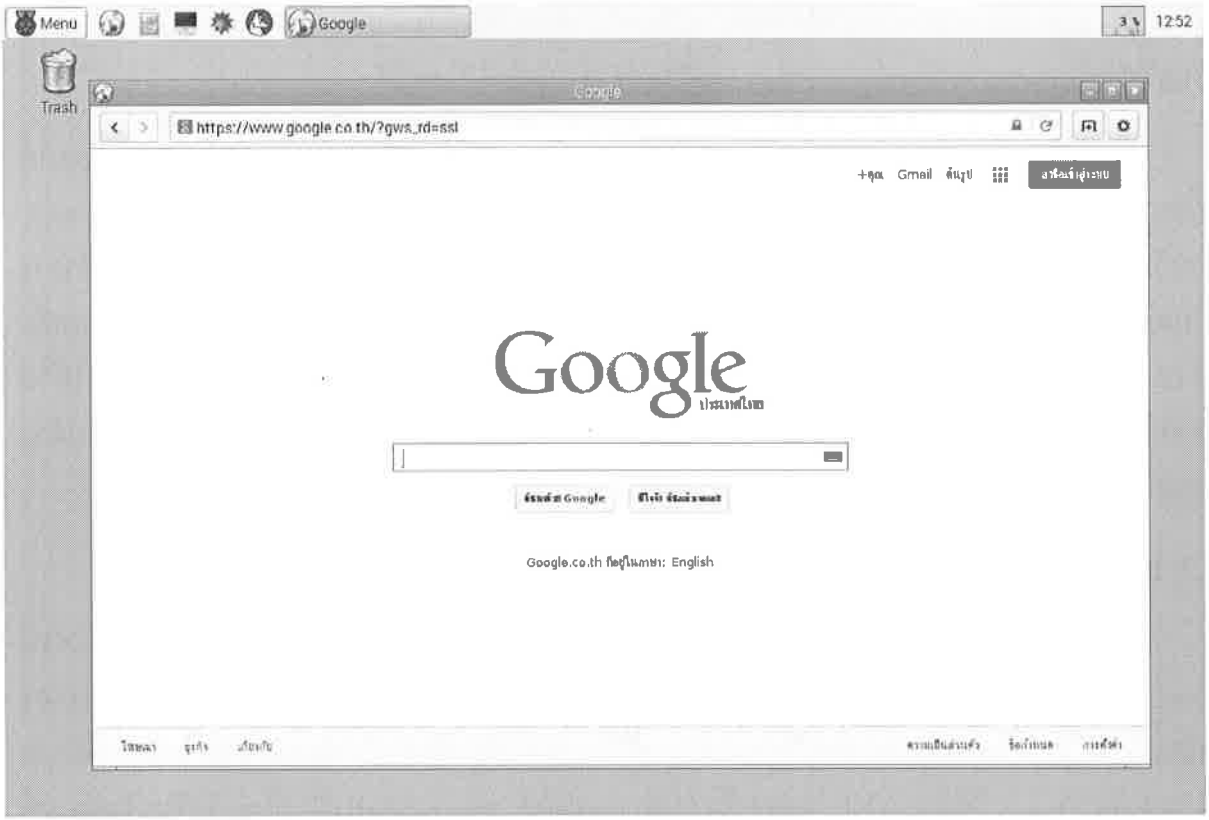
ระบบจะจดค่าของการเชื่อมต่อนี้ไว้ใช้ในการเชื่อมต่ออัตโนมัติครั้งต่อไป



(3.2.8) ทดสอบการเชื่อมต่อกับเครือข่ายอินเทอร์เน็ต โดยเข้าไปที่หัวข้อ **Menu > Internet > Web Browser**



(3.2.9) หน้าต่างของโปรแกรมเว็บบราวเซอร์ปรากฏขึ้นมา ทดลองป้อนตำแหน่งเว็บไซต์หรือ URL เป็น ***https://www.google.co.th*** ระบบจะเชื่อมต่อกับเว็บไซต์ **google** พร้อมใช้งานต่อไป



3.3 การควบคุม Raspberry Pi ผ่าน Secure Shell (SSH)

ถึงแม้ว่าบอร์ดคอมพิวเตอร์ Raspberry Pi ทุกรุ่นจะรองรับการเชื่อมต่อกับอุปกรณ์ USB อย่าง คีย์บอร์ด เมาส์ และจอแสดงผลได้อยู่แล้ว แต่ในการใช้งานจริงก็ไม่จำเป็นต้องต่อกับอุปกรณ์เหล่านี้เสมอไป เพื่อลดขนาดของของระบบลง แต่ยังคงทำงานหรือใช้งานได้

โดยปกติผู้ใช้งานบอร์ด Raspberry Pi มักใช้งานในแบบเดียวกับใช้คอมพิวเตอร์ นั่นคือ ต่อเมาส์ คีย์บอร์ด และจอภาพเข้ากับบอร์ด Raspberry Pi จากนั้นทำการเขียนโปรแกรมหรือทำงานตามที่ต้องการ สำหรับในบทความนี้ขอนำเสนอการใช้งาน Raspberry Pi ในอีกลักษณะหนึ่ง โดยใช้คอมพิวเตอร์ทำการรีโมตหรือเชื่อมต่อกับบอร์ด Raspberry Pi จากภายนอกผ่านระบบเครือข่าย แล้วทำการรันโปรแกรมหรือใช้งานบอร์ด Raspberry Pi ผ่านการควบคุมหรือกำหนดการใช้งานจากคอมพิวเตอร์อีกตัวหนึ่ง ทำให้ไม่ต้องใช้จอภาพ, เมาส์ และคีย์บอร์ดเพื่อต่อใช้งาน บอร์ด Raspberry Pi จึงทำงานได้อย่างคล่องตัว ไม่ต้องมีอุปกรณ์ต่อพ่วงมากมาย การติดต่อในลักษณะดังกล่าวนี้ใช้กระบวนการที่ชื่อว่า **SSH**

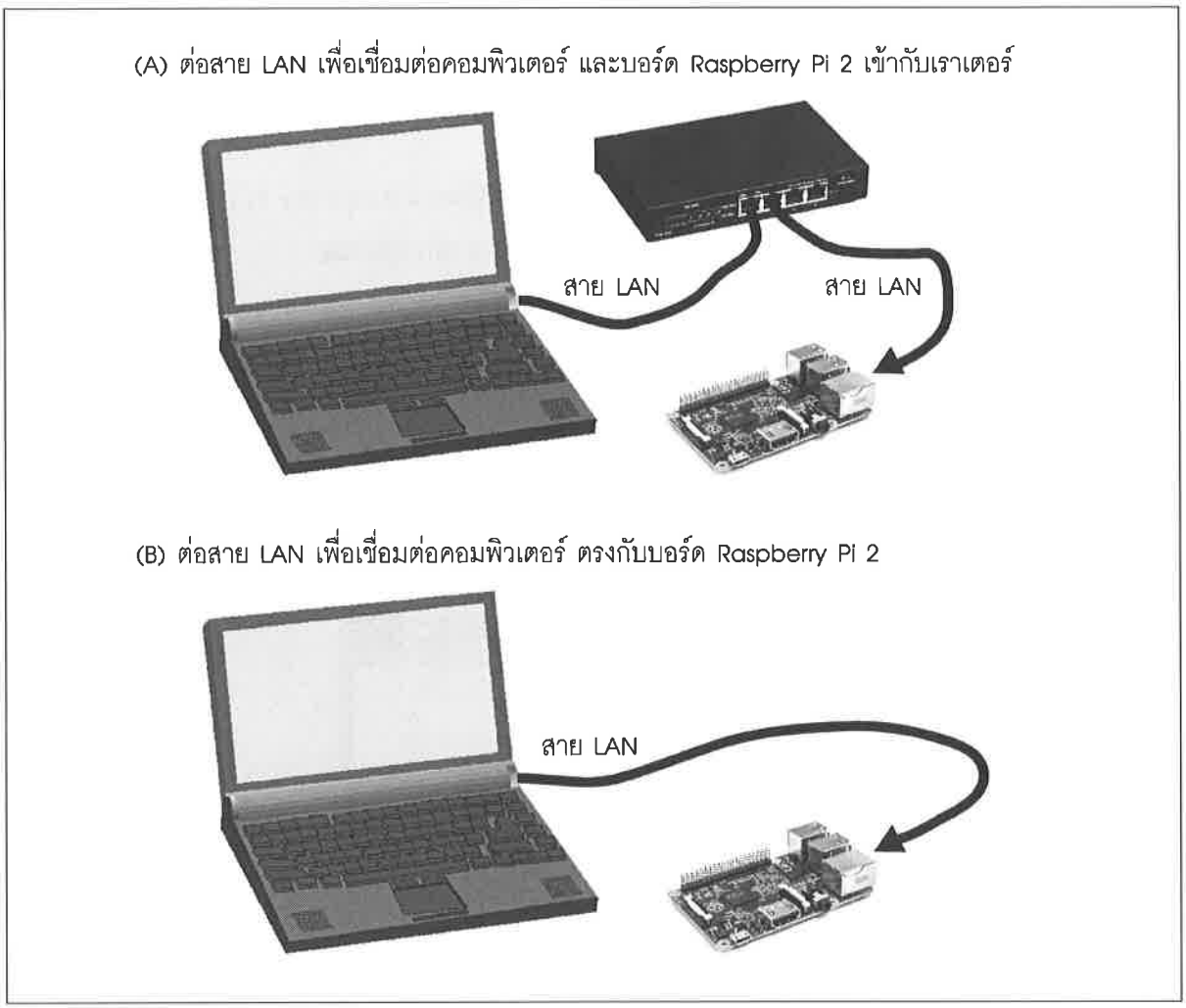
3.3.1 SSH คืออะไร

เดิมทีการควบคุมคอมพิวเตอร์ระยะไกลผ่านทางระบบเครือข่ายหรือเน็ตเวิร์ก มักใช้โปรแกรม telnet, rlogin หรือ rsh แต่มีปัญหาจากแฮกเกอร์ที่คอยดักจับข้อมูลในเน็ตเวิร์กระหว่างคอมพิวเตอร์ทั้งสองเครื่อง ทั้งนี้เป็นเพราะข้อมูลจากโปรแกรมที่กล่าวมาข้างต้นทั้งหมดไม่มีการเข้ารหัส จนกระทั่งในปี ค.ศ. 1955 **Tatu Ylonen** ได้พัฒนาโปรโตคอลชื่อ **SSH** ย่อมาจาก **Secure Shell** เป็นเน็ตเวิร์กโปรโตคอลใช้ควบคุมคอมพิวเตอร์ระยะไกลที่มีการเข้ารหัสด้วย public key กล่าวคือ SSH จะเข้ารหัสข้อมูล ส่งไปทางเน็ตเวิร์ก เมื่อถึงปลายทางก็จะถอดรหัสข้อมูลให้เอง ทำให้การดักจับข้อมูลทำได้ยากขึ้น

จึงอาจกล่าวได้ง่ายๆ ว่า SSH เป็นการใช้คอมพิวเตอร์สักเครื่องที่อยู่ในวงเครือข่ายหรือวง LAN เดียวกับบอร์ด Raspberry Pi ต่อเข้าไปควบคุมสั่งงานนั่นเอง ด้วยวิธีนี้ไม่จำเป็นต้องต่อบอร์ด Raspberry Pi กับจอแสดงผล, คีย์บอร์ด และเมาส์แต่อย่างใด เพราะสั่งงานจากคอมพิวเตอร์ที่เชื่อมต่อได้ทันที โดยมีเงื่อนไขว่า ทั้งคอมพิวเตอร์และบอร์ด Raspberry Pi จะต้องอยู่ในเครือข่ายเดียวกัน หรือในวง LAN เดียวกัน

3.3.2 การเชื่อมต่อทางฮาร์ดแวร์

ในการใช้งานบอร์ด Raspberry Pi ด้วยวิธีการ SSH หรือรีโมตเข้าไปทำงานนั้น ให้ต่อคอมพิวเตอร์ที่ใช้งานและบอร์ด Raspberry Pi เข้ากับเราเตอร์ผ่านทางจุดต่อ LAN ดังรูปที่ 3-3



รูปที่ 3-3 การเชื่อมต่อบอร์ด Raspberry Pi เพื่อใช้งานด้วยวิธีการ SSH ซึ่งทำได้ทั้งแบบต่อผ่านเราเตอร์ (รูป A) หรือต่อตรงในแบบ peer-to-peer (รูป B)

3.3.3 ติดต่อแบบ SSH ด้วยโปรแกรม Tera Term

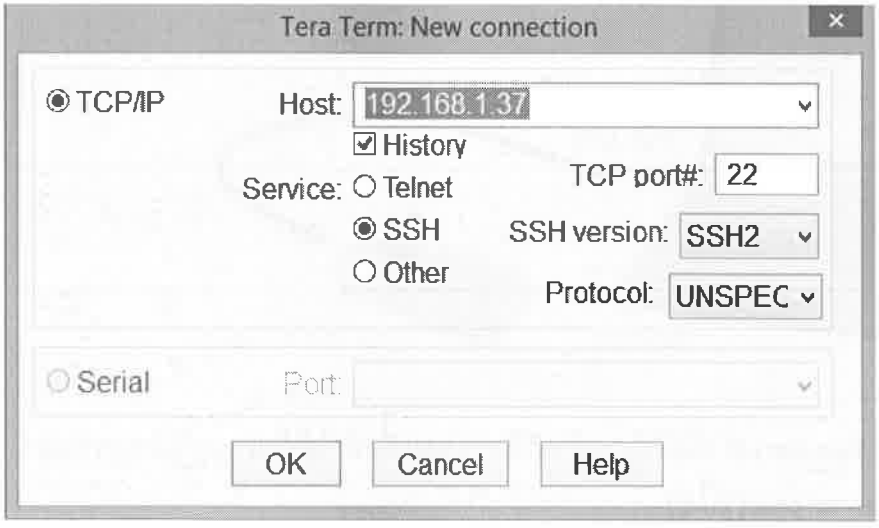
สำหรับโปรแกรมบนคอมพิวเตอร์ที่ใช้ควบคุมหรือติดต่อกับบอร์ด Raspberry Pi ผ่าน SSH นั้น หลายตัว ในที่นี้ใช้โปรแกรม **Tera Term** ดาวน์โหลดได้ฟรีจาก ***http://en.sourceforge.jp/projects/ttssh2/releases/***

ที่บอร์ด Raspberry Pi นั้น ระบบปฏิบัติการ Raspbian รุ่นใหม่ๆ ได้กำหนดค่าและเปิดใช้งาน SSH ไว้แล้ว จึงเชื่อมต่อผ่าน SSH ได้ทันที

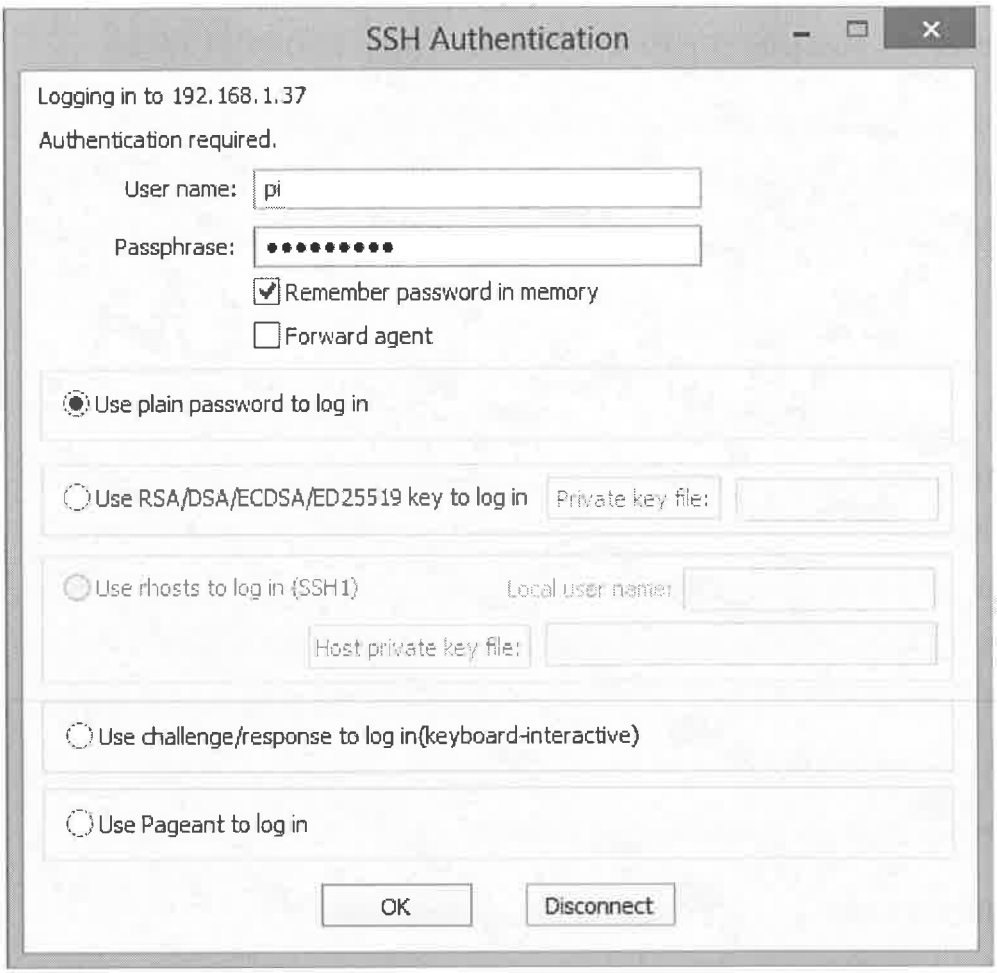
เมื่อเปิดโปรแกรม Tera Term ขึ้นมาให้ใส่ IP แอดเดรสของ Raspberry Pi (ตรวจสอบได้ด้วยคำสั่ง **`ifconfig`**) และเลือก **Service** เป็น **SSH** ดังรูปที่ 3-4 คลิกปุ่ม **OK**

เมื่อเชื่อมต่อกับ Raspberry Pi ได้แล้ว จะต้องทำการล็อกอินก่อน โดยใช้ Username และ Password เดียวกันกับที่ใช้ในตอนบูตระบบ จากนั้นคลิกปุ่ม OK เพื่อทำการเชื่อมต่อ ดังรูปที่ 3-5

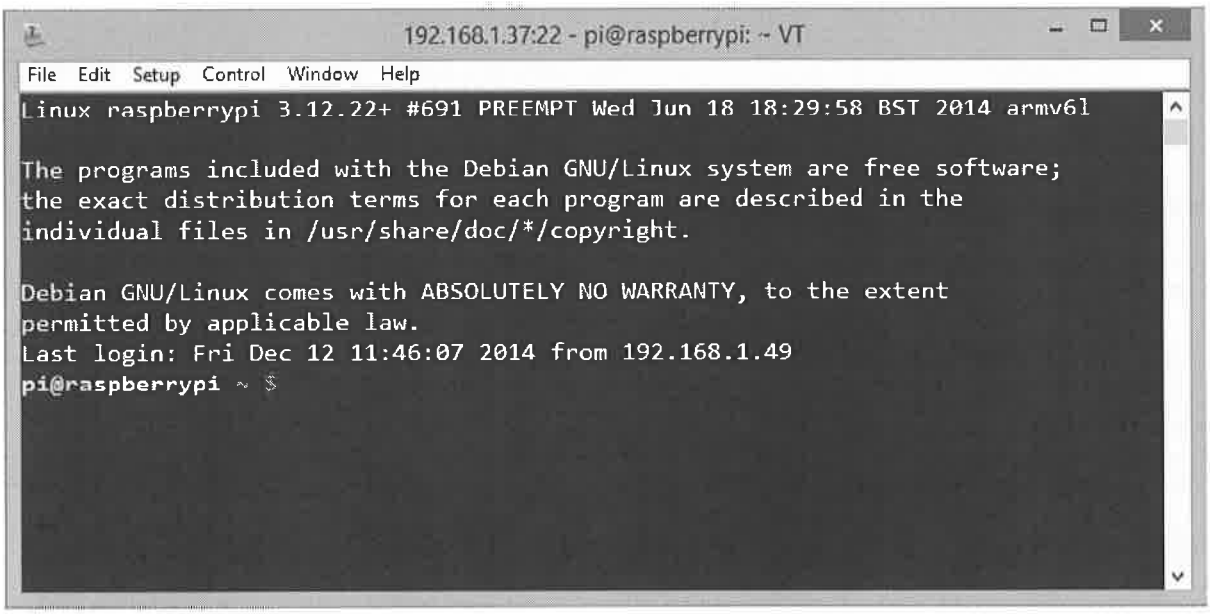
หลังจากนั้น หน้าต่าง VT มอนิเตอร์จะปรากฏขึ้นมา ดังรูปที่ 3-6 อันเป็นหน้าจอการทำงานในปัจจุบันของบอร์ด Raspberr Pi 2 นับจากนี้ผู้ใช้งานสามารถควบคุมหรือใช้งานบอร์ด Raspberry Pi ผ่านทางหน้าต่างนี้ได้แล้ว



รูปที่ 3-4 หน้าต่างของโปรแกรม Tera Term ที่ใช้ในการเชื่อมต่อเพื่อควบคุมการทำงานของบอร์ด Raspberry Pi ด้วยวิธีการ SSH



รูปที่ 3-5 ล็อกอินเข้าสู่ระบบ



รูปที่ 3-6 หน้าต่าง VT มอนิเตอร์ของโปรแกรม Tera Term ที่ใช้เป็นจอแสดงผลของบอร์ด Raspberry Pi หลังจากเชื่อมต่อกับคอมพิวเตอร์เพื่อแลกเปลี่ยนข้อมูลด้วยวิธีการ SSH

3.4 การควบคุมบอร์ด Raspberry Pi 2 จากระยะไกลด้วยคอมพิวเตอร์

ปกติแล้วการใช้งานบอร์ด Raspberry Pi จะเน้นไปที่การใช้งานแบบโดยลำพังเหมือนกับคอมพิวเตอร์เครื่องหนึ่งที่ต่อคีย์บอร์ด, เมาส์, จอภาพ และแหล่งจ่ายไฟ หากต้องการเชื่อมต่อกับเครือข่ายอินเทอร์เน็ต ก็ต่อสาย LAN เข้าที่จุดต่อพอร์ตอีเธอร์เน็ตหรือใช้ USB WiFi ดองเกิลต่อเข้าที่พอร์ต USB ก็ได้

ยังมีการใช้งานบอร์ด Raspberry Pi ในอีกรูปแบบหนึ่ง โดยไม่ต้องต่อเมาส์, คีย์บอร์ด และจอภาพ เพียงต่อกับคอมพิวเตอร์อีกตัวหนึ่งด้วยสาย LAN ผ่านจุดต่อพอร์ตอีเธอร์เน็ต เรียกวิธีการในลักษณะนี้ว่า **การเชื่อมต่อระยะไกล** หรือ **Remote Desktop Connection (RDC)** หรือเรียกสั้นๆ ว่า **การรีโมต (remote)** ด้วยวิธีการนี้จะทำให้ผู้ใช้งานสามารถสั่งงานหรือใช้งานบอร์ด Raspberry Pi ผ่านทางหน้าจอของคอมพิวเตอร์ได้ จึงไม่ต้องต่อเมาส์, คีย์บอร์ด และจอภาพให้กับบอร์ด Raspberry Pi แต่ประการใด เพราะจะใช้อุปกรณ์เหล่านั้นจากคอมพิวเตอร์ที่ทำการเชื่อมต่อ จึงทำให้ผู้ใช้งานสามารถสร้างระบบควบคุมที่ติดต่อหรือตรวจสอบการทำงานกับคอมพิวเตอร์ที่ใช้ควบคุมในระยะไกลได้

ขั้นตอนในการรีโมตบอร์ด Raspberry Pi ด้วยคอมพิวเตอร์อีกเครื่องหนึ่งทำได้ไม่ยาก โดยใช้โปรแกรมที่ชื่อว่า **xrdp (x-windows remote desktop protocol)**

3.4.1 ติดตั้งโปรแกรม xrdp

มีขั้นตอนโดยสรุปดังนี้

(1) เชื่อมต่อเมาส์, คีย์บอร์ด และจอภาพให้กับ Raspberry Pi 2

(2) เชื่อมต่อกับบอร์ด Rapberry Pi 2 กับเครือข่ายอินเทอร์เน็ต

(3) เปิดหน้าต่างเทอร์มินอลในโหมดคอมมานด์ไลน์หรือเข้าสู่โหมดกราฟิกด้วยการรันคำสั่ง **`startx`** ก็ได้ รอจนกระทั่งระบบพร้อมทำงาน แล้วเปิดหน้าต่าง **LX Terminal**

(4) พิมพ์คำสั่งต่อไปนี้เพื่อทำการดาวน์โหลดและติดตั้งโปรแกรม

```
sudo apt-get install xrdp
```

3.4.2 ติดตั้งและใช้งานโปรแกรม Advanced IP Scanner

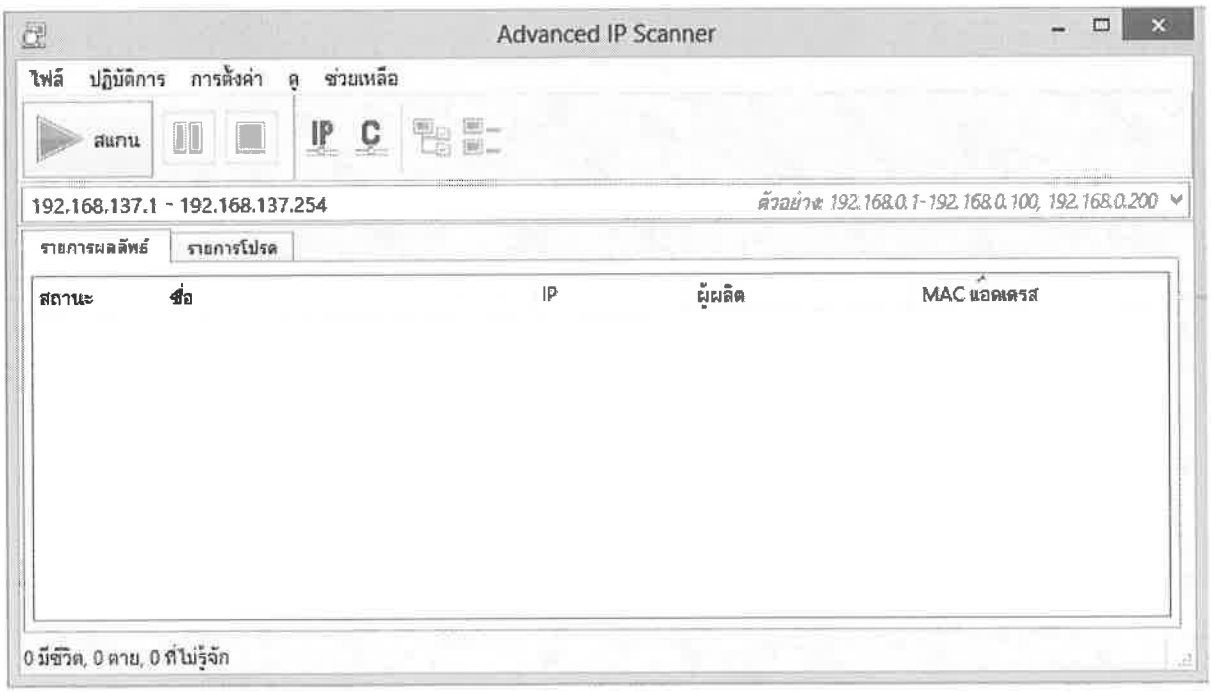
เนื่องจากการเชื่อมต่อแบบ RDC ต้องใช้ช่องทางผ่านพอร์ตอีเธอร์เน็ต สิ่งหนึ่งที่ต้องทราบคือ ***หมายเลข IP แอดเดรสของบอร์ด Raspberry Pi 2 ที่นำมาเชื่อมต่อ*** ซึ่งหมายเลข IP แอดเดรสนี้คอมพิวเตอร์ที่เชื่อมต่อด้วยจะเป็นตัวกำหนดให้ การตรวจสอบหมายเลข IP แอดเดรส จะต้องใช้เครื่องมือช่วย ในที่นี้ขอแนะนำโปรแกรม **Advanced IP Scanner**

Advanced IP Scanner เป็นซอฟต์แวร์ที่ช่วยให้ผู้ใช้ค้นหาและตรวจสอบหมายเลข IP แอดเดรสของอุปกรณ์ที่เชื่อมต่ออยู่บนเครือข่ายเดียวกัน โดยซอฟท์แวร์นี้รองรับการติดต่อกับโปรโตคอลแบบต่างๆ มากมาย อาทิ HTTP, HTTPS, FTP รวมถึงการค้นหาเครือข่ายเพื่อรับทราบข้อมูลเพิ่มเติมเกี่ยวกับอุปกรณ์ที่เชื่อมต่อกัน ได้แก่ ชื่อของคอมพิวเตอร์หรืออุปกรณ์ และ MAC แอดเดรส ทำการดาวน์โหลดโปรแกรมมาใช้งานได้โดยไม่มีค่าใช้จ่ายจาก ***http://www.advanced-ip-scanner.com*** เมื่อดาวน์โหลดมาแล้ว ติดตั้งให้เรียบร้อย ซึ่งมีขั้นตอนเหมือนกับการติดตั้งโปรแกรมประยุกต์อื่นๆ บนระบบปฏิบัติการวินโดวส์

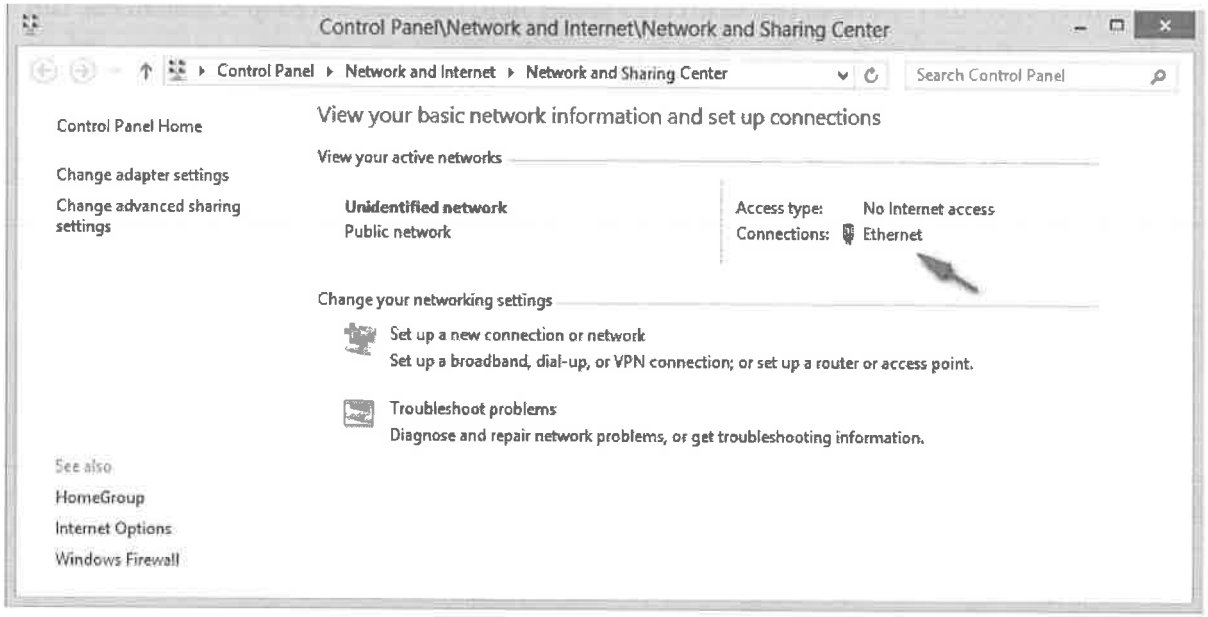
(1) เชื่อมต่อและค้นหาขอบเขตของเลข IP แอดเดรส

(1.1) เริ่มต้นการค้นหา IP แอดเดรสของบอร์ด Raspberry Pi 2 ด้วยการต่อสาย LAN ระหว่างพอร์ตอีเธอร์เน็ตของ Raspberry Pi 2 กับคอมพิวเตอร์

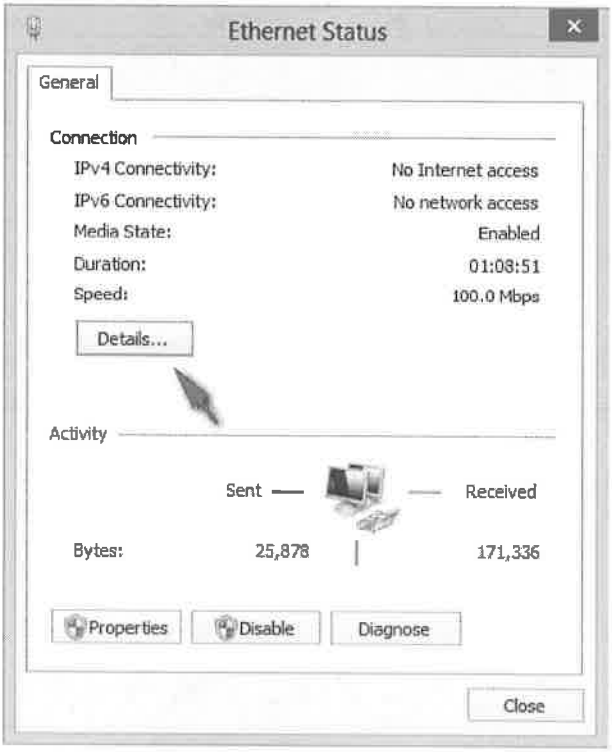
(1.2) เปิดโปรแกรม Advanced IP Scanner ขึ้นมา จะมีรายละเอียดดังรูป



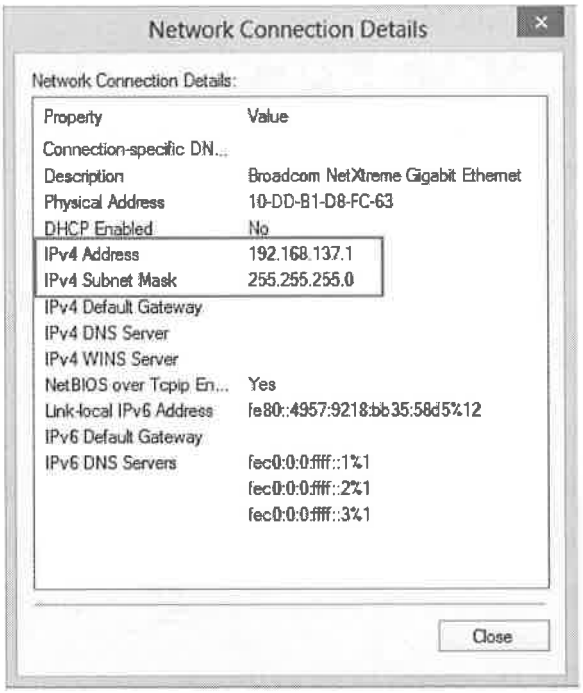
(1.3) การตรวจหาเลข IP แอดเดรสนั้น จะต้องทราบขอบเขตของจำนวน IP แอดเดรสก่อน ดังนั้นอุปกรณ์ที่ต้องการทราบเลข IP แอดเดรสจึงต้องอยู่ในวง LAN เดียวกัน จึงจะตรวจหาหรือสแกนหาเลข IP แอดเดรสได้ เปิด **Control Panel** แล้วไปที่ **Network and Internet >> Network and Sharing Center** จากนั้นคลิกที่ **Ethernet** ดังรูป



(1.4) หน้าต่าง **Ethernet Status** ที่แสดงสถานะของ Ethernet จะปรากฏขึ้น คลิกที่ **Details...**



(1.5) เมื่อคลิกแล้ว หน้าต่าง **Network Connection Details** ปรากฏขึ้นมา แสดงรายละเอียดของการเชื่อมต่อดังรูป



ข้อมูลที่สำคัญสำหรับการหาขอบเขต IP แอดเดรสคือ **IPv4 Address** และ **IPv4 Subnet Mask** โดยที่

IPv4 Address *คือ หมายเลข IP แอดเดรสของคอมพิวเตอร์ที่ทำหน้าที่จ่ายหรือกำหนดเลข IP แอดเดรสให้แก่อุปกรณ์ที่ต่ออยู่ในวง LAN เดียวกัน*

IPv4 Subnet Mask *คือ จำนวนหมายเลข IP แอดเดรสของอุปกรณ์ที่เป็นไปได้ภายในวง LAN เดียวกัน*

(2) การคำนวนหาขอบเขตของเลข IP แอดเดรส

(2.1) <u>การคำนวนหาหมายเลขเริ่มต้นของ IP แอดเดรสหรือ Network IP</u>

เลข IP แอดเดรส มีข้อมูล 4 ชุด ชุดละ 8 บิต และจะคั่นด้วย . (จุด) หากมีการกระทำทางด้านลอจิกจะต้องกระทำให้ตรงชุดด้วย จากขั้นตอนที่ (1.5) ในหัวข้อ 3.4.2 ได้หมายเลขของ **IPv4 Address = 192.168.137.1** เมื่อแปลงเป็นเลขฐานสอง จะได้ **11000000.10101000.10001001.00000001**

ส่วน **IPv4 Subnet Mask = 255.255.255.0** เมื่อทำการแปลงเป็นเลขฐานสอง จะได้เป็น **11111111.11111111.11111111.00000000**

นำค่าของ IPv4 Address มาแอนด์กับ IPv4 Subnet Mask จะได้ผลลัพธ์เป็น

11000000.10101000.10001001.00000000

แปลงเป็นเลขฐานสิบจะได้

192.168.137.0

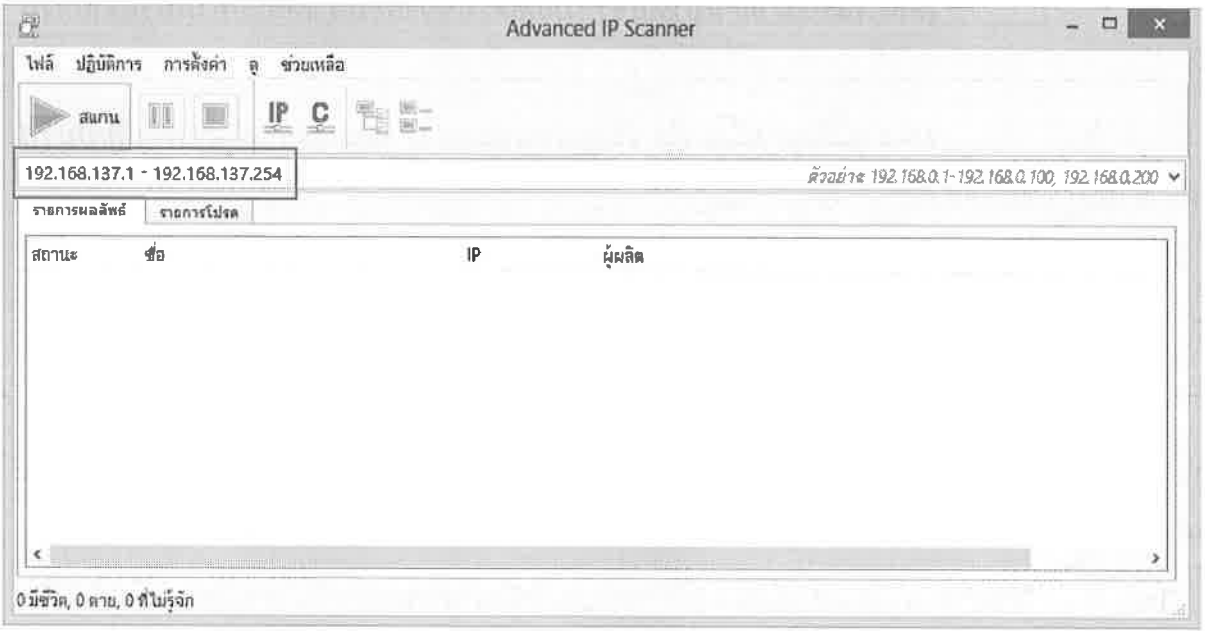
จึงได้ค่าเริ่มต้นของ IP แอดเดรสเท่ากับ 192.168.137.0

(2.2) การคำนวนหาหมายเลขสิ้นสุดของ IP แอดเดรสหรือ Broadcast IP

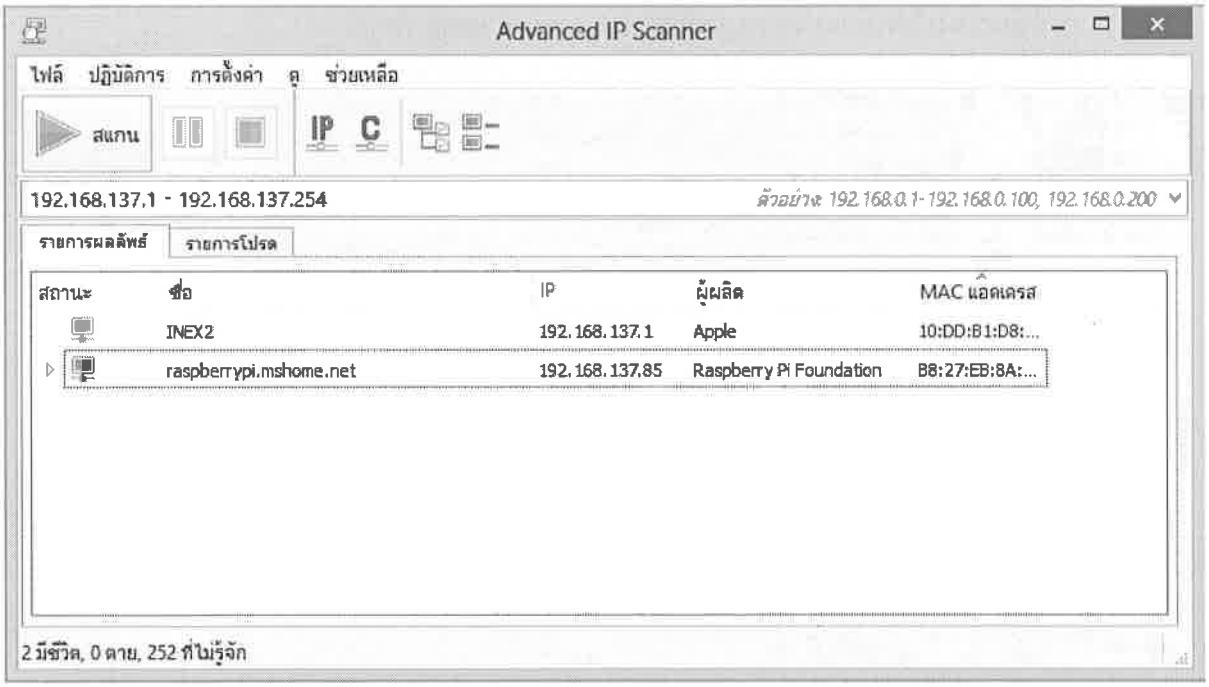
วิธีที่ง่ายที่สุดคือ เมื่อทราบแล้วว่า ในแต่ละชุดข้อมูล มี 8 บิต มีค่า 0 ถึง 255 ให้พิจารณาจาก Subnet Mask ก่อนจะเห็นได้ว่าชุดที่ 4 ของ Subnet Mask เป็นเลข 0 นอกนั้นมีค่าเป็นหมายเลข 255 ทั้ง 3 ชุด ดังนั้นจำนวนของหมายเลข Subnet Mask ที่จะทำให้จากเดิมคือ 255.255.255.0 ไปถึง 255.255.255.255 มีหมายเลขที่เป็นไปได้ 256 จำนวนหรือ 0 ถึง 255 ส่งผลให้หมายเลข IP แอดเดรสที่อยู่ในวง LAN นี้เริ่มจาก Network IP คือ **192.168.137.0** ไปถึง**192.168.137.255** ก็จะได้ทั้งหมด 256 หมายเลข *ดังนั้นขอบเขตหมายเลข IP แอดเดรสคือ 192.168.137.0 ถึง 192.168.137.255*

(3) เริ่มต้นการตรวจหา IP แอดเดรสของอุปกรณ์

(3.1) เมื่อได้ขอบเขตที่ต้องการหาหมายเลข IP แอดเดรสแล้ว ให้นำหมายเลขนั้นใส่ในช่องหมายเลขแอดเดรส ดังรูป จากนั้นคลิกที่ปุ่มสแกน เพื่อทำการตรวจหา



(3.2) เมื่อสแกนหรือตรวจหาเสร็จแล้ว โปรแกรมจะแสดงหมายเลข IP แอดเดรสของอุปกรณ์ทุกตัวที่ต่ออยู่ในวง LAN เดียวกันดังรูป



ด้วยขั้นตอนที่นำเสนอ ผู้ใช้งาน Raspberry Pi จะทราบว่า หมายเลข IP แอดเดรสของ Raspberry Pi 2 เป็นหมายเลขอะไร ซึ่งอาจจะดูจากชื่อหรือผู้ผลิตก็ได้ โดยนำหมายเลข IP แอดเดรสไปทำการรีโมตหรือเรียกอีกอย่างคือการควบคุมคอมพิวเตอร์ระยะไกลได้

3.4.3 ทดสอบใช้งาน

หลังจากติดตั้งโปรแกรม xrdp และ Advanced IP Scanner แล้ว ให้ทำการตรวจสอบหาเลข IP แอดเดรส ของบอร์ด Raspberry Pi 2 เมื่อได้เลขมาแล้ว จะเข้าสู่การทดสอบใช้งาน

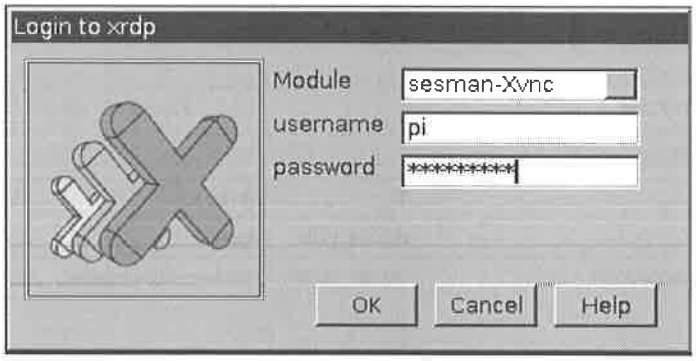
(1) ปลดเมาส์, คีย์บอร์ด และจอภาพออกจากบอร์ด Raspberry Pi

(2) ต่อสาย LAN ระหว่างพอร์ตอีเธอร์เน็ตของ Raspberry Pi กับคอมพิวเตอร์

(3) เปิดโปรแกรม Remote Desktop Connection ดังรูป

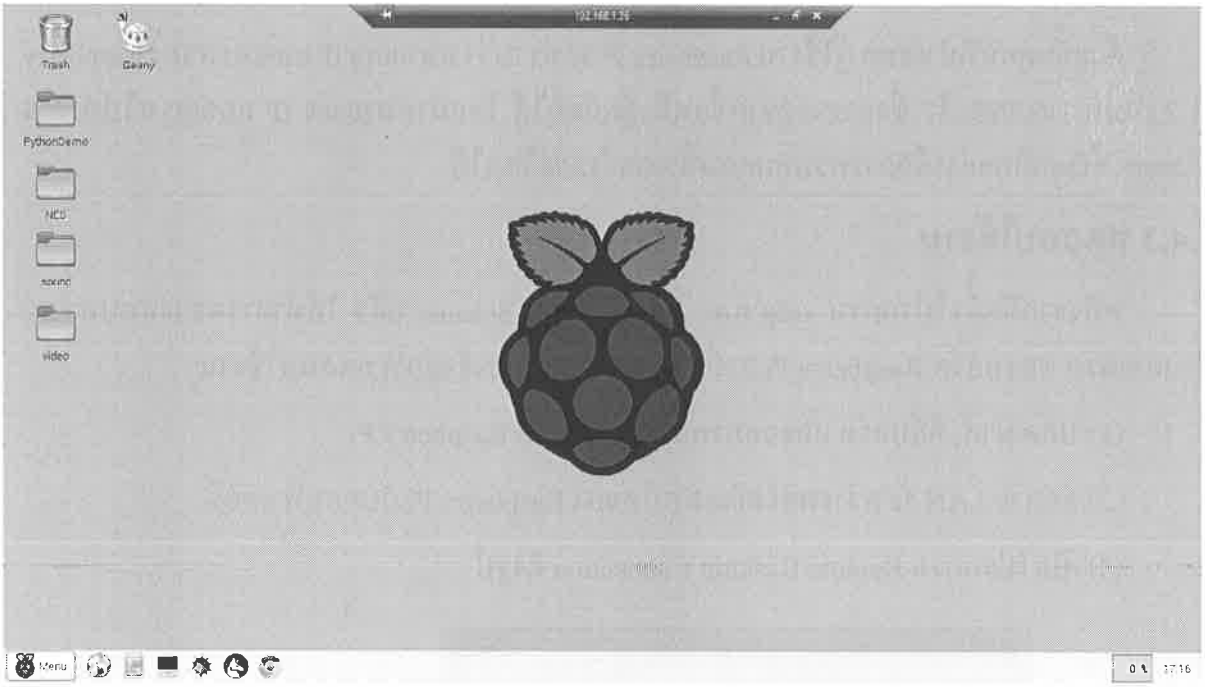
(4) ใส่ IP แอดเดรสของบอร์ด Raspberry Pi 2 (ได้มาจากการตรวจหาด้วยโปรแกรม Advanced IP Scanner) ที่ช่อง **Computer:** แล้วคลิกปุ่ม **Connect** เพื่อทำการเชื่อมต่อ

เมื่อเชื่อมได้แล้วจะปรากฏหน้าต่าง **Login to xrdp** ดังรูป



(5) ใส่ **username** เป็น **pi** และ **password** เป็น **raspberry** *(เป็น username และ password มาตรฐาน - ให้ป้อนรหัสผ่านและชื่อผู้ใช้งานหากมีการกำหนดให้แตกต่างออกไป)* แล้วคลิกปุ่ม **OK**

(6) จะปรากฏหน้า Desktop ของ Raspberry Pi ขึ้นมา



ด้วยขั้นตอนทั้งหมดจะช่วยให้ใช้งานบอร์ด Raspberry Pi (ทั้ง B, B+ และ 2) ผ่านคอมพิวเตอร์อีกเครื่องหนึ่งที่ต่อห่างออกไปได้

บทที่ 4
คำสั่งพื้นฐานบนเทอร์มินอลของระบบปฏิบัติการ Linux

ระบบปฏิบัติการหลักที่ติดตั้งลงในบอร์ดคอมพิวเตอร์ Raspberry Pi ในทุกรุ่นคือ Raspbian ซึ่งเป็นระบบปฏิบัติการ Linux แบบหนึ่งที่พัฒนาขึ้นมาเพื่อรองรับบอร์ดคอมพิวเตอร์ขนาดเล็กอย่าง Raspberry Pi โดยเฉพาะ **ในการใช้งานระบบปฏิบัติการ Linux มี 2 โหมดคือ** *โหมดคอมมานด์ไลน์* (คล้ายกับระบบปฏิบัติการ DOS) และ *โหมดกราฟิก* (คล้ายระบบปฏิบัติการวินโดวส์หรือ MAC OS) โดยส่วนใหญ่จะนิยมทำงานกันในโหมดคอมมานด์ไลน์เนื่องจากเร็วและไม่ใช้ทรัพยากรมาก อย่างไรก็ตาม การทำงานในโหมดกราฟิกแท้ที่จริงก็มีคำสั่งและการประมวลผลแบบคอมมานด์ไลน์จำนวนมากอยู่เบื้องหลัง ดังนั้นการทำความรู้จักคำสั่งพื้นฐานบนเทอร์มินอลของระบบปฏิบัติการ Linux จึงเป็นเรื่องที่จำเป็น

รูปที่ 4-1 หน้าจอเมื่อบอร์ด Raspberry Pi 2 ทำการบูต, เตรียมความพร้อมและตรวจสอบทรัพยากรทั้งหมด

4.1 เทอร์มินอลคืออะไร

โดยปกติเมื่อบอร์ด Raspberry Pi เริ่มทำงาน มันจะทำการเตรียมความพร้อมและตรวจสอบทรัพยากรที่เกี่ยวข้องทั้งหมด ซึ่งหากมีการต่อจอแสดงผลไว้ จะเห็นกระบวนการต่างๆ แสดงออกมาเรียงกันเป็นบรรทัด ดังรูปที่ 4-1 ซึ่งขั้นตอนนี้มีในคอมพิวเตอร์ทุกระบบปฏิบัติการ เพียงแต่ว่า ผู้ผลิตแต่ละรายทำการซ่อนหน้าจอในส่วนนี้ ที่ผู้ใช้งานเห็นในคอมพิวเตอร์ส่วนใหญ่อาจเป็นจอว่างๆ หรือเป็นหน้าจอแสดงโลโก้ของระบบปฏิบัติการนั้นๆ จนกระทั่งการเตรียมความพร้อมเสร็จสิ้นก็จะเข้าสู่หน้าจอของการเริ่มต้นการทำงาน

สำหรับระบบปฏิบัติการ Raspbian หากบอร์ด Raspberry Pi ไม่ได้ถูกตั้งให้บูตเข้าสู่โหมดกราฟิกโดยอัตโนมัติ เมื่อการบูตเสร็จสิ้น มันจะแสดงหน้าจอในโหมดคอมมานด์ไลน์และเชลพร้อมพ์ **pi@raspberrypi $** เพื่อรอรับการป้อนคำสั่ง ดังรูปที่ 4-2 หน้าจอในลักษณะนี้เรียกว่า **เทอร์มินอล (Terminal)** สำหรับผู้ที่เคยใช้งาน DOS จะคุ้นเคย แต่สำหรับผู้ใช้งานที่เคยใช้งานเฉพาะวินโดวส์หรือ MAC OS อาจไม่รู้จัก

เมื่อมาถึงตรงนี้ บอร์ด Raspberry Pi พร้อมทำงาน เมื่อมีการป้อนคำสั่งซึ่งจะได้นำเสนอข้อมูลโดยสรุปหัวข้อต่อไป

```
raspberrypi login: pi
Password:
Last login: Mon Feb 23 19:37:59 GMT-7 2015 from 192.168.1.50 on pts/2
Linux raspberrypi 3.18.7-v7+ #755 SMP PREEMPT Thu Feb 12 17:20:48 GMT 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi ~ $
```

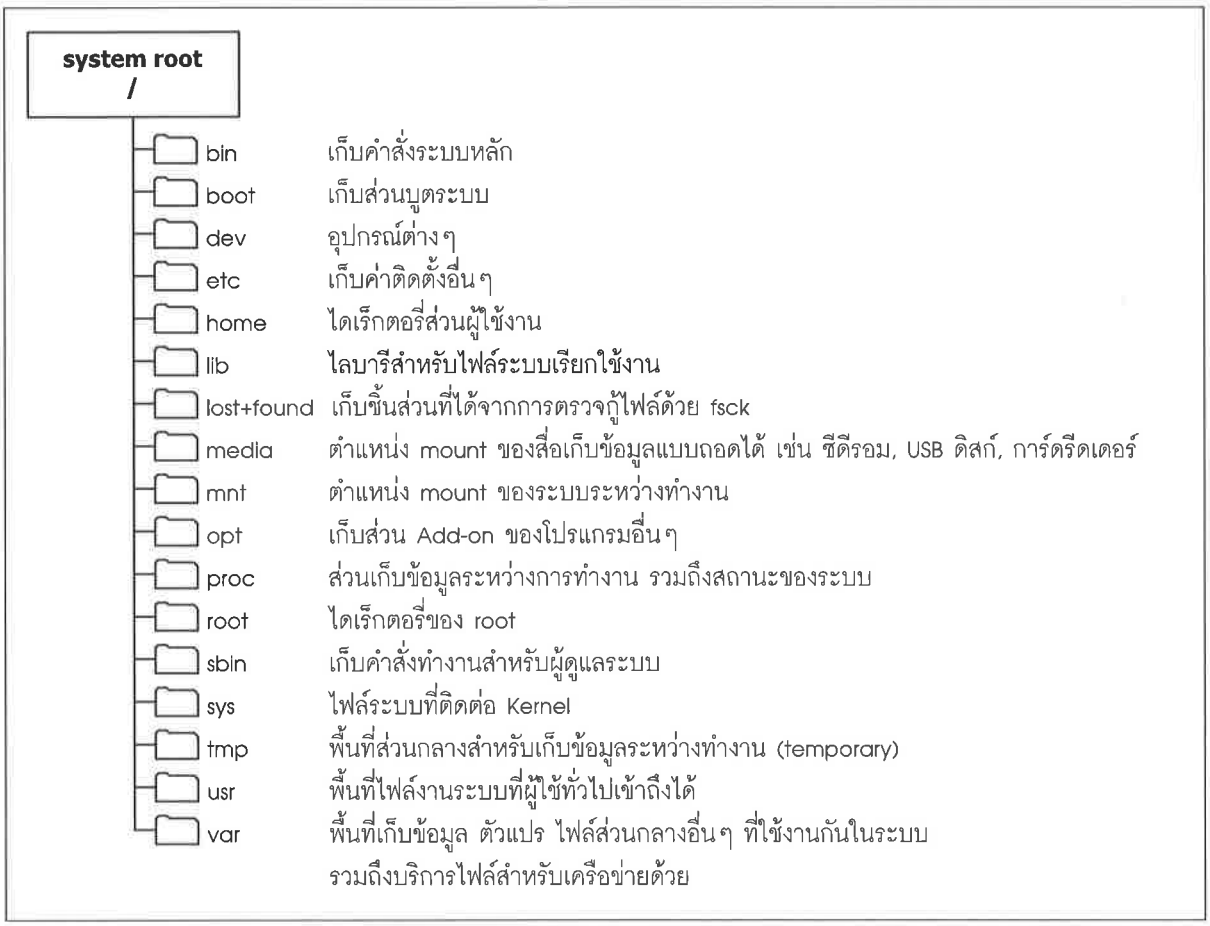
รูปที่ 4-2 เมื่อบูตระบบเสร็จสิ้น และป้อนรหัสผ่านเพื่อเข้าสู่การทำงาน หน้าจอของการทำงานจะเข้าสู่โหมดคอมมานด์ไลน์ พร้อมรับคำสั่งเพื่อทำงานต่อไป

4.2 โครงสร้างไดเร็กตอรี่หลักของระบบปฏิบัติการ Linux

เพื่อให้การเรียนรู้เกี่ยวกับระบบปฏิบัติการ Raspbian ที่ติดตั้งลงบนบอร์ด Raspberry Pi 2 เป็นไปอย่างมีประสิทธิภาพ ในหัวข้อต่อจากนี้จะได้อธิบายถึงภาพรวมของโครงสร้างการจัดการไฟล์และไดเร็กตอรี่ของระบบปฏิบัติการ Linux เพื่อให้เกิดความเข้าใจที่ตรงกัน

เหตุผลที่ต้องทำความเข้าใจในเรื่องนี้เนื่องจากระบบปฏิบัติการ Raspbian เป็นระบบปฏิบัติการที่ได้รับการสืบทอดมาจาก Linux โดยมีการปรับปรุงแก้ไขไฟล์ระบบบางส่วนเพื่อให้ใช้ทรัพยากรลดลง ทำให้ติดตั้งลงบนบอร์ด Raspberry Pi 2 ได้ ดังนั้นการทำความเข้าใจในโครงสร้างไดเร็กตอรี่ของระบบปฏิบัติการ Linux จึงเป็นการทำความเข้าใจโครงสร้างไดเร็กตอรี่ของระบบปฏิบัติการ Raspbian ไปด้วย

โครงสร้างไดเร็กตอรี่หลักของระบบปฏิบัติการ Linux แสดงด้วยแผนภาพในรูปที่ 4-3 ผู้เรียนรู้และพัฒนาโปรแกรมบน Raspberry Pi 2 จะได้ทราบถึงตำแหน่ง, เส้นทาง (พาธ) ของไดเร็กตอรี่และไฟล์ เพื่อให้เข้าถึงและเรียกใช้ง่ายขึ้น เนื่องจากโดยส่วนใหญ่การพัฒนาโปรแกรมบน Raspberry Pi 2 มักกระทำในโหมดคอมมานด์ไลน์ผ่านหน้าต่างเทอร์มินอล หากไม่ทราบถึงโครงสร้างไดเร็กตอรี่ อาจทำให้ต้องใช้เวลาในการค้นหาไดเร็กตอรี่ที่จัดเก็บไฟล์ หรืออาจหาไม่พบได้ แผนภาพนี้จะช่วยลดปัญหาดังกล่าวลง



รูปที่ 4-3 โครงสร้างไดเร็กตอรี่หลักของระบบปฏิบัติการ Linux

4.3 การใช้งานคำสั่งบนเทอร์มินอลของระบบปฏิบัติการ Rasbian

เมื่อบูตระบบจนเสร็จเรียบร้อย หน้าจอแรกสุดบนเทอร์มินอลคือ รอล็อกอินด้วยชื่อผู้ใช้ pi รหัสผ่านเป็น raspberry (จะไม่ปรากฏให้เห็นขณะพิมพ์ป้อนรหัส) ขั้นตอนนี้อาจข้ามได้ หากมีการตั้งให้ทำการล็อกอินแบบอัตโนมัติ

```
login: pi
Password: raspberry
Last login: Thu Aug 27 14:38:30 ICT 2015 from service.mshome.net on pts/0
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi ~ $ _
```

เมื่อเข้าสู่เชลพร้อมพ์สำหรับป้อนคำสั่ง จะแสดงดังนี้

```
pi@raspberrypi ~ $ _
```

มีความหมายว่า ชื่อผู้ใช้ pi ทำงานบนโฮสต์เนม (hostname) ชื่อ raspberry

~ คือ home directory หรือไดเร็กตอรี่ตั้งต้นของผู้ใช้ ซึ่งอยู่ที่ /home/pi

ทดลองเปลี่ยนไปที่ไดเร็กตอรี่ **root** / โดยการใช้คำสั่ง **cd /**

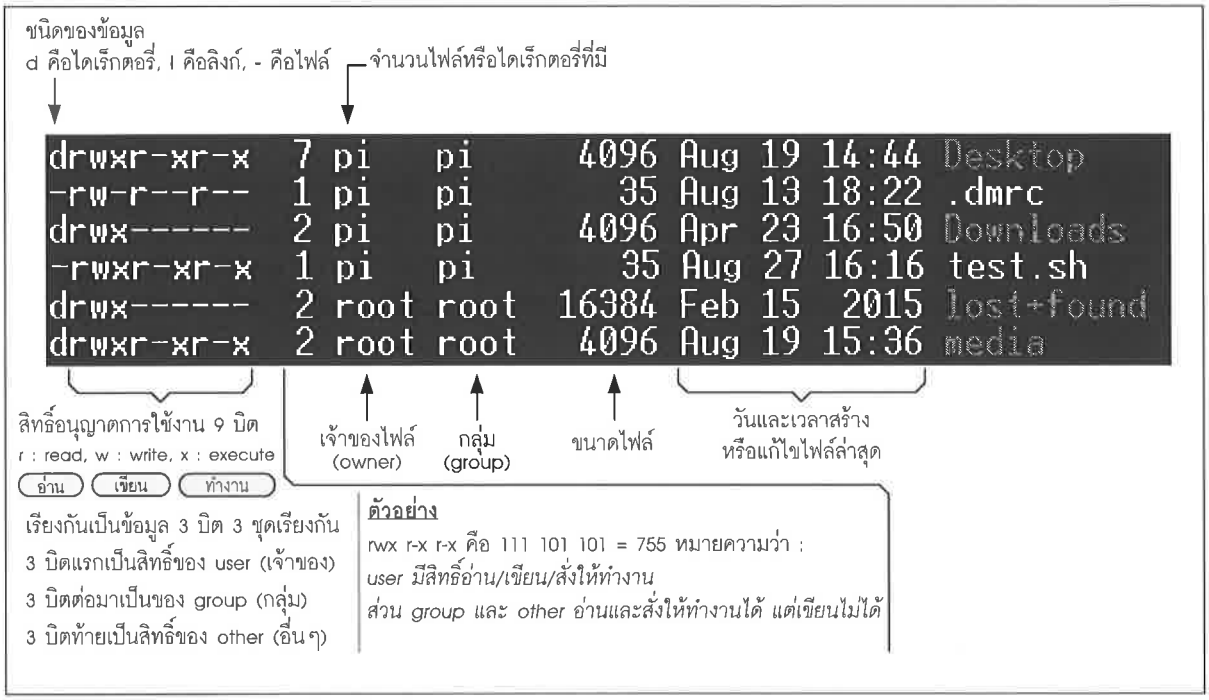
```
pi@raspberrypi ~ $ cd /
pi@raspberrypi / $ _
```

ถ้าต้องการดูว่ามีอะไรอยู่ในไดเร็กตอรี่ที่อยู่ ณ ขณะนี้บ้าง ให้พิมพ์คำสั่ง **ls** เพื่อแสดงรายการไฟล์แบบสรุป แต่ถ้าต้องการดูรายละเอียดรวมถึงรายการที่ซ่อนอยู่ ใช้คำสั่ง **ls -al** จะแสดงผลต่างกัน ดังรูปที่ 4-4

รายละเอียดที่แสดงนั้นประกอบด้วยข้อมูลหลายส่วน ดังแสดงรายละเอียดโดยสรุปไว้ในรูปที่ 4-5 นี้

```
pi@raspberrypi / $ ls
bin   boot.bak  etc   lib         media  opt   root  sbin    srv  tmp  var
boot  dev       home  lost+found  mnt    proc  run   selinux sys  usr

pi@raspberrypi / $ ls -al
total 100
drwxr-xr-x  23 root root  4096 Aug 24 09:52 .
drwxr-xr-x  23 root root  4096 Aug 24 09:52 ..
drwxr-xr-x   2 root root  4096 Aug 13 11:48 bin
drwxr-xr-x   6 root root 16384 Jan  1  1970 boot
drwxr-xr-x   3 root root  4096 Apr 23 15:26 boot.bak
drwxr-xr-x  12 root root  3220 Aug 26 16:23 dev
drwxr-xr-x 113 root root  4096 Aug 27 15:51 etc
drwxr-xr-x   3 root root  4096 Feb 15  2015 home
drwxr-xr-x  13 root root  4096 Apr 23 21:51 lib
drwx------   2 root root 16384 Feb 15  2015 lost+found
drwxr-xr-x   2 root root  4096 Aug 19 15:36 media
drwxr-xr-x   8 root root  4096 Aug 19 15:36 mnt
drwxr-xr-x   6 root root  4096 Feb 16  2015 opt
dr-xr-xr-x 100 root root     0 Jan  1  1970 proc
drwx------   4 root root  4096 Aug 19 14:47 root
drwxr-xr-x  14 root root   560 Aug 26 16:24 run
drwxr-xr-x   2 root root  4096 Aug 13 11:53 sbin
drwxr-xr-x   2 root root  4096 Jun 20  2012 selinux
drwxr-xr-x   3 root root  4096 Aug 19 14:09 srv
dr-xr-xr-x  11 root root     0 Aug 27 14:27 sys
drwxrwxrwx   4 root root  4096 Aug 27 15:39 tmp
drwxr-xr-x  11 root root  4096 Mar 24 22:33 usr
drwxrwxrwx  12 root root  4096 Aug 19 14:47 var
```

รูปที่ 4-4 หน้าต่างเทอร์มินอลแสดงการใช้คำสั่ง `ls` และ `ls -al` ในการแสดงรายการ ของไฟล์ในไดเร็กตอรี่



รูปที่ 4-5 คำอธิบายของข้อมูลภายในไดเร็กตอรี่เมื่อเรียกให้แสดงด้วยคำสั่ง `ls -al` ของระบบปฏิบัติการ Raspbian ที่ใช้พื้นฐานสำคัญมาจากระบบปฏิบัติการ Linux

4.4 สิ่งที่ควรทราบเบื้องต้นในการใช้งานบนระบบปฏิบัติการ Linux

การใช้งานคำสั่งบนระบบปฏิบัติการ Linux แตกต่างจากระบบปฏิบัติการ DOS (หรือ Command Prompt บนระบบปฏิบัติการวินโดวส์) ดังนี้

● ชื่อคำสั่ง ชื่อไดเร็กตอรี่ ชื่อไฟล์ และพารามิเตอร์ต่างๆ ทั้งหมดในระบบปฏิบัติการ Linux อักษรตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ มีความหมายต่างกัน ทดแทนกันไม่ได้ จำเป็นต้องเรียกให้ถูกต้องเสมอ

● ระบบปฏิบัติการ Linux ใช้ช่องว่าง (space seperation) ในการแยกคำสั่งกับพารามิเตอร์ เช่นเดียวกับระบบปฏิบัติการ UNIX ดั้งเดิม จะพิมพ์ย่อติดๆ กันแบบระบบปฏิบัติการ DOS เช่น **cd..** ไม่ได้ จะต้องพิมพ์เป็น **cd ..**

● ตัวคั่นไดเร็กตอรี่ใช้ / (ในขณะที่ระบบปฏิบัติการ DOS ใช้ \)

● ~ หมายถึงตำแหน่ง home directory หรือไดเร็กตอรี่ตั้งต้นของผู้ใช้

● . หมายถึงตำแหน่งไดเร็กตอรี่ปัจจุบัน (เหมือนระบบปฏิบัติการ DOS)

● .. หมายถึงตำแหน่งไดเร็กตอรี่ที่อยู่ลำดับบนขึ้นไปหนึ่งอันดับของปัจจุบัน (เหมือนระบบปฏิบัติการ DOS) เช่น ถ้าปัจจุบันอยู่ที่ตำแหน่ง **/var/www** ถ้าพิมพ์คำสั่ง **cd ..** จะเป็นการกลับมายัง **/var**

● โดยปกติไฟล์สั่งงานระบบเกือบทุกตัว จะมีคู่มือการใช้คำสั่งหรือ manual มาด้วย หากต้องการทราบวิธีการใช้พารามิเตอร์ของคำสั่งอย่างละเอียด ให้พิมพ์ **man** นำหน้าคำสั่ง เพื่อเปิดวิธีใช้งาน เช่น **man ls** หน้าจอจะแสดงรายละเอียดของคำสั่ง **ls** พร้อมพารามิเตอร์ที่ใช้งานได้ขึ้นมา

● โดยปกติระบบปฏิบัติการ Raspbian จะล็อกอินด้วย **pi** ซึ่งถือสิทธิ์ผู้ใช้ปกติ หากต้องการทำงานในสิทธิ์ของ super user หรือ root เพื่อเข้าถึงหรือแก้ไขระบบ จะใช้การสั่งงานผ่านโหมดคำสั่ง **sudo (super user do)** เช่น

```
sudo nano /boot/config.txt
```

เรียกตัวแก้ไขไฟล์ตั้งค่าในระบบ ซึ่งต้องใช้สิทธิ์ของ super user ในการแก้ไข

● ระหว่างทำงานคำสั่งบนเทอร์มินอล หากต้องการจบการทำงาน โดยปกติจะใช้การกดแป้น **Ctrl** และ **C** ซึ่งจะส่งสัญญาณ SIGTERM (signal terminate) ให้ระบบเพื่อหยุดทำงาน แต่ถ้าโปรแกรมที่ทำงานอยู่ขณะนั้นเป็นกระบวนการหรือ process ที่ทำงานในระดับที่สูงกว่าหรือไม่ตอบสนองสัญญาณนี้ จำเป็นต้องทำการเปิดใช้เทอร์มินอลอื่น แล้วใช้คำสั่ง **ps** เพื่อดู process ID ที่ทำงานก่อนหน้า แล้วใช้คำสั่ง **kill -9** ตามด้วยเลข process ID ที่ได้จากคำสั่ง **ps** เพื่อบังคับให้หยุด

● ระหว่างทำงานคำสั่งบนเทอร์มินอล หากกดแป้น **Ctrl** และ **Z** แล้วออกมาที่หน้าเชลพร้อมพ์ เป็นเพียงการหยุดกระบวนการแล้วสลับมากลับมาที่เชลพร้อมพ์ชั่วคราวเท่านั้น แต่ตัว process ยังคงอยู่ และพร้อมจะกลับไปทำงานต่อเมื่อพิมพ์คำสั่ง **fg**

● ห้ามถอด microSD การ์ดออกจากบอร์ดอย่างเด็ดขาดในขณะทำงาน

● ก่อนปิดเครื่องหรือตัดไฟเลี้ยงออกจากระบบ ควรทำการหยุดระบบให้เรียบร้อยก่อน โดยใช้คำสั่ง **sudo halt** หรือ **sudo poweroff** หรือ **sudo shutdown now** รอจนระบบหยุดทำงานโดยสมบูรณ์ก่อน จึงตัดไฟเลี้ยง

4.5 ตัวประมวลไปป์ไลน์ (Pipelines) และ ตัวจัดการเปลี่ยนทิศทาง I/O

ในระบบปฏิบัติการ Linux มีตัวควบคุมการทำงานระหว่างคำสั่งในคราวเดียว อันได้แก่

> ใช้เปลี่ยนทิศทางของเอาต์พุต I/O ไปยังไฟล์ I/O ที่ระบุได้ เช่น การเขียนหน้าจอลงไฟล์

```
ls -al  >  output.txt
```

>> ใช้สำหรับเปลี่ยนทิศทางของเอาต์พุต I/O ไปยังไฟล์ I/O เช่นเดียวกับ **>** แต่เป็นการเพิ่มท้ายไฟล์

```
echo "Hello" > test.txt
echo "Raspberry" >> test.txt
```

ถ้าเปิดไฟล์ test.txt ดู จะเห็นข้อความ

```
Hello
Raspberry
```

< ใช้ในกรณีรับอินพุตจากไฟล์ I/O มักใช้ในการเปิดไฟล์ในโปรแกรม

| เป็นไปป์ใช้สำหรับการประมวลผลโดยส่งผ่านเอาต์พุตมาผ่านคำสั่งอีกทอดหนึ่ง เช่น

```
ls -al  |  more
```

กรณีไฟล์ที่แสดงมีจำนวนมากเกินกว่าจะแสดงได้ในหน้าจอเดียว คำสั่ง **more** จะเป็นตัวจำกัดให้เห็นเพียงหน้าจอเดียวก่อน แล้วรอผู้ใช้กดแป้นคีย์บอร์ดเพื่อดำเนินการแสดงหน้าจอถัดไป

tee ใช้ในกรณีที่เป็นไปป์กลาง เพื่อพักข้อมูลก่อน แล้วส่งผ่านไปยังไปป์หลังอีกทอดหนึ่ง เช่น

```
ps ax | tee list_process.txt  |  more
```

อันดับแรกจะเก็บผลลัพธ์ที่ได้จากคำสั่ง **ps** มาไว้ในไฟล์ list_process.txt ก่อน แล้วส่งผ่านไปแสดงบนหน้าจอผ่านทางคำสั่ง **more** ซึ่งถ้า process มีมากกว่าการแสดงในหน้าจอเดียว คำสั่ง tee จะจัดการแสดงหน้าจอในหน้าต่อไปหลังจากมีการกดคีย์บอร์ด

4.6 คำสั่งที่ควรทราบของระบบปฏิบัติการ Linux ที่ใช้กับ Raspberry Pi 2

4.6.1 `ls` (List Directory Contents)

คำสั่งแสดงรายชื่อไฟล์ในไดเร็กตอรี่นั้นๆ ใช้ร่วมกับอักขระช่วยเหลือ (wildcard) เช่น ? และ * ได้ รองรับพารามิเตอร์ต่อท้ายได้หลายแบบ (รายละเอียดให้ดูจาก man ls) ที่เรียกใช้งานบ่อยได้แก่

-a แสดงชื่อไฟล์ที่ซ่อนไว้ให้เห็น (มี . นำหน้าชื่อ)

```
pi@raspberrypi ~ $ ls -a
.                   .config         .gstreamer-0.10  .Mathematica    .thumbnails
..                  .dbus           .gvfs            .pki           .vnc
Adafruit_Python_DHT Desktop         .icons           .profile       wiringPi
.arduino            .dmrc           .idlerc          .pulse         .WolframEngine
.bash_history       Downloads       .java            .pulse-cookie  .Xauthority
.bash_logout        .fceux          .lazarus         python_games   .xsession-errors
.bashrc             .fontconfig     .local           sketchbook     .xsession-errors.old
.cache              .gconf          logs             .sonic-pi
```

-l แสดงชื่อไฟล์แบบแจกแจงรายละเอียด

```
pi@raspberrypi ~ $ ls -l
total 28
drwxr-xr-x 10 pi pi 4096 Aug 14 11:33 Adafruit_Python_DHT
drwxr-xr-x  7 pi pi 4096 Aug 19 14:44 Desktop
drwx------  2 pi pi 4096 Apr 23 16:50 Downloads
drwxr-xr-x  2 pi pi 4096 Aug 19 13:26 logs
drwxrwxr-x  2 pi pi 4096 Jan 27  2015 python_games
drwxr-xr-x  2 pi pi 4096 Mar 24 22:42 sketchbook
drwxr-xr-x  9 pi pi 4096 Apr 23 16:28 wiringPi
```

-al แสดงชื่อไฟล์แบบแจกแจงรายละเอียดและแสดงชื่อที่ซ่อนด้วย

```
pi@raspberrypi ~ $ ls -al
total 324
drwxr-xr-x 30 pi   pi     4096 Aug 27 13:48 .
drwxr-xr-x  3 root root   4096 Feb 15  2015 ..
drwxr-xr-x 10 pi   pi     4096 Aug 14 11:33 Adafruit_Python_DHT
drwxr-xr-x  2 pi   pi     4096 Mar 24 22:42 .arduino
-rw-------  1 pi   pi     1598 Aug 26 16:23 .bash_history
-rw-r--r--  1 pi   pi      220 Feb 15  2015 .bash_logout
-rw-r--r--  1 pi   pi     3243 Feb 15  2015 .bashrc
drwxr-xr-x 12 pi   pi     4096 Apr 23 22:17 .cache
drwxr-xr-x 21 pi   pi     4096 Aug 13 16:05 .config
drwx------  3 pi   pi     4096 Mar 24 21:53 .dbus
drwxr-xr-x  7 pi   pi     4096 Aug 19 14:44 Desktop
-rw-r--r--  1 pi   pi       35 Aug 13 18:22 .dmrc
drwx------  2 pi   pi     4096 Apr 23 16:50 Downloads
drwx------  8 pi   pi     4096 Apr 23 21:20 .fceux
drwxr-xr-x  2 pi   pi     4096 Aug 19 13:25 .fontconfig
drwx------  3 pi   pi     4096 Apr 23 22:02 .gconf
drwxr-xr-x  2 pi   pi     4096 Apr 23 22:01 .gstreamer-0.10
drwx------  2 pi   pi     4096 Aug 24 09:08 .gvfs
drwx------  3 pi   pi     4096 Mar 24 22:26 .icons
drwxr-xr-x  2 pi   pi     4096 Apr 23 15:55 .idlerc
drwxr-xr-x  3 pi   pi     4096 Mar 24 22:42 .java
drwxr-xr-x  3 pi   pi     4096 Apr 23 20:57 .lazarus
drwx------  3 pi   pi     4096 Mar 24 21:53 .local
drwxr-xr-x  2 pi   pi     4096 Aug 19 13:26 logs
drwxr-xr-x  9 pi   pi     4096 Apr 23 18:28 .Mathematica
drwx------  3 pi   pi     4096 Apr 23 15:59 .pki
```

4.6.2 cd (Change Directory)

คำสั่งเปิดเข้าสู่ไดเร็กตอรี่ที่ระบุ ใช้การอ้างอิงแบบสัมบูรณ์หรือ absolute ระบุตำแหน่งโดยตรง หรือแบบสัมพัทธ์หรือ relative โดยใช้ . หรือ .. ร่วมได้

ตัวอย่างที่ 4-1

คำสั่งเปิดเข้าสู่ไดเร็กตอรี่ที่ระบุ ใช้การอ้างอิงแบบสัมบูรณ์หรือ absolute

```
cd  /
cd  /home/pi
cd  ~      ( ~ แทนตำแหน่ง home ของผู้ใช้ ซึ่งมีความหมายเดียวกับ /home/pi )
cd  ~/Desktop
```

ตัวอย่างที่ 4-2

คำสั่งเปิดเข้าสู่ไดเร็กตอรี่ที่ระบุ ใช้การอ้างอิงแบบสัมพัทธ์หรือ relative

ถ้าเดิมอยู่ที่ ~/Desktop ต้องการไปที่ ~/Download ใช้ .. ขึ้นไป 1 อันดับ แล้วเข้าสู่ไดเร็กตอรี่ Downloads

```
pi@raspberrypi ~/Desktop $ cd /var/www
pi@raspberrypi /var/www $
```

ถ้าเดิมอยู่ที่ /usr/local/bin ต้องการไปที่ไดเร็กตอรี่ /var/www ใช้ .. ต่อกัน 3 ครั้งแล้วต่อด้วย var/www ก็ได้

```
pi@raspberrypi ~/Desktop $ cd ../Downloads
pi@raspberrypi ~/Downloads $ _
```

แต่ถ้าหากผู้ใช้ไม่มี permission หรือไม่ได้รับสิทธิ์ในการเข้าถึงไดเร็กตอรี่ จะเข้าสู่ไดเร็กตอรี่ที่เรียกไม่ได้

4.6.3 mkdir (Make Directory)

เป็นคำสั่งสร้างไดเร็กตอรี่ใหม่ตามที่ระบุ ใช้การอ้างอิงแบบสัมบูรณ์ ระบุตำแหน่งโดยตรงหรือแบบสัมพัทธ์ก็ได้

ตัวอย่างที่ 4-3 สร้าง Examples บน Desktop

```
pi@raspberrypi ~/Desktop $ mkdir Examples
pi@raspberrypi ~/Desktop $ ls -l
total 16
drwxr-xr-x 2 pi pi 4096 Aug 30 00:40 Examples
```

ใช้คำสั่ง **ls** *เพื่อเข้าดูรายละเอียด จะเห็นว่า pi เป็น owner และ pi มี permission หรือมีสิทธิ์เข้าถึงจึงเห็นรายละเอียดของไฟล์ Examples ได้*

4.6.4 sudo (Super User Do)

ใช้นำหน้าคำสั่งใดๆ ที่ต้องการใช้สิทธิ์ (Granted) ของ super user หรือ root ซึ่งเข้าถึง ปรับแต่งแก้ไขส่วนประกอบของระบบได้ทุกอย่าง และอยู่เหนือ permission หรือสิทธิ์ในการเข้าถึงไฟล์และไดเร็กตอรี่ทั้งปวง

ตัวอย่างที่ 4-4

โดยปกติสำหรับไฟล์ตั้งค่าระบบ ผู้ใช้ปกติจะแก้ไขและบันทึกไม่ได้ เมื่อใช้คำสั่ง sudo นำหน้า nano ก็จะเข้าถึงไฟล์นั้นได้

```
pi@raspberrypi ~ $ sudo nano /boot/config.txt
```

4.6.5 rmdir (Remove Directory)

เป็นคำสั่งลบไดเร็กทอรี่ตามที่ระบุ ก่อนใช้คำสั่งนี้ จะต้องไม่มีไฟล์เหลือในไดเร็กตอรี่ด้วย จึงจะลบออกได้ แต่หากต้องการลบโดยไม่สนใจไฟล์ด้านใน ให้ใช้คำสั่ง **rm -Rf** แทน

รูปแบบ

rmdir ชื่อไดเร็กตอรี่ที่ต้องการลบ

ตัวอย่างที่ 4-5 ลบ Examples บน Desktop

```
pi@raspberrypi ~/Desktop $ rmdir Examples
```

4.6.6 rm (Remove)

เป็นคำสั่งลบไฟล์ที่รองรับทั้งการระบุไฟล์โดยตรง และใช้ร่วมกับอักขระช่วยเหลือทั้ง ? และ * ได้ รวมถึงรองรับพารามิเตอร์ต่อท้ายได้หลายแบบ (รายละเอียดให้ดูจาก man rm)

```
pi@raspberrypi ~/Desktop $ ls -al
total 20
drwxr-xr-x  2 pi pi 4096 Aug 30 00:48 .
drwxr-xr-x 25 pi pi 4096 Aug 30 17:52 ..
-rw-r--r--  1 pi pi 5316 Aug 29 10:21 geany.desktop
-rw-r--r--  1 pi pi   16 Aug 29 09:24 test.py
pi@raspberrypi ~/Desktop $ rm test.py
pi@raspberrypi ~/Desktop $ ls -al
total 16
drwxr-xr-x  2 pi pi 4096 Aug 30 17:55 .
drwxr-xr-x 25 pi pi 4096 Aug 30 17:52 ..
-rw-r--r--  1 pi pi 5316 Aug 29 10:21 geany.desktop
pi@raspberrypi ~/Desktop $ _
```

แต่ถ้าไม่มีสิทธิ์หรือ permission ในการแก้ไข ก็ต้องใช้ **sudo** นำหน้าไฟล์เพื่อลบด้วยสิทธิ์ของ super user อีกกรณีที่ใช้บ่อยคือ พารามิเตอร์ **-Rf** เพื่อลบไดเร็กตอรี่พร้อมไฟล์ด้านในทั้งหมด

ตัวอย่างที่ 4-6

```
pi@raspberrypi ~/Examples $ cd ..
pi@raspberrypi ~ $ rm -Rf Examples
pi@raspberrypi ~ $ ls
Adafruit_Python_DHT  Downloads  Pictures      simpletest.py
Desktop              indiecity  Public        Templates
Documents            Music      python_games  Videos
```

แสดงการใช้คำสั่ง **rm** ที่ต่อท้ายด้วยพารามิเตอร์ **-Rf** สำหรับลบไดเร็กตอรี่และไฟล์ภายในทั้งหมด

4.6.7 cp (Copy)

เป็นคำสั่งคัดลอกไฟล์ที่ระบุไปยังที่อยู่ปลายทาง ใช้ร่วมกับอักขระช่วยเหลือทั้ง ? และ * ได้ นอกจากนั้นยังรองรับการใส่พารามิเตอร์ต่อท้ายได้หลายแบบ (รายละเอียดให้ดูจาก man cp)

```
pi@raspberrypi ~/Desktop $ ls
geany.desktop  test.py
pi@raspberrypi ~/Desktop $ cp test.py ../Examples
```

ตัวอย่างที่ 4-7

```
pi@raspberrypi ~/Adafruit_Python_DHT/examples $ ls
AdafruitDHT.py  google_spreadsheet.py  simpletest.py
pi@raspberrypi ~/Adafruit_Python_DHT/examples $ cp *.py ~/Examples
pi@raspberrypi ~/Adafruit_Python_DHT/examples $ ls ~/Examples
AdafruitDHT.py  google_spreadsheet.py  simpletest.py
```

*ใช้อักขระช่วยเหลือ * เพื่อคัดลอกไฟล์ชื่อใดก็ได้ที่มี .py ต่อท้าย*

4.6.8 mv (Move)

เป็นคำสั่งย้ายไฟล์ที่ระบุไปยังปลายทาง คล้ายกับคำสั่ง cp แต่จะลบไฟล์ต้นทางด้วย

รูปแบบ

1. ในกรณีที่ต้องการย้ายไฟล์ไปยังไดเร็กตอรี่ปลายทางเท่านั้น

   mv ไฟล์ที่ต้องการย้าย ไดเร็กตอรี่ปลายทาง

2. ในกรณีที่ต้องการย้ายไฟล์ไปยังไดเร็กตอรี่ปลายทาง และเปลี่ยนชื่อไฟล์

   mv ไฟล์ที่ต้องการย้าย ไดเร็กตอรี่ปลายทาง/กำหนดชื่อไฟล์ปลายทางที่เกิดขึ้นหลังการย้ายไฟล์

ตัวอย่างที่ 4-8

```
pi@raspberrypi ~/Examples $ ls
AdafruitDHT.py  google_spreadsheet.py  simpletest.py  test.py
pi@raspberrypi ~/Examples $ mv simpletest.py ..
pi@raspberrypi ~/Examples $ ls
AdafruitDHT.py  google_spreadsheet.py  test.py
pi@raspberrypi ~/Examples $ ls ../*.py
../simpletest.py
```

4.6.9 cat (Concatenation)

เป็นคำสั่งแสดงข้อมูลที่อยู่ในไฟล์นั้นๆ ในรูปของแฟ้มข้อความ หรือ Plain Text

รูปแบบ

cat filenames

โดยที่

`filenames` - ไฟล์ที่ต้องการให้แสดงข้อมูล

ตัวอย่างที่ 4-9

```
pi@raspberrypi ~/Desktop/rpi $ cat sample.py
import RPi.GPIO
import timer

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

print "Hello Raspberry Pi"
pi@raspberrypi ~/Desktop/rpi $
```

เป็นการกำหนดให้แสดงข้อมูลของไฟล์ sample.py ในรูปของข้อความ เมื่อป้อนคำสั่งแล้วกด Enter บอร์ด Raspberry Pi แสดงข้อมูลของไฟล์ในบรรทัดต่อๆ มา จนครบ แล้วกลับมาที่เชลพร้อมพ์

นอกจากนั้น คำสั่งนี้ยังนำข้อมูลที่ได้ไปรวมกับอีกไฟล์ได้ โดยข้อมูลจะอยู่ต่อท้ายจากของเดิมที่มีอยู่ในไฟล์นั้นๆ และถ้าไฟล์ปลายทางยังไม่มี ก็จะสร้างไฟล์ขึ้นมาให้ทันที ดังตัวอย่างที่ 4-11

ตัวอย่างที่ 4-10

```
pi@raspberrypi ~/Desktop/rpi $ ls -l
total 8
-rwxrw-r-- 1 pi users 105 Apr 14 00:29 sample.py
-rw-r--r-- 1 pi pi     50 Apr 14 00:38 test.py
pi@raspberrypi ~/Desktop/rpi $ cat test.py
#Hello Raspberry Pi
#This is test file for python
pi@raspberrypi ~/Desktop/rpi $ cat sample.py >> test.py
pi@raspberrypi ~/Desktop/rpi $ cat test.py
#Hello Raspberry Pi
#This is test file for python
import RPi.GPIO
import timer

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

print "Hello Raspberry Pi"
pi@raspberrypi ~/Desktop/rpi $
```

ในตัวอย่างนี้เป็นการนำข้อมูลของไฟล์ sample.py ไปต่อท้ายข้อมูลในไฟล์ test.py โดยในตอนเริ่มต้นใช้คำสั่ง **cat test.py** เพื่อแสดงให้เห็นข้อมูลในไฟล์ test.py ก่อน ซึ่งมี 2 บรรทัดคือ

```
#Hello Raspberry Pi
#This is test file for python
```

จากนั้นรันคำสั่ง **cat sample.py >> test.py** เพื่อนำข้อมูลจากไฟล์ sample.py ซึ่งมีรายละเอียดแสดงในตัวอย่างที่ 4-9 ไปต่อท้ายข้อมูลในไฟล์ test.py

ถัดจากนั้น รันคำสั่ง **cat test.py** อีกครั้ง เพื่อแสดงข้อมูลในไฟล์ test.py หลังจากที่มีการนำข้อมูลจากไฟล์ sample.py มาต่อท้าย จะเห็นข้อมูลเพิ่มขึ้นอีก 5 บรรทัดดังนี้

```
#Hello Raspberry Pi
#This is test file for python
import Rpi.GPIO
import timer
GPIO.setwaring(False)
GPIO.setmode(GPIO.BCM)
print "Hello Raspberry Pi"
```

แล้วกลับมาที่เชลพร้อมพ์

4.6.10 echo

เป็นคำสั่งส่งผ่านข้อความที่พิมพ์ไปยังเอาต์พุต ถ้าไม่ระบุปลายทางจะเป็นการแสดงออกมาที่เทอร์มินอล

● ถ้าระบุเอาต์พุตด้วย > และต่อท้ายด้วยชื่อไฟล์ จะเป็นการเขียนข้อความลงไฟล์นั้น ถ้ามีไฟล์เดิมจะถูกลบและเขียนข้อความใหม่ทับ

```
pi@raspberrypi ~ $ ls
Adafruit_Python_DHT  Documents  indiecity  Pictures  python_games  Videos
Desktop              Downloads  Music      Public    Templates
pi@raspberrypi ~ $ echo "Hello Raspberry Pi" > output.txt
pi@raspberrypi ~ $ ls
Adafruit_Python_DHT  Documents  indiecity  output.txt  Public        Templates
Desktop              Downloads  Music      Pictures    python_games  Videos
pi@raspberrypi ~ $ cat output.txt
Hello Raspberry Pi
pi@raspberrypi ~ $ echo "This is echo from terminal" > output.txt
pi@raspberrypi ~ $ cat output.txt
This is echo from terminal
```

● ถ้าต้องการเขียนเพิ่มต่อท้ายไฟล์ จะใช้พารามิเตอร์ >> แทน

```
pi@raspberrypi ~ $ echo "Test message #1" >> output.txt
pi@raspberrypi ~ $ cat output.txt
This is echo from terminal
Test message #1
```

4.6.11 pwd (Print Working Directory)

เป็นคำสั่งแสดงตำแหน่งของไดเร็กตอรี่ปัจจุบัน

```
pi@raspberrypi ~/Desktop $ pwd
/home/pi/Desktop
```

4.6.12 hostname

เป็นคำสั่งแสดงชื่อเครื่องปัจจุบันที่ทำงาน

```
pi@raspberrypi ~/Desktop $ hostname
raspberrypi
```

4.6.13 df (Disk Free)

เป็นคำสั่งสำหรับดูการใช้พื้นที่ว่างสำหรับเก็บข้อมูลใน microSD การ์ด

```
pi@raspberrypi ~/Desktop $ df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root        7319248 3978640   2992976  58% /
devtmpfs          372316       0    372316   0% /dev
tmpfs              75328     236     75092   1% /run
tmpfs              5120       0      5120   0% /run/lock
```

4.6.14 date

เป็นคำสั่งแสดงหรือตั้งค่าวันเวลาของเครื่อง

● ถ้าพิมพ์ date จะเป็นการแสดงเวลาของระบบออกมา

● ถ้าต้องการตั้งค่าวันเวลา จะต้องใช้พารามิเตอร์ **-s** ตามด้วยข้อความวันเวลา ดังตัวอย่าง และต้องใช้สิทธิ์ super user ด้วย

ตัวอย่างที่ 4-11

```
pi@raspberrypi ~/Desktop $ date
Sun Aug 30 18:09:38 ICT 2015
pi@raspberrypi ~/Desktop $ date -s "Sun Aug 30 18:08:38 ICT 2015"
date: cannot set date: Operation not permitted
Sun Aug 30 18:08:38 ICT 2015
pi@raspberrypi ~/Desktop $ sudo date -s "Sun Aug 30 18:08:38 ICT 2015"
Sun Aug 30 18:08:38 ICT 2015
```

4.6.15 chmod (Change file Mode bit)

เป็นคำสั่งเปลี่ยนโหมดไฟล์และไดเร็กตอรี่ ใช้ในการแก้ไขสิทธิ์ในการเข้าถึงไฟล์หรือ permission การแก้ไข permission ด้วยคำสั่ง chmod นิยมทำสองวิธีหลักๆ คือ

1. อ้างอิงแบบ u g o a (user,group,other,all permission) +- (เปิด,ปิด) r w x (read,write,execute)

```
pi@raspberrypi ~ $ ls -l output.txt
-rw------- 1 pi pi 19 Aug 30 20:51 output.txt
pi@raspberrypi ~ $ chmod u+rwx output.txt
pi@raspberrypi ~ $ ls -l output.txt
-rwx------ 1 pi pi 19 Aug 30 20:51 output.txt
pi@raspberrypi ~ $ chmod go+rx output.txt
pi@raspberrypi ~ $ ls -l output.txt
-rwxr-xr-x 1 pi pi 19 Aug 30 20:51 output.txt
```

2. อีกวิธีหนึ่งที่ยืดหยุ่นกว่าและนิยมใช้มากกว่าคือ **ระบุ permission rwx 3 บิต 3 ชุดเรียงกัน** เขียนในรูปแบบเลขฐานแปด เช่น **rwx r-x r-x** เรียกเป็น 755

```
pi@raspberrypi ~ $ ls -l output.txt
-rw------- 1 pi pi 19 Aug 30 20:51 output.txt
pi@raspberrypi ~ $ chmod 755 output.txt
pi@raspberrypi ~ $ ls -l output.txt
-rwxr-xr-x 1 pi pi 19 Aug 30 20:51 output.txt
```

แต่ถ้าไม่มีสิทธิ์ในการแก้ไข ก็ต้องใช้คำสั่ง **sudo** นำหน้าไฟล์เพื่อตั้งค่าด้วยสิทธิ์ของ super user

4.6.16 chgrp (Change Group ownership)

เป็นคำสั่งเปลี่ยนกลุ่มไฟล์และไดเร็กตอรี่

รูปแบบคำสั่ง

chgrp ชื่อกลุ่ม ชื่อไฟล์หรือไดเร็กตอรี่ที่ต้องการเปลี่ยน

```
pi@raspberrypi ~ $ ls -l output.txt
-rwxr-xr-x 1 pi pi 19 Aug 30 20:51 output.txt
pi@raspberrypi ~ $ sudo chgrp root output.txt
pi@raspberrypi ~ $ ls -l output.txt
-rwxr-xr-x 1 pi root 19 Aug 30 20:51 output.txt
```

4.6.17 chown (Change file Owner and Group)

เป็นคำสั่งเปลี่ยนเจ้าของไฟล์และไดเร็กตอรี่

รูปแบบคำสั่ง

chown ชื่อผู้ใช้ ชื่อไฟล์หรือไดเร็กตอรี่ที่ต้องการเปลี่ยน

```
pi@raspberrypi ~ $ ls -l output.txt
-rwxr-xr-x 1 pi pi 19 Aug 30 20:51 output.txt
pi@raspberrypi ~ $ sudo chown root output.txt
pi@raspberrypi ~ $ ls -l output.txt
-rwxr-xr-x 1 root pi 19 Aug 30 20:51 output.txt
```

หากต้องการเปลี่ยนทั้ง owner และ group ในคราวเดียวกันจะใช้ owner:group ดังนี้

```
pi@raspberrypi ~ $ ls -l output.txt
-rwxr-xr-x 1 pi pi 19 Aug 30 20:51 output.txt
pi@raspberrypi ~ $ sudo chown root:root output.txt
pi@raspberrypi ~ $ ls -l output.txt
-rwxr-xr-x 1 root root 19 Aug 30 20:51 output.txt
```

4.6.18 `grep` (Print line matching a pattern)

เป็นคำสั่งสำหรับตรวจและแสดงข้อความที่ค้นหา ปกติจะใช้ร่วมกับคำสั่งที่ส่งข้อความผ่านไปป์ | มาอีกทอดหนึ่ง (| คืออักขระที่เป็นคีย์บนของคีย์ \ ไม่ใช่ตัวอักษรแอลตัวพิมพ์เล็ก)

ตัวอย่างที่ 4-12

```
pi@raspberrypi ~ $ ls -l
total 48
drwxr-xr-x 10 pi   pi   4096 Aug 29 05:44 Adafruit_Python_DHT
drwxr-xr-x  2 pi   pi   4096 Aug 30 17:55 Desktop
drwxr-xr-x  2 pi   pi   4096 Aug 29 10:12 Documents
drwxr-xr-x  2 pi   pi   4096 Aug 29 10:12 Downloads
drwxr-xr-x  3 pi   pi   4096 Aug 29 09:12 indiecity
drwxr-xr-x  2 pi   pi   4096 Aug 29 10:12 Music
-rwxr-xr-x  1 root root   19 Aug 30 20:51 output.txt
drwxr-xr-x  2 pi   pi   4096 Aug 29 10:12 Pictures
drwxr-xr-x  2 pi   pi   4096 Aug 29 10:12 Public
drwxrwxr-x  2 pi   pi   4096 Jan 27  2015 python_games
drwxr-xr-x  2 pi   pi   4096 Aug 29 10:12 Templates
drwxr-xr-x  2 pi   pi   4096 Aug 29 10:12 Videos
pi@raspberrypi ~ $ ls -l | grep output
-rwxr-xr-x  1 root root   19 Aug 30 20:51 output.txt
```

ในตัวอย่างนี้ต้องการค้นหาข้อความ output ผลการค้นหาคือ พบไฟล์ output.txt

4.6.19 `ps` (Process report snapshot)

เป็นคำสั่งดูกระบวนการหรือ process ของคำสั่งต่างๆ ที่ทำงานในระบบ ณ เวลานั้น

ปกติมักจะต่อท้ายด้วยพารามิเตอร์ **`ax`** เพื่อดูคำสั่งที่ทำงานอยู่เบื้องหลังด้วย และเพื่อความสะดวกในการไล่ดู จะใช้ร่วมกับ **`|`** **`grep`** เพื่อแยกเอาเฉพาะ process ที่สนใจมาแสดง

ตัวอย่างที่ 4-13

```
pi@raspberrypi ~ $ ps ax | grep python3
 2814 pts/1    S+     0:00 sudo python3 test.py
 2815 pts/1    S+     0:00 python3 test.py
 2818 pts/0    S+     0:00 grep --color=auto python3
```

ในตัวอย่างนี้ ต้องการดู process ของ python3 ที่ทำงานอยู่

4.6.20 `kill` (kill process)

เป็นคำสั่งยกเลิกการทำงานของ process ID ที่ระบุ

ถ้าหากเป็น process ที่สามารถหยุดได้ด้วยสัญญาณ SIGTERM ปกติ เพียงเรียกคำสั่ง

`sudo kill` ตามด้วยเลข process ID

กระบวนการหรือ process ที่ระบุนั้นจะหยุดทำงานทันที

áµ¶éà»ç process ·ëÁèÃÑÊÑ- Ò³ SIGTERM ¨Ðã¾Ô¾ÒÃÒÁà¾ÔÍ Ãì-9 นำหน้าเลข process ID เพื่อบังคับให้หยุด ดังรูป

```
pi@raspberrypi ~ $ ps ax | grep python3
 2814 pts/1    S+     0:00 sudo python3 test.py
 2815 pts/1    S+     0:00 python3 test.py
 2837 pts/0    S+     0:00 grep --color=auto python3
pi@raspberrypi ~ $ sudo kill -9 2815
pi@raspberrypi ~ $ ps ax | grep python3
 2842 pts/0    R+     0:00 grep --color=auto python3
```

4.6.21 `passwd` (change password)

เป็นคำสั่งเปลี่ยนรหัสผ่านของผู้ใช้ปัจจุบัน เมื่อรันคำสั่งจะถามรหัสผ่านเดิมก่อน จากนั้นจึงพิมพ์รหัสผ่านใหม่สองครั้งเพื่อยืนยันการแก้ไข

```
pi@raspberrypi ~ $ passwd
Changing password for pi.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

4.6.22 clear

เป็นคำสั่งล้างข้อความบนหน้าจอเทอร์มินอล

ตัวอย่างที่ 4-14

หน้าจอเทอร์มินอลก่อนรันคำสั่ง **clear**

```
drwxr-xr-x  2 pi   pi        4096 Aug 29 10:12 Templates
drwxr-xr-x  3 pi   pi        4096 Aug 29 05:26 .themes
drwx------  4 pi   pi        4096 Aug 29 09:22 .thumbnails
drwxr-xr-x  2 pi   pi        4096 Aug 29 10:12 Videos
drwx------  2 pi   pi        4096 Aug 30 00:00 .vnc
-rw-------  1 pi   pi          57 Aug 30 21:25 .Xauthority
-rw-------  1 pi   pi       46742 Aug 30 00:00 .xsession-errors
pi@raspberrypi ~ $ ls
Adafruit_Python_DHT  Documents  indiecity  output.txt  Public        Templates
Desktop              Downloads  Music      Pictures    python_games  Videos
pi@raspberrypi ~ $ clear_
```

หน้าจอเทอร์มินอลหลังรันคำสั่ง **clear**

```
pi@raspberrypi ~ $
```

4.6.23 ifconfig (Network Interface Config)

เป็นคำสั่งตรวจสอบการต่อเชื่อมเครือข่าย

ตัวอย่างที่ 4-15

```
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:07:b4:e0
          inet addr:192.168.1.44  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:25456 errors:0 dropped:3 overruns:0 frame:0
          TX packets:3749 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1291062 (1.2 MiB)  TX bytes:477509 (466.3 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

จากตัวอย่างแสดงให้เห็นว่า บอร์ด Raspberry Pi 2 เชื่อมต่อกับเครือข่ายผ่านพอร์ตอีเธอร์เน็ต (LAN) ทางช่องทาง eth0

4.6.24 ping

เป็นคำสั่งตรวจสอบการตอบสนองของ IP หรือ hostname ที่ระบุ

ถ้าไม่ใช้พารามิเตอร์ -c ตามด้วยจำนวนครั้งที่ต้องการ ping โปรแกรมจะทำการ ping อย่างµé ä é § Ë Ò µé §¡ ÒÃ ¨ ° ¡ ÒÃ·íÒ§Ò¹ ãËé ´ ¤Ø Ctrl และ C

```
pi@raspberrypi ~ $ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=0.866 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=0.663 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=0.668 ms
64 bytes from 192.168.1.1: icmp_req=4 ttl=64 time=0.639 ms
64 bytes from 192.168.1.1: icmp_req=5 ttl=64 time=0.624 ms
^Z
[3]+  Stopped                 ping 192.168.1.1
```

4.6.25 nano

เป็นคำสั่งเรียกโปรแกรมเท็กซ์เอดิเตอร์ที่ชื่อว่า **nano** ขึ้นมาทำงาน โดย nano เป็นโปรแกรมอำนวยความสะดวกสำหรับแก้ไขไฟล์ข้อความ

<u>รูปแบบคำสั่ง</u>

nano ไฟล์ที่ต้องการแก้ไข

เมื่อรัน จะเห็นหน้าโปรแกรมดังรูป

```
  GNU nano 2.2.6          File: Desktop/test.py

import time
while True:
        print ("Test Message")
        time.sleep(1000)

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

4.6.26 apt-get (Advance Package Tools : get)

เป็นคำสั่งจัดการแพ็กเกจภายในเครื่อง ช่วยในค้นหา ติดตั้ง และอัปเดตแพ็กเกจให้สะดวกขึ้น ควรใช้ควบคู่กับคำสั่ง **sudo**

<u>รูปแบบคำสั่ง</u>

ที่ใช้งานบ่อยคือ

sudo apt-get install ชื่อแพ็กเก็จที่ต้องการติดตั้งเพิ่ม

sudo apt-get remove ชื่อแพ็กเก็จที่ต้องการถอนออกจากระบบ

หากต้องการตรวจสอบการปรับเวอร์ชั่นแพ็กเกจโปรแกรมต่างๆที่มีอยู่ทั้งหมด จะใช้

sudo apt-get update

หลังจากตรวจสอบแล้ว ถ้าต้องการปรับเวอร์ชั่นใหม่ จะใช้คำสั่งนี้ตามหลัง

sudo apt-get upgrade

4.2.27 wget

เป็นคำสั่งเรียกใช้งานตัวช่วยเหลือในการดาวน์โหลดไฟล์จากอินเทอร์เน็ต

<u>รูปแบบ</u>

wget URL

โดยที่

URL - ตำแหน่งเว็บไซต์ของไฟล์ที่ต้องการดาวน์โหลด

<u>ตัวอย่างที่ 4-16</u>

```
pi@raspberrypi ~/Desktop/rpi $ wget http://sleepingforless.com/downloads/sample_music.mp3
--2015-04-14 00:57:36--  http://sleepingforless.com/downloads/sample_music.mp3
Resolving sleepingforless.com (sleepingforless.com)... 162.243.35.63
Connecting to sleepingforless.com (sleepingforless.com)|162.243.35.63|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3080306 (2.9M) [audio/mpeg]
Saving to: `sample_music.mp3.1'

100%[======================================================================>] 3,080,306    241K/s   in 13s

2015-04-14 00:57:50 (223 KB/s) - `sample_music.mp3.1' saved [3080306/3080306]

pi@raspberrypi ~/Desktop/rpi $
```

จากตัวอย่างเป็นการสั่งให้บอร์ด Raspberry Pi ดาวน์โหลดไฟล์ sample_music.mp3 จากเว็บไซต์ sleepingforless.com โดยมีการแสดงข้อมูลของไฟล์ที่ทำการดาวน์โหลดด้วย เมื่อการดาวน์โหลดเสร็จสิ้นจะกลับมายังเชลพร้อมพ์

4.6.28 reboot

เป็นคำสั่งรีบูตระบบใหม่

รูปแบบคำสั่ง

```
sudo   reboot
```

4.6.29 poweroff หรือ halt หรือ shutdown

เป็นคำสั่งปิดระบบ

รูปแบบคำสั่ง

sudo poweroff

หรือ

sudo halt

หรือ

sudo shutdown เวลา

โดยที่ค่าเวลามีหน่วยเป็นวินาที

ตัวอย่างที่ 4-17

```
sudo shutdown 60
```

เป็นการสั่งปิดการระบบภายใน 60 วินาที

ตัวอย่างที่ 4-18

```
sudo poweroff
```

เป็นคำสั่งปิดการทำงานของระบบทันที

```
sudo halt
```

เป็นคำสั่งปิดการทำงานของระบบทันที

```
sudo shutdown now
```

เป็นคำสั่งปิดการทำงานของระบบทันที

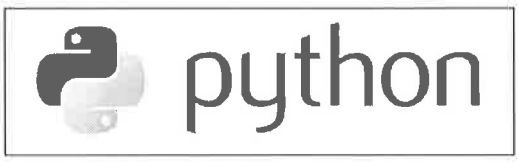
บทที่ 5
เริ่มต้นรู้จักกับภาษา Python

ในการพัฒนาโปรแกรมสำหรับ Raspberry Pi 2 ทำได้หลากหลายภาษา ทั้ง C, BASIC, JAVA, Pascal, Assembly หรือ Python ซึ่งเหมือนกับคอมพิวเตอร์ที่ใช้งานทั่วไป เพราะ Raspberry Pi 2 ก็คือบอร์ดคอมพิวเตอร์ 32 บิตที่ติดตั้งระบบปฏิบัติการ Raspbian อันเป็นหนึ่งในระบบปฏิบัติการ Linux ที่มีการปรับให้ทำงานบนบอร์ด Raspberry Pi ได้อย่างมีประสิทธิภาพ

ภาษาที่นิยมใช้ในการพัฒนาโปรแกรมบน Raspberry Pi คือ Python และมีตัวอย่างโปรแกรมให้ดาวน์โหลดมาเรียนรู้และใช้งานมากมายในเครือข่ายอินเทอร์เน็ต ในบทนี้นำเสนอเนื้อหาอันเป็นการแนะนำเกี่ยวกับ Python เบื้องต้น

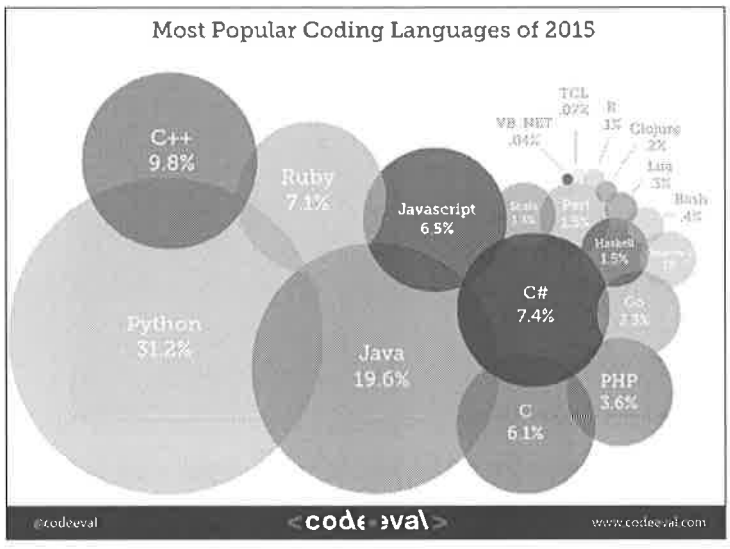
5.1 Python คืออะไร?

Python เป็นภาษาโปรแกรมระดับสูงที่ถือกำเนิดขึ้นเมื่อปี ค.ศ. 1990 โดย **Guido van Rossum** ซึ่งเป็นโปรแกรมเมอร์ชาวดัตช์หรือเนเธอร์แลนด์ *โดยชื่อ Python มาจากภาพยนต์ซีรีส์ของสถานีโทรทัศน์ BBC ที่ชื่อว่า Monty Python's Flying Circus* ในปัจจุบันนี้ภาษา Python อยู่ภายใต้การดูแลของมูลนิธิซอฟต์แวร์ Python



รูปที่ 5-1 ตราสัญลักษณ์ของภาษา Python

รูปที่ 5-2 Guido van Rossum บิดาแห่ง Python



รูปที่ 5-3 แสดงสัดส่วนของจำนวนการใช้งานและพัฒนาโปรแกรมด้วยภาษาคอมพิวเตอร์ จะเห็นว่า ภาษา Python ได้รับความนิยมสูงสุด

Python เป็นอีกหนึ่งภาษาโปรแกรมคอมพิวเตอร์ที่ได้รับความนิยมสูงมากในปัจจุบัน เนื่องจากตัวภาษาทำความเข้าใจได้ไม่ยาก มี Syntax ที่ไม่ยุ่งยากเมื่อเปรียบเทียบกับภาษาอื่น จึงทำให้เรียนรู้ได้ง่าย และมีการทำงานแบบอินเตอร์พรีตเตอร์ (Interpreter) หรือแบบแปลภาษา

5.2 เริ่มใช้งาน Python บน Raspberry Pi

ในการใช้งาน Python บน Raspberry Pi ทำได้ 2 แบบคือ

1. เชลสคริปต์ (Shell Script) โดยเขียนชุดคำสั่งไว้ในไฟล์ใดๆ แล้วสั่งให้ Python ทำการเอ็กซิคิวต์ไฟล์นั้นๆ

2. อินเตอร์แอกตีฟเชล (Interactive Shell) ที่ให้ผู้พัฒนาโปรแกรมพิมพ์คำสั่งลงในเทอร์มินอลได้ทันที แล้ว Python จะทำงานทันทีในทุกๆ บรรทัดที่พิมพ์ เหมาะแก่การฝึกเขียนโปรแกรม Python ได้เป็นอย่างดี

5.2.1 การใช้งานเชลสคริปต์ของ Python บน Raspberry Pi

หลักการของเชลสคริปต์คือ สร้างไฟล์ที่มีชุดคำสั่งที่พิมพ์เตรียมไว้แล้ว แล้วสั่งให้ชุดคำสั่งที่อยู่ในไฟล์นั้นทำงาน

แนวทางการใช้งานเชลสคริปต์มีดังนี้

(1) สร้างไฟล์ขึ้นมาหนึ่งไฟล์ โดยใช้ Nano ในการสร้างไฟล์ แล้วใส่คำสั่งไว้ข้างใน โดยพิมพ์คำสั่งว่า **`sudo nano mycode.py`**

```
pi@raspberrypi ~ $ sudo nano mycode.py
```

(2) ที่ข้าง **Nano Text Editor** ให้เพิ่มคำสั่ง **`print ("Python is very cool").`**

```
  GNU nano 2.2.6       File: mycode.py          Modified

print ("Python is very cool")



^G Get He^O WriteO^R Read F^Y Prev P^K Cut Te^C Cur Pos
^X Exit  ^J Justif^W Where ^V Next P^U UnCut ^T To Spell
```

(3) ทำการปิดโปรแกรม **Nano** ลง พร้อมกับบันทึกไฟล์ดังกล่าวไว้ด้วย *(กด Ctrl และ X ตามด้วย Y แล้วกด Enter)*

(4) วิธีการสั่งให้ไฟล์ดังกล่าวทำงาน จะใช้คำสั่งว่า **python** แล้วตามด้วยชื่อไฟล์นั้นๆ ดังนั้น คำสั่งจึงเป็น **python mycode.py**

```
pi@raspberrypi ~ $ python3 mycode.py
```

(5) เมื่อกด **Enter** จะเห็นข้อความ **Python is very cool** แสดงขึ้นมา ซึ่งเกิดจากคำสั่งที่สร้างไว้ในไฟล์ **mycode.py** นั่นเอง

```
pi@raspberrypi ~ $ python3 mycode.py
Python is very cool
pi@raspberrypi ~ $
```

5.2.2 การใช้งานอินเตอร์แอกตีฟเชลของ Python บน Raspberry Pi 2

(1) เข้าสู่หน้าต่างเทอร์มินอล แล้วรันคำสั่ง **python** เพื่อเรียกอินเตอร์แอกตีฟเชลขึ้นมา

```
pi@raspberrypi ~ $ python3
```

(2) เมื่อกด **Enter** อินเตอร์แอกตีฟเชลของ Python เริ่มทำงาน และหยุดที่บรรทัด **>>>** ซึ่งบรรทัดดังกล่าวนี้คือ ตำแหน่งสำหรับพิมพ์คำสั่งของโปรแกรม

```
pi@raspberrypi ~ $ python3
Python 3.2.3 (default, Mar  1 2013, 11:53:50)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

(3) ให้พิมพ์คำสั่ง **print ("Python is very cool")**

```
pi@raspberrypi ~ $ python3
Python 3.2.3 (default, Mar  1 2013, 11:53:50)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Python is very cool")
```

(4) เมื่อกด **Enter** จะเห็นว่า ที่บรรทัดถัดไปมีการแสดงข้อความว่า **Python is very cool**

```
pi@raspberrypi ~ $ python3
Python 3.2.3 (default, Mar  1 2013, 11:53:50)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Python is very cool")
Python is very cool
>>>
```

ข้อความที่แสดงนั้นเกิดมาจากการใช้คำสั่ง print หากต้องการให้แสดงข้อความใด ข้อความนั้นๆ จะถูกแสดงในบรรทัดถัดไป นี่คือรูปแบบการทำงานของอินเตอร์แอกตีฟเชลนั่นเอง เมื่อทำงานเสร็จแล้วก็จะแสดงสัญลักษณ์ >>> เพื่อให้พิมพ์คำสั่งต่อไปได้ทันที

5.3 คำสั่ง print

เป็นคำสั่งใช้แสดงข้อมูลหรือข้อความใดๆ บนหน้าจอแสดงผล นับเป็นหนึ่งในคำสั่งพื้นฐานที่สำคัญของ Python

รูปแบบ

print (ข้อมูลหรือข้อความ)

ตัวอย่างที่ 5-1

```
>>> print ("Innovative Experiment")
```

ผลที่ได้ : **Innovative Experiment**

```
>>> print ('Innovative Experiment')
```

ผลที่ได้ : **Innovative Experiment**

```
>>> print ("Hi! I'm R-Pi lover")
```

ผลที่ได้ : **Hi! I'm R-Pi lover**

```
>>> print ('Hi! I\'m R-Pi lover')
```

ผลที่ได้ : **Hi! I'm R-Pi lover**

```
>>> print ("""This is
... new line
... command""")
```

ผลที่ได้ :

This is
new line
command

```
>>> print ("This is\nnew line\ncommand")
```

ผลที่ได้ :

This is
new line
command

5.4 เครื่องหมายพิเศษบางส่วนใน Python

รูปแบบ	ผลลัพธ์
\'	'
\"	"
\\	\
\a	มีเสียงดังหนึ่งครั้ง
\f	ขึ้นแถวใหม่

```
>>> print ("AA\fAA")
AA
  AA
```

\n	ขึ้นบรรทัดใหม่และชิดซ้ายมือ

```
>>> print ("AA\nAA")
AA
AA
```

\t	แท็บหนึ่งครั้ง

python Note

สำหรับการพัฒนาแอปพลิเคชั่นให้แก่บอร์ด Raspberry Pi ด้วยโปรแกรมภาษา Python เป็นสิ่งที่ได้รับความนิยมสูงมาก มีโค้ดตัวอย่างจำนวนไม่น้อยในเว็บไซต์พัฒนาขึ้นด้วย Python2 ในขณะที่ปัจจุบัน (ในขณะที่ทำหนังสือเล่มนี้) นักพัฒนาโปรแกรมส่วนใหญ่เริ่มปรับมาใช้ Python3 แล้ว โดย Python3 ได้รับการพัฒนาขึ้นมาใหม่และไม่คอมแพตติเบิลหรือไม่เข้ากันกับPython2 ที่มีอยู่เดิม ดังนั้น หากนำโค้ดที่เขียนขึ้นจาก Python2 มาใช้งานกับ Python3 จะต้องตรวจสอบความเข้ากันคำสั่งเสียก่อน

5.5 ตัวแปร (Variables)

ตัวแปรหรือ Variable เป็นชื่อเรียกที่เก็บข้อมูลเพื่อใช้ในโปรแกรม ผู้พัฒนาสามารถตั้งชื่อตัวแปรได้ โดยมีเงื่อนไขดังนี้

- ใช้ตัวอักษร a ถึง z และ A ถึง Z
- ใช้เครื่องหมาย _ (Underscore) ได้เท่านั้น ห้ามใช้เครื่องหมายพิเศษอื่นๆ
- ใช้ตัวเลข 0 ถึง 9
- ตัวอักษรตัวแรกของชื่อตัวแปร ห้ามใช้ตัวเลข
- ห้ามตั้งชื่อตรงกับคำสงวน

5.5.1 คำสงวน

ประกอบด้วย `false, none, true, and, as, assert, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, not, or, pass, raise, return, try, while, with` และ `yield`

ทดสอบได้ด้วยการพิมพ์คำสั่ง

```
>>> import keyword
>>> print (keyword.kwlist)
```

ผลที่ได้

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally',
'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda',
'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while',
'with', 'yield']
```

5.5.2 ตัวอย่างการสร้างตัวแปร

ตัวอย่างที่ 5-2

```
>>> age = 25
>>> print (age)
```

ผลที่ได้ : 25

จากตัวอย่างเป็นการสร้างตัวแปรที่มีชื่อว่า `age` *และเก็บข้อมูลไว้เป็นเลข 25 ดังนั้นเมื่อสั่งให้แสดงข้อมูลที่อยู่ในตัวแปร* `age` *ก็จะแสดงค่า 25 ออกมา*

ตัวอย่างที่ 5-3

```
>>> message = "Hi everyone."
>>> print (message + "I'm python developer.")
```

ผลที่ได้ : `Hi everyone. I'm python developer.`

ตัวอย่างที่ 5-4

```
>>> message1 = "Hi everyone."
>>> message2 = "I'm Python developer."
>>> message3 = "Nice to meet you."
>>> print (message1 + message2 + message3)
```

ผลที่ได้ :

Hi everyone.I'm Python developer.Nice to meet you.

จะเห็นว่า ด้วยการใช้คำสั่งแบบนี้ เมื่อแสดงผลจะไม่มีเว้นวรรคระหว่างข้อความ

ตัวอย่างที่ 5-5

```
>>> message1 = "Hi everyone."
>>> message2 = "I'm Python developer."
>>> message3 = "Nice to meet you."
>>> print (message1, message2, message3)
```

ผลที่ได้ :

Hi everyone. I'm Python developer. Nice to meet you.

จะเห็นว่า ด้วยการใช้คำสั่งแบบนี้ เมื่อแสดงผลจะเว้นวรรคระหว่างข้อความ

ตัวอย่างที่ 5-6

```
>>> width = 10.56
>>> height = 3.38
>>> area = width * height
>>> print (area)
```

ผลที่ได้ : 35.6928

เป็นการคูณค่าระหว่างตัวแปร `width` *และ* `height` *แล้วเก็บไว้ในตัวแปร* `area` *ถ้าสังเกตนี่คือ การเขียนคำสั่งเพื่อคำนวณหาพื้นที่*

5.5.3 ประเภทของตัวแปร (Primitive Data Types)

ตัวแปรในภาษา Python มี 4 ประเภทคือ

1. float (Floating-point number) : เลขทศนิยม มีค่า

2. int (integer) : เลขจำนวนเต็ม มีค่า -2^{31} ถึง $2^{31}-1$

3. long (Long integer) : เลขจำนวนเต็มขนาดใหญ่ (เก็บข้อมูลตัวเลขได้มากกว่า int) มีค่าไม่จำกัด

4. st (Character string) : ข้อความตัวอักษร

5.5.4 วิธีตรวจสอบประเภทของตัวแปรใดๆ

เนื่องจากภาษา Python ไม่จำเป็นต้องประกาศประเภทของตัวแปร แต่จะถูกกำหนดจากข้อมูล (ประเภทตัวแปรสามารถเปลี่ยนแปลงได้ตลอดเวลา) ดังนั้นหากต้องการทราบว่า ตัวแปรนั้นๆ เป็นประเภทใด จะต้องใช้คำสั่ง **type** เพื่อตรวจสอบ

ตัวอย่างที่ 5-7

```
>>> money = 2500
>>> type (money)
```

ผลที่ได้ :

```
<type 'int'>
```

แสดงประเภทหรือชนิดของตัวแปร `money` *เป็นเลขจำนวนเต็ม*

ตัวอย่างที่ 5-8

```
>>> height = 179.3
>>> type (height)
```

ผลที่ได้ :

```
<type 'float'>
```

แสดงประเภทหรือชนิดของตัวแปร `height` *เป็นเลขทศนิยม*

ตัวอย่างที่ 5-9

```
>>> milli_sec = 310200000000
>>> type (milli_sec)
```

ผลที่ได้ :

```
<type 'long'>
```

แสดงประเภทหรือชนิดของตัวแปร `milli_sec` *เป็นเลขจำนวนเต็มขนาดใหญ่*

ตัวอย่างที่ 5-10

```
>>> name = "INEX"
>>> type (name)
```

ผลที่ได้ :

```
<type 'str'>
```

แสดงประเภทหรือชนิดของตัวแปร name เป็นข้อความ

ในกรณีที่ต้องการตรวจสอบว่า ตัวแปรใดๆ เป็นประเภทเดียวกับที่ต้องการตรวจสอบหรือไม่ จะต้องใช้คำสั่ง type ในลักษณะตามตัวอย่างต่อไปนี้

ตัวอย่างที่ 5-11

```
>>> numberOfDay = 301
>>> type (numberOfDay) is int
```

ผลที่ได้ : True

เป็นการตรวจสอบว่า ค่าในตัวแปร `numberOfDay` *เป็นตัวแปรแบบเลขจำนวนเต็ม (int) หรือไม่ จากการกำหนดค่าตัวแปร พบว่า ตัวแปร* `numberOfDay` *มีการกำหนดค่าเป็น 301 ดังนั้น จึงเป็นตัวแปรชนิดเลขจำนวนเต็ม ผลลัพธ์ที่ได้จึงเป็นจริงหรือ True*

ตัวอย่างที่ 5-12

```
>>> numberOfDay = 301
>>> type (numberOfDay) is str
```

ผลที่ได้ : False

จากตัวอย่างนี้เป็นการตรวจสอบว่า ค่าในตัวแปร `numberOfDay` *เป็นตัวแปรแบบข้อความ (st) หรือไม่ จากการกำหนดค่าตัวแปร* `numberOfDay` *พบว่า ตัวแปรนี้เป็นแบบเลขจำนวนเต็ม ดังนั้น ผลลัพธ์ที่ได้จึงเป็นเท็จหรือ False*

5.6 การรับค่าจากคีย์บอร์ด (Keyboard Input)

นอกเหนือจากการกำหนดข้อมูลให้กับตัวแปรโดยตรงแล้ว ใน Python ยังกำหนดชนิดข้อมูลให้กับข้อมูลที่รับมาจากคีย์บอร์ดได้ด้วย โดยใช้คำสั่ง `input` เก็บข้อมูลเป็น int, long หรือ float เท่านั้น

ตัวอย่างที่ 5-13

```
>>> age = input ("How old are you? : ")
How old are you : 31
>>> print (age)
31
>>> type (age)
<type 'int'>
```

เป็นการใช้คำสั่ง `input` *ในการรับค่าตัวเลข แล้วนำไปเก็บไว้ในตัวแปร* `age` *แล้วทำการพิมพ์ค่าที่ได้ออกมาและชนิดของตัวแปร*

ตัวอย่างที่ 5-14

```
>>> weight = input ("Weight : ")
weight : 75.3
>>> print (weight)
75.3
>>> type (weight)
<type 'float'>
```

ตัวอย่างที่ 5-15

```
>>> height = input ("Height : ")
Height = 10
>>> width = input ("Widht : ")
Widht = 20
>>> print ("Area =", width * height)
```

ในกรณีที่ต้องการรับข้อมูลจากคีย์บอร์ดเป็นแบบ str จะใช้คำสั่ง `raw_input`

ตัวอย่างที่ 5-16

```
>>> name = raw_input ("What's your name? : ")
What's your name? : Python
>>> print (name)
Python
>>> type (name)
<type="str">
```

จากตัวอย่างเป็นการรับข้อความเพื่อเก็บไว้ในตัวแปร name จากนั้นทำการพิมพ์แสดงข้อมูลของตัวแปร `name` *และชนิดของตัวแปร นั่นคือ str โดยมีข้อความ Python เก็บไว้ในตัวแปร* `name`

5.7 เครื่องหมายคำนวณทางคณิตศาสตร์พื้นฐาน (Math Operators)

ประกอบด้วย

+	Addition	บวก
-	Subtraction	ลบ
*	Multiplication	คูณ
/	Division	หาร (ส่วน)
%	Modulus	หารเอาเศษ
//	Floor division	หาร ปัดเศษ
**	Exponentiation	ยกกำลัง

ตัวอย่างที่ 5-17

`x += 4`	มีค่าเท่ากับ	x = x + 4
`x *= 3`	มีค่าเท่ากับ	x = x * 3
`x /= 6`	มีค่าเท่ากับ	x = x / 6
`x **= 2`	มีค่าเท่ากับ	x = x ** 2
`x %= 5`	มีค่าเท่ากับ	x = x % 5

5.8 List และ Tuple

5.8.1 List

List เป็นตัวแปรที่อยู่ในรูปของอะเรย์ มีหน้าที่เก็บข้อมูลจำนวนหลายๆ ตัว

5.8.1.1 การสร้างตัวแปรแบบ List

การสร้างตัวแปรแบบ List ทำได้ง่ายๆ ดังตัวอย่าง

ตัวอย่างที่ 5-18

```
>>> my_list = [15, 12, 17, 13]
>>> print (my_list)
```

ผลลัพธ์ที่ได้ : [15, 12, 17, 13]

จากตัวอย่างนี้ ตัวแปร `my_list` *มีสมาชิก 4 ตัวคือ 15, 12, 17 และ 13*

ตัวอย่างที่ 5-19

```
>>> my_list = [15, 12, 17, 13]
>>> type (my_list)
```

ผลลัพธ์ที่ได้ : `<type 'list'>` *จากตัวอย่างนี้เป็นการตรวจสอบชนิดของตัวแปร* `my_list` *ซึ่งได้ผลเป็นตัวแปรแบบ List*

5.8.1.2 วิธีการเรียกใช้งานข้อมูลที่อยู่ใน List

ทำได้ด้วยการระบุตำแหน่งของสมาชิกในตัวแปรแบบ list โดยตำแหน่งของตัวแปรอาจเรียกว่า เลขดัชนี โดยมีค่าตั้งแต่ 0 เริ่มจากสมาชิกที่อยู่ซ้ายสุด

ตัวอย่างที่ 5-20

```
>>> data = [30, 15, 20, 60, 5, 100]
>>> print (data[3])
```

ผลลัพธ์ที่ได้ : 60 เป็นการสั่งให้แสดงค่าของสมาชิกลำดับ 3 ในตัวแปร `data` *ซึ่งเป็นตัวแปรแบบ list ซึ่งก็คือค่า 60 โดยลำดับ 0 คือ 30, ลำดับ 1 คือ 15, ลำดับ 2 คือ 20, ลำดับ 4 คือ 5 และลำดับ 5 สุดท้ายมีค่า 100*

ตัวอย่างที่ 5-21

```
>>> names = ["Ivy", "Dan", "Thomas"]
>>> person = names[1]
>>> print (person)
```

ผลลัพธ์ที่ได้ : Dan จากตัวอย่างนี้เป็นการสั่งให้แสดงชื่อของสมาชิกลำดับ 1 ในตัวแปร `names` *ซึ่งเป็นตัวแปรแบบ str ซึ่งก็คือ ข้อความ Dan แล้วนำเก็บไว้ในตัวแปร* `person` *จากนั้นแสดงค่าของตัวแปร* `person` *ด้วยคำสั่ง* `print`

ข้อควรระวัง - กรณีที่ระบุตำแหน่งของข้อมูลเกิน จะทำให้โปรแกรมทำงานผิดพลาดได้ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 5-22

```
>>> numbers = [10, 20, 30]
>>> print (number[5])
```

ผลลัพธ์ที่ได้

```
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
Index Error: list index out of range
```

จากตัวอย่างแสดงให้เห็นการทำงานที่ผิดพลาด เนื่องจากกำหนดขอบเขตของตำแหน่งหรือเลขดัชนีของตัวแปรเกินขอบเขตที่กำหนดไว้

5.8.1.3 การเรียกใช้งานข้อมูลหลายตัวพร้อมกัน

ผู้พัฒนาโปรแกรมสามารถเรียกใช้งานข้อมูลของตัวแปร List ได้หลายตัวพร้อมกัน มีรูปแบบเป็น

ชื่อตัวแปร[X : Y]

โดยที่

X คือลำดับของข้อมูลตัวแรกสุดที่ต้องการเรียกใช้งาน

Y คือลำดับของข้อมูลที่ต้องการสิ้นสุด

ตัวอย่างที่ 5-23

```
>>> letters = ["Ant", "Bird", "Cat", "Dog", "Egg", "Fish"]
>>> new_letters = letters[1 : 5]
>>> print (new_letters]
```

ผลลัพธ์ที่ได้ : **['Bird', 'Cat', 'Dog', 'Egg']**

จากตัวอย่างเป็นการเรียกข้อมูลในลำดับ 1 ไปจนถึงข้อมูลลำดับ 4 เนื่องจากกำหนดลำดับที่สิ้นสุดไว้ที่ลำดับ 5 ดังนั้นจากคำสั่งในตัวอย่างนี้จึงเรียกใช้งานข้อมูลลำดับ 1 ถึง 4 ของตัวแปร `new_letters`

ในกรณีที่ระบุลำดับของข้อมูลที่ต้องการสิ้นสุดเกินจำนวนที่มีในตัวแปรชนิด List ตัวนั้นๆ โปรแกรมยังคงทำงานได้ปกติ แต่ข้อมูลที่ได้ก็จะเป็นข้อมูลเท่าที่มี เพราะไม่สามารถเรียกข้อมูลในลำดับที่เกินออกมาใช้งานได้

ตัวอย่างที่ 5-24

```
>>> letters = ["Ant", "Bird", "Cat", "Dog", "Egg", "Fish"]
>>> new_letters = letters[2 : 10]
>>> print (new_letters]
```

ผลลัพธ์ที่ได้ :

```
['Cat', 'Dog', 'Egg', 'Fish']
```

จากตัวอย่างนี้ ตัวแปร `new_letters` *เป็นตัวแปรชนิด List ที่มีสมาชิก 6 ตัว แต่มีการเรียกข้อมูลของสมาชิกมาใช้งานตั้งแต่ลำดับ 2 ไปถึง 9 (เนื่องจากกำหนดขอบเขตสิ้นสุดไว้ที่ลำดับ 10) ซึ่งเกินจากจำนวนของสมาชิกที่มีอยู่ ดังนั้นในการแสดงผลจึงแสดงข้อมูลของสมาชิกเท่าที่มี*

5.8.1.4 การแก้ไขข้อมูลที่อยู่ใน List

ด้วยความยืดหยุ่นของภาษา Python ทำให้การแก้ไขข้อมูลของสมาชิกในตัวแปรชนิด List ทำได้ง่ายมาก เพียงระบุลำดับของสมาชิกแล้วกำหนดค่าหรือข้อมูลที่ต้องการแก้ไข ข้อมูลของสมาชิกในตัวแปรก็จะเปลี่ยนแปลงได้ทันที

ตัวอย่างที่ 5-25

```
>>> days = ["Monday", "Tuesday", "Saturday"]
>>> days[1] = "Friday"
>>> print (days)
```

ผลลัพธ์ที่ได้ :

```
['Monday', 'Friday', 'Saturday']
```

จากตัวอย่าง ต้องการแก้ไขข้อมูลในลำดับ 1 จากเดิมคือ Tuesday เปลี่ยนเป็น Friday ทำได้ง่ายตามที่แสดงในบรรทัด `>>> days[1] = "Friday"` *เลข 1 คือ ลำดับ 1 ของตัวแปร* `days` *ซึ่งก็คือ* `Tuesday` *เมื่อรันคำสั่ง* `days[1] = "Friday"` *แล้ว ค่าข้อมูลในลำดับ 1 จะเปลี่ยนเป็น* `Friday` *ตามที่แสดงด้วยคำสั่ง* `print`

5.8.1.5 การรองรับข้อมูลต่างประเภทของตัวแปร List

ข้อมูลที่เก็บไว้ในตัวแปรชนิด List ไม่จำเป็นต้องเป็นประเภทเดียวกันก็ได้ เพราะ List นั้นรองรับข้อมูลต่างประเภทได้ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 5-26

```
>>> data = [162, 91.4, "Wording"]
>>> print (data)
```

ผลลัพธ์ที่ได้ :

```
[162, 91.4, 'Wording']
```

จากตัวอย่างนี้ ตัวแปร `data` *ประกอบด้วยข้อมูลที่เป็นเลขจำนวนเต็ม (int) คือ 162, เลขทศนิยม (float) คือ 91.4 และตัวอักษร (str) คือ Wording*

หากต้องการตรวจสอบประเภทของข้อมูลแต่ละตัวในตัวแปรชนิด List ก็ทำได้ด้วยคำสั่ง **type** ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 5-27

```
>>> data = [162, 91.4, "Wording"]
>>> type (data[0])
```

ผลลัพธ์ที่ได้ :

```
<type 'int'>
```

จากตัวอย่างนี้ต้องการตรวจสอบประเภทของข้อมูลในลำดับ 0 ของตัวแปร `data` *นั่นคือ 162 เป็นข้อมูลประเภท int*

ตัวอย่างที่ 5-28

```
>>> data = [162, 91.4, "Wording"]
>>> type (data[1])
```

ผลลัพธ์ที่ได้ :

```
<type 'float'>
```

จากตัวอย่างนี้ต้องการตรวจสอบประเภทของข้อมูลในลำดับ 1 ของตัวแปร `data` *นั่นคือ 91.4 เป็นข้อมูลประเภท float*

ตัวอย่างที่ 5-29

```
>>> data = [162, 91.4, "Wording"]
>>> type (data[2])
```

ผลลัพธ์ที่ได้ :

```
<type 'str>
```

จากตัวอย่างนี้เป็นตรวจสอบประเภทของข้อมูลในลำดับ 2 ของตัวแปร `data` *นั่นคือ Wording พบว่าเป็นข้อมูลประเภทตัวอักษรหรือ str*

5.8.1.6 ชุดข้อมูลอะเรย์หลายมิติ

ตัวแปรชนิด List รองรับการกำหนดค่าในแบบชุดข้อมูลอะเรย์หลายมิติได้

ตัวอย่างที่ 5-30

```
>>> matrix = [[1, 2, 3], [35, 50, 71], [5, 25, 125]]
>>> print (matrix[0][2])
```

ผลลัพธ์ที่ได้ : 3 ในตัวอย่างนี้กำหนดให้ตัวแปร `matrix` *เป็นตัวแปรอะเรย์ 3x3 และเลือกแสดงค่าของข้อมูลในตำแหน่ง [0,2] ซึ่งก็คือ เลข 3*

5.8.2 Tuple

Tuple เป็นตัวแปรที่อยู่ในรูปของอะเรย์ที่มีหน้าที่เก็บข้อมูลจำนวนหลายๆ ตัว มีคุณสมบัติคล้ายกับ List แต่ไม่สามารถแก้ไขข้อมูลที่อยู่ใน Tuple ได้

5.8.2.1 การสร้างตัวแปรแบบ Tuple

การสร้างตัวแปรแบบ Tuple เหมือนกับตัวแปรชนิด List เพียงเปลี่ยนสัญลักษณ์ขอบเขตของข้อมูลจาก [] เป็น () ดังตัวอย่าง

ตัวอย่างที่ 5-31

```
>>> my_tuple = (2, 4, 5, 10)
>>> print (my_tuple)
```

ผลลัพธ์ที่ได้ :

```
(2, 4, 5, 10)
```

ในตัวอย่างนี้ตัวแปร `my_tuple` *มีสมาชิก 4 ตัวคือ 2, 4, 5 และ 10*

ตัวอย่างที่ 5-32

```
>>> my_tuple = (2, 4, 5, 10)
>>> type (my_tuple)
```

ผลลัพธ์ที่ได้ :

```
<type 'tuple'>
```

ในตัวอย่างนี้เป็นการตรวจสอบชนิดของข้อมูลในตัวแปร `my_tuple`

5.8.2.2 การแก้ไขข้อมูลที่อยู่ใน Tuple

การแก้ไขข้อมูลของสมาชิกในตัวแปรชนิด Tuple ทำไม่ได้ เว้นแต่ว่าจะทำการกำหนดค่าใหม่ทั้งหมด

ตัวอย่างที่ 5-33 (กำหนดค่าใหม่ทั้งหมด)

```
>>> tuples = 2, 4, 6, 8, 10, 12
>>> tuples = 1, 3, 5
>>> print (tuples)
```

ผลลัพธ์ที่ได้ :

```
(1, 3, 5)
```

ในตัวอย่างนี้ ช่วงแรกทำการกำหนดค่าของตัวแปร `tuples` *ไว้อย่างหนึ่ง เนื่องจากเข้าไปแก้ไขค่าในตัวแปรชนิด Tuple ไม่ได้ จึงต้องใช้การกำหนดค่าใหม่ดังคำสั่งบรรทัด* `>>> tuples = 1, 3, 5`

ตัวอย่างที่ 5-34 (แก้ไขข้อมูลใน Tuple ไม่ได้)

```
>>> tuples = 2, 4, 6, 8, 10
>>> tuples[1] = 5
```

ผลลัพธ์ที่ได้ :

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name '5' is not defined
```

ในตัวอย่างนี้ ต้องการเปลี่ยนค่าของข้อมูลลำดับ 1 ของตัวแปร `tuples` *จะพบว่า* ***ทำไม่ได้*** *จึงเกิดการแจ้งความผิดพลาดขึ้นมาให้ทราบ*

5.9 การทำงานแบบมีเงื่อนไข (Condition Operation)

ในการเขียนโปรแกรมทุกภาษาจะต้องมีการตัดสินใจ เลือกกระทำคำสั่งตามเงื่อนไขที่กำหนด เพื่อให้เกิดการทำงานของโปรแกรมที่มีความซับซ้อนสูงมากขึ้น ในภาษา Python มีการทำงานแบบมีเงื่อนไขอยู่ 3 แบบหลักๆ คือ **`if`**, **`for`** และ **`while`**

สิ่งหนึ่งที่ต้องทราบเป็นความรู้พื้นฐานของการทำงานแบบมีเงื่อนไขคือ เครื่องหมายที่ใช้ในการเปรียบเทียบค่า ประกอบด้วย

==	เท่ากับ
!=	ไม่เท่ากับ
<>	ไม่เท่ากับ
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ

5.9.1 การทำงานแบบมีเงื่อนไขด้วยกลุ่มคำสั่ง `if`

5.9.1.1 `if`

เป็นคำสั่งสำหรับกำหนดให้ตัดสินใจเลือกทำงาน เมื่อเงื่อนไขตรงตามที่กำหนดไว้

รูปแบบ

`if` (เงื่อนไข):

คำสั่ง 1

คำสั่ง 2

ในภาษา Python การเขียนบล็อกของคำสั่งเพื่อควบคุมการทำงานตามเงื่อนไข จะไม่ใช้เครื่องหมาย {} (ปีกกาเปิดปิด) แต่จะกำหนดช่วงด้วยการย่อหน้าโดยกดปุ่ม Tab แทน

ตัวอย่างที่ 5-35

```
x = 30
if (x > 6):
   print ("Value is more than 6")
```

ผลลัพธ์ที่ได้ : **`Value is more than 6`**

ตัวอย่างที่ 5-36

```
x = 67
if (x > 10):
   print ("More than 10")
if (x > 40):
   print ("More than 40")
if (x > 70):
   print ("More than 70")
if (x > 100):
   print ("More than 100")
```

ผลลัพธ์ที่ได้ :

More than 10

More than 40

ในตัวอย่างนี้เป็นการตรวจสอบว่า ค่าของตัวแปร x อยู่ในช่วงใด และพิมพ์แสดงข้อความของช่วงนั้นๆ ออกมา จากตัวอย่างตัวแปร x มีค่า 67 ดังนั้นเงื่อนไขที่ตรงมี 2 เงื่อนไขคือ

```
if (x > 10):
   print ("More than 10")
```

และ

```
if (x > 40):
   print ("More than 40")
```

จึงแสดงข้อความ **More than 10** *และ* **More than 40**

จะเห็นว่า เมื่อเงื่อนไขที่อยู่ใน `if` นั้นถูกต้อง คำสั่งที่อยู่ใน `if` จะทำงานโดยทันที และในคำสั่ง `if` แต่ละตัวรองรับคำสั่งได้มากกว่าหนึ่งคำสั่ง ยกตัวอย่าง

ตัวอย่างที่ 5-37

```
y = 5
if (y > 2):
   print ("Value is more than 2")
y = y + 10
print ("Result=", y)
```

ผลลัพธ์ที่ได้ :

Value is more than 2

Result = 15

ในตัวอย่างนี้ แสดงให้เห็นถึงการกระทำคำสั่งภายใต้ `if` *หากค่าของ* `y` *มากกว่า 2 จะแสดงข้อความ* `Value is more than 2` *และทำการคำนวณต่อ นั่นคือ* `y = y + 10` *จากนั้นจึงพิมพ์ค่าของตัวแปรแสดงออกมา*

ตัวอย่างที่ 5-38

```
y = 5
if (y < 2):
   print ("Value is less than 2")
   y = y + 10
print ("Result=", y)
```

ผลลัพธ์ที่ได้ : **Result = 5**

ในตัวอย่างนี้ แสดงให้เห็นถึงการทำงานเมื่อเงื่อนไขของคำสั่ง `if` *ไม่เป็นจริง ก็จะไม่มีการทำงานในบล็อกของคำสั่ง* `if` *โปรแกรมจะกระโดดข้ามมายังคำสั่ง* `print ("Result=", y)` *เพื่อแสดงผลค่าของ y ทันที จึงได้ผลลัพธ์เป็น Result = 5*

5.9.1.2 else

else เป็นคำสั่งที่ทำงานแบบมีเงื่อนไขนอกเหนือจากที่กำหนดไว้ใน `if` จะต้องใช้คู่กับ `if` เสมอ และไม่สามารถใช้งานซ้ำในเงื่อนไขชุดนั้นๆ ได้

รูปแบบ

if (เงื่อนไข):

คำสั่ง 1

else:

คำสั่ง 2

จะเห็นว่า `else` ไม่จำเป็นต้องกำหนดเงื่อนไขใดๆ เพราะ `else` จะทำงานก็ต่อเมื่อมีการกำหนดเงื่อนไขอื่นนอกเหนือไปจากที่กำหนดไว้ในคำสั่ง `if`

ตัวอย่างที่ 5-39

```
x = 15
if (x < 5):
   print ("Less than 5")
else:
   print ("Equal or more than 5")
```

ผลลัพธ์ที่ได้ : **Equal or more than 5**

ในตัวอย่างนี้ กำหนดเงื่อนไขแรกที่คำสั่ง `if` *ว่า ถ้า* `x` *น้อยกว่า 5 ให้พิมพ์ข้อความ Less than 5 จากนั้นจะมีการทดสอบเงื่อนไขเพิ่มด้วยคำสั่ง* `else` *ว่า ถ้า* `x` *ไม่น้อยกว่า 5 ให้พิมพ์ข้อความ Equal or more than 5*

else จะใช้ต่อท้ายคำสั่ง if ได้เพียง 1 ครั้งเท่านั้น แต่ใช้งานร่วมกับคำสั่ง if หลายๆ ตัวได้

ตัวอย่างที่ 5-40

```
x = 0
if (x > 1):
   print ("More than 1")
else:
   print ("Equal or less than 1")
if (x < 4):
   print ("Less than 4")
else:
   print ("Equal or more than 4")
```

ผลลัพธ์ที่ได้

```
Equal or less than 1
Less than 4
```

ในตัวอย่างนี้ มีการใช้คำสั่ง if รวม 2 ช่วง

ช่วงแรกตรวจสอบว่า x มากกว่า 1 หรือไม่ และอีกช่วงคือ x น้อยกว่า 4 หรือไม่

ในช่วงแรก จะตรวจสอบว่า x มากกว่า 1 หรือไม่ ถ้าใช่ จะพิมพ์ข้อความ More than 1 ถ้าไม่ใช่ คำสั่ง else จะเข้ามารับช่วงการทำงานต่อ เพื่อกำหนดให้แสดงข้อความ Equal or less than 1 เช่นเดียวกับในช่วงที่ 2 ที่ตรวจสอบว่า x น้อยกว่า 4 หรือไม่ ถ้าใช่ จะแสดงข้อความ Less than 4 หากไม่ใช่ คำสั่ง else จะกำหนดให้ทำงานต่อเพื่อแสดงข้อความ Equal or more than 4

จะเห็นว่า คำสั่ง else จะใช้งานต่อจากคำสั่ง if เสมอ และถ้ามีคำสั่ง if กี่ชุด คำสั่ง else ก็จะนำมาใช้งานได้ตามจำนวนของคำสั่ง if ที่ปรากฏในโปรแกรม

ตัวอย่างที่ 5-41

```
y = 1
if (y == 0):
   print ("Equal 0")
else :
   print ("Not Equal 0")
else :
   print ("Not Equal 0 (again)")
```

ผลลัพธ์ที่ได้

```
pi@raspberrypi ~ $ sudo python ifelse.py
  File "ifelse.py", line 4
    else
       ^
SyntaxError: invalid syntax
```

จากตัวอย่างนี้ จะเห็นว่าไม่อาจใช้คำสั่ง else ได้มากกว่า 1 ครั้งต่อการใช้คำสั่ง if หนึ่งชุด

5.9.1.3 else-if

เนื่องจาก `if` เป็นคำสั่งใช้ตรวจสอบเงื่อนไขตามที่กำหนดไว้ เมื่อมีการใช้งาน `if` จำนวนมาก อาจทำให้เกิดการทำงานภายในคำสั่ง `if` แต่ละตัวซ้ำซ้อนกันได้ เช่น

```
x = 2
if (x > 5):
   print ("Result >", 5)
if (x > 4):
   print ("Result >", 4)
if (x > 3):
   print ("Result >", 3)
if (x > 2):
   print ("Result >", 2)
if (x > 1):
   print ("Result >", 1)
```

ผลลัพธ์ที่ได้

```
Result > 5
Result > 4
Result > 3
```

จะเห็นว่ามี 3 เงื่อนไขที่ถูกต้อง จึงทำให้คำสั่งที่อยู่ในทั้ง 3 เงื่อนไขนั้นทำงาน

ทำให้คำสั่ง `else-if` (เวลาใช้งานจริงจะเหลือเพียง `elif`) ถูกนำมาใช้งาน เพื่อให้โปรแกรมของการตรวจสอบเงื่อนไขในกลุ่มนั้นๆ เลือกทำงาน "เงื่อนไขใดเงื่อนไขหนึ่ง" เท่านั้น

รูปแบบ

if (เงื่อนไข 1):
คำสั่ง 1
elif (เงื่อนไข 2):
คำสั่ง 2
elif (เงื่อนไข 3):
คำสั่ง 3

กรณีใช้งานร่วมกับคำสั่ง else

if (เงื่อนไข 1):
คำสั่ง 1
elif (เงื่อนไข 2):
คำสั่ง 2
elif (เงื่อนไข 3):
คำสั่ง 3
else
คำสั่งนอกเหนือจากทุกเงื่อนไข

การใช้คำสั่ง else-if เหมือนกับ else คือ ต้องใช้งานร่วมกับif เท่านั้น แต่ else-if เรียกใช้ได้หลายตัวภายใต้คำสั่ง if ชุดเดียวกัน

ตัวอย่างที่ 5-42

```
x = 2
if (x > 5):
   print ("Result >", 5)
elif (x > 4):
   print ("Result >", 4)
elif (x > 3):
   print ("Result >", 3)
elif (x > 2):
   print ("Result >", 2)
elif (x > 1):
   print ("Result >", 1)
```

ผลลัพธ์ที่ได้ :

```
Result > 1
```

การทำงานของตัวอย่างนี้คือ โปรแกรมจะตรวจสอบว่า x มากกว่า 5 หรือไม่ ถ้าใช่ ก็ทำการแสดงข้อความ Result > 5 แล้วออกจากบล็อกคำสั่งของ if แต่ถ้าไม่ใช่จะตรวจสอบต่อว่า มากกว่า 4, มากกว่า 3, มากกว่า 2 และมากกว่า 1 ด้วยคำสั่ง elif จากตัวอย่างค่าของ x คือ 2 จึงตรงกับเงื่อนไขสุดท้ายที่ว่า elif (x > 1) จึงเข้าไปทำคำสั่งในบล็อก นั่นคือ print ("Result >", 1) เป็นการแสดงข้อความ Result > 1

จากโปรแกรมตัวอย่างที่ 5-42 เมื่อเงื่อนไขใดก็ตามเกิดเป็นจริงและมีการกระทำคำสั่งภายใต้เงื่อนไขนั้นๆ ไปแล้ว โปรแกรมจะออกจากการตรวจสอบด้วยคำสั่ง if ในชุดนั้นๆ ทันที แม้ว่า จะมีเงื่อนไขอื่นๆ ที่ถูกต้องรวมอยู่ก็ตาม

5.9.2 การทำงานวนรอบแบบมีเงื่อนไขด้วยคำสั่ง for

for เป็นการวนการทำงานวนลูปแบบมีเงื่อนไข โดยจะวนทำงานของคำสั่งชุดเดิมๆ ไปเรื่อยๆ แต่จะมีจำนวนครั้งในการวนที่เจาะจง จึงเหมาะสำหรับการกำหนดรอบของการวนทำงานที่ทราบจำนวนครั้งที่แน่นอน

รูปแบบ

```
for    ตัวแปร  in   ชุดข้อมูล:
   คำสั่ง 1
   คำสั่ง 2
```

ตัวอย่างที่ 5-43

```
for number in [1, 3, 5, 7, 9, 11]:
   print (number)
```

ผลลัพธ์ที่ได้ :

```
1
3
5
7
9
11
```

จากตัวอย่างนี้ จะเห็นว่า คำสั่ง `for` *จะวนทำงาน 6 ครั้ง ตามจำนวนข้อมูลที่มีในตัวแปร* `number` *ที่เป็นตัวแปรแบบ List นี่คืออีกหนึ่งข้อดีของ Python ที่สามารถนำตัวแปร List หรือ Tuple มาใช้ในคำสั่ง* `for` *ได้*

ถ้าต้องการวนทำงานโดยมีค่าเป็นชุดตัวเลขที่เรียงลำดับอย่างแน่นอน ผู้พัฒนาโปรแกรมสามารถใช้คำสั่ง `range` เข้ามาช่วยได้

ตัวอย่างที่ 5-44

```
for x in range (4):
   print (x)
```

ผลลัพธ์ที่ได้

```
0
1
2
3
```

เลข 4 ที่กำหนดลงไปใน `range` *คือจำนวนครั้ง โดยค่าตัวเลขที่ได้จะเริ่มตั้งแต่ 0 และนับไปจนครบ 4 ครั้ง จึงเป็น 0, 1, 2 และ 3*

ในกรณีที่ต้องการกำหนดเลขเริ่มต้นด้วย ก็ทำการกำหนดค่าในคำสั่ง range ได้เลย

ตัวอย่างที่ 5-45

```
for x in range (5, 10):
   print (x)
```

ผลลัพธ์ที่ได้ :

```
5
6
7
8
9
```

เลข 5 ที่กำหนดลงไปในคำสั่ง range คือ เลขเริ่มต้น และเลข 10 คือ ขอบเขตสิ้นสุด

การกำหนดจำนวนการวนทำงานของคำสั่ง for โดยใช้คำสั่ง range เข้ามาช่วย ยังมีอีกหนึ่งความสามารถที่น่าสนใจคือ **กำหนดค่าด้วยผลต่างของเลขตัวถัดไป** ดังตัวอย่างที่ 5-46

ตัวอย่างที่ 5-46

```
for x in range (2, 20, 3):
   print (x)
```

ผลลัพธ์ที่ได้ :

```
2
5
8
11
14
17
```

จากตัวอย่างนี้ที่คำสั่ง range เป็นการกำหนดให้เริ่มต้นที่เลข 2 และสิ้นสุดที่ 20 โดยเพิ่มค่าขึ้นครั้งละ เมื่อโปรแกรมนี้ทำงานจริง จะสังเกตเห็นว่า คำสั่ง for จะทำงานจนถึงค่าที่เป็น 17 ไม่ใช่ 20 เนื่องจากเลข 20 เป็นขอบเขต จึงไม่อาจนำมาประมวลผลได้

5.9.3 การทำงานวนรอบแบบมีเงื่อนไขด้วยคำสั่ง while

while เป็นการวนการทำงานหรือวนลูปแบบมีเงื่อนไขว่า จะวนทำงานคำสั่งชุดเดิมไปเรื่อยๆ จนกว่าเงื่อนไขที่กำหนดไว้จะไม่เป็นจริง โดยจะทำการตรวจสอบเงื่อนไขก่อน แล้วจึงทำงานตามชุดคำสั่ง

รูปแบบ

while (เงื่อนไข):

คำสั่ง 1

คำสั่ง 2

คำสั่ง 3

ตัวอย่างที่ 5-47

```
count = 0
while (count < 20):
   count += 3
   print ("Count =", count)
```

ผลลัพธ์ที่ได้

```
Count = 3
Count = 6
Count = 9
Count = 12
Count = 15
Count = 18
Count = 21
```

จากตัวอย่างนี้ ตัวแปร `count` *เพิ่มค่าขึ้นครั้งละ 3 เมื่อมีค่าน้อยกว่า 20 เมื่อวนทำงานไปเรื่อยๆ จนถึงเมื่อตัวแปร* `count` *มีค่าเป็น 18 ซึ่งยังน้อยกว่า 20 ทำให้ต้องเพิ่มค่าอีก 3 กลายเป็น 21 และทำให้ไม่ตรงกับเงื่อนไข จึงทำให้ชุดคำสั่งที่อยู่ในลูป* `while` *หยุดทำงานทันที*

ในกรณีที่ต้องการออกจากการทำงานในลูป while กลางคัน ให้ใช้คำสั่ง break

ตัวอย่างที่ 5-48

```
import random
count = 0
while (1):
   count += 1
   x = random.randint (1, 10)
   print ("Random", x)
   if (x == 10):
      break;
      print ("Random count", count)
```

ผลลัพธ์ที่ได้

```
Random 6
Random 4
Random 2
Random 10
Random Count 4
```

จากตัวอย่างเป็นการสุ่มตัวเลขระหว่าง 1 ถึง 10 ขึ้นมา แล้วแสดงผลออกทางหน้าจอ ในระหว่างการสุ่มแต่ละครั้งจะมีการนับด้วยว่า สุ่มไปกี่ครั้ง โดยเก็บไว้ในตัวแปร count *เมื่อสุ่มได้เลข 10 ก็จะทำการออกจากลูป* while *ในทันที แล้วแสดงจำนวนครั้งที่ทำการสุ่ม จากโปรแกรมตัวอย่างจำนวนครั้งในการสุ่มคือ 4 จากข้อความแสดงผลตัวสุดท้าย Random Count 4*

5.10 ฟังก์ชั่นของ Python

ฟังก์ชั่นถือเป็นส่วนสำคัญของภาษาคอมพิวเตอร์ทุกภาษาที่ขาดไปไม่ได้ เพราะเป็นเสมือนชุดคำสั่งที่เตรียมพร้อมสำหรับใช้งาน เหมาะสำหรับการเรียกชุดคำสั่งเดิมๆ ซ้ำ ช่วยลดความซับซ้อนของคำสั่งในโปรแกรม อีกทั้งยังช่วยให้คำสั่งในโปรแกรมดูเข้าใจได้ง่ายขึ้น

รูปแบบฟังก์ชั่นในโปรแกรมภาษา Python เป็นดังนี้

def ชื่อฟังก์ชั่น (พารามิเตอร์):

คำสั่ง

ตัวอย่างที่ 5-49

```
def print_ok ():
   print ("OK")
print_ok()
print_ok()
print_ok()
```

ผลลัพธ์ที่ได้ :

OK

OK

OK

การสร้างฟังก์ชั่นในภาษา Python มีเงื่อนไขหลักคือ จะต้องสร้างการเรียกใช้งานฟังก์ชั่นนั้นๆ ด้วย เพราะว่า ภาษา Python ทำงานแบบอินเตอร์พรีตเตอร์ จึงไม่อาจประกาศไว้หลังจากคำสั่งเรียกใช้งานได้

นอกจากนั้น ฟังก์ชั่นในภาษา Python ยังรองรับการกำหนดพารามิเตอร์เพื่อกำหนดข้อมูลตัวแปรที่นำมาใช้งานภายในโปรแกรมได้ด้วย

ตัวอย่างที่ 5-50

```
def get_triangle_area (width, height):
   area = width * height / 2
   print ("Area =", area)
get_triangle_area (100, 30)
```

ผลลัพธ์ที่ได้ : Area = 1500

ในตัวอย่างนี้เป็นการสร้างฟังก์ชั่นหาพื้นที่สามเหลี่ยม ชื่อฟังก์ชั่นคือ `get_triangle_area` *มีพารามิเตอร์ 2 ตัวคือ* `width` *ความกว้างของฐาน และ* `height` *คือ ความสูงของรูปสามเหลี่ยม*

ในกรณีที่ต้องการส่งค่าผลลัพธ์ที่ได้จากฟังก์ชั่น เรียกใช้งานคำสั่ง **Return** ได้เหมือนภาษาอื่นๆ

ตัวอย่างที่ 5-51

```
def get_triangle_area (width, height):
   area = width * height / 2
   return (area)
result = get_triangle_area (50, 12)
print ("Result Area =", result)
```

ผลลัพธ์ที่ได้ : Result Area = 300

ในตัวอย่างนี้เป็นการสร้างฟังก์ชั่นหาพื้นที่สามเหลี่ยม ชื่อฟังก์ชั่นคือ `get_triangle_area` *โดยต่อยอดมาจากตัวอย่างที่ 5-50 นั่นคือ เพิ่มคำสั่ง* `return (area)` *เข้ามา เพื่อทำให้ฟังก์ชั่นสามารถส่งค่ากลับไปได้ จากนั้นจึงนำค่าผลลัพธ์ของฟังก์ชั่นไปเก็บไว้ในตัวแปร แล้วทำการแสดงผลด้วยคำสั่ง* `print` *ต่อไป*

นอกจากนี้ภาษา Python ยังมีความพิเศษตรงที่ กำหนดค่าเริ่มต้นให้กับพารามิเตอร์ในฟังก์ชั่นได้ด้วย จึงทำให้ผู้พัฒนาโปรแกรมเรียกใช้งานฟังก์ชั่นนั้นๆ ได้โดยกำหนดค่าพารามิเตอร์เหล่านั้นหรือไม่ก็ได้

ตัวอย่างที่ 5-52

```
def get_square_area (width = 30, height = 5)
   return width * height
result1 = get_square_area ()
print ("1st Result =", result1)
result2 = get_square_area (10)
print ("2nd Result =", result2)
result3 = get_square_area (12, 15)
print ("3rd Result =", result3)
```

ผลลัพธ์ที่ได้ :

```
1st Result = 150
2nd Result = 50
3rd Result = 180
```

ในตัวอย่างนี้ กำหนดให้ค่าเริ่มต้นของพารามิเตอร์คือ `width = 30` *และ* `height =5` *เมื่อเรียกใช้ฟังก์ชั่น* `get_square_area ()` *ในแบบไม่กำหนดพารามิเตอร์ มันจะดึงค่า* `width = 30` *และ* `height =5` *มาใช้ในการคำนวณ ผลลัพธ์ที่ได้จึงเป็น 150 ตามคำตอบแรกที่เก็บไว้ในตัวแปร* `Result1`

ส่วนใน `result2` *กำหนดพารามิเตอร์เพียงตัวเดียว ฟังก์ชั่นจะมองว่า กำหนดให้พารามิเตอร์ตัวแรก นั่นคือ* `width` *ดังนั้นในการทำงาน จะดึงค่า* `height =5` *มาคำนวณ จึงได้คำตอบเป็น 50 (มาจาก 10 x 5)*

สุดท้ายใน `result3` *มีการกำหนดพารามิเตอร์มาครบทั้งสองตัว จึงนำค่ามาคูณกันได้ทันที ได้ผลลัพธ์เป็น 180 (มาจาก 12 x 15)*

จะเห็นว่า ถ้ามีการกำหนดค่าเริ่มต้นให้กับพารามิเตอร์แล้ว เวลาเรียกใช้งานฟังก์ชั่นจะสามารถเลือกได้ว่าจะกำหนดค่าให้กับพารามิเตอร์นั้นๆ ใหม่หรือไม่ ถ้าไม่ได้กำหนด โปรแกรมจะก็เรียกค่าเริ่มต้นที่กำหนดไว้ในฟังก์ชั่นมาใช้งานแทน

ข้อควรระวัง– ถ้าไม่ได้มีการกำหนดค่าเริ่มต้นให้กับฟังก์ชั่น เวลาเรียกใช้งานฟังก์ชั่นนั้นๆ จะต้องกำหนดค่าพารามิเตอร์ด้วยทุกครั้ง ไม่เช่นกันโปรแกรมจะเกิดข้อผิดพลาดขึ้น ดังตัวอย่างที่ 5-53 ต่อไปนี้

ตัวอย่างที่ 5-53

```
def get_square_area(width, height=5):
   return width * height
result = get_square_area()
print ("Result =", result)
```

ผลลัพธ์ที่ได้ :

```
pi@raspberrypi ~ $ sudo python simple_fn.py
Traceback (most recent call last):
  File "simple_fn.py", line 4, in <module>
    result = get_square_area()
TypeError: get_square_area() takes at least 1 argument (0 given)
```

บทที่ 6

Raspberry Pi 2 กับการพัฒนาโปรแกรมภาษา Python ด้วยซอฟต์แวร์ Geany

ในบทที่ 5 นำเสนอเกี่ยวกับที่มาและพื้นฐานของภาษา Python รวมถึงแนวทางในการพัฒนาโปรแกรม โดยทั้งหมดในบทที่ 5 เป็นการพัฒนาโปรแกรมบนเท็กซ์เอดิเตอร์ Nano ที่มากับระบบปฏิบัติการ Raspbian ซึ่งมีลักษณะเป็นหน้าต่างแบบ DOS สำหรับผู้ใช้งานและนักพัฒนาที่ต้องการเครื่องมือพัฒนาโปรแกรมภาษา Python ในแบบกราฟิกหรือแบบที่เรียกว่า IDE (Integrated Development Environment) ที่รวบรวมคำสั่งและเครื่องมือต่างๆ ไว้ในหน้าต่างเดียว ในที่นี้ขอแนะนำซอฟต์แวร์ที่ชื่อ **Geany**

6.1 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 ทั้งการประมวลผลทางซอฟต์แวร์และการติดต่อสั่งงานทางฮาร์ดแวร์ผ่านพอร์ตอินพุตเอาต์พุตอเนกประสงค์หรือ GPIO (General Purpose Input Output) ของบอร์ด Raspberry Pi

สำหรับท่านที่จัดซื้อชุดเรียนรู้ Raspberry Pi 2 จาก INEX ในแผ่น microSD การ์ดได้ทำการติดตั้ง Geany ไว้พร้อมใช้งานบนระบบปฏิบัติการ Raspbian แล้ว

หากระบบปฏิบัติการ Raspbian ที่มีอยู่ไม่มีซอฟต์แวร์ Geany ติดตั้งไว้ ผู้ที่ต้องการใช้งานสามารถทำการติดตั้งได้เอง โดย

(1) เชื่อมต่อบอร์ด Raspberry Pi 2 เข้ากับเครือข่ายอินเทอร์เน็ต

(2) หลังจากบูตระบบเสร็จสิ้น ให้เข้าสู่หน้าต่างเทอร์มินอล นั่นคือ **LX Terminal**

(3) ติดตั้งโปรแกรมด้วยการพิมพ์คำสั่ง

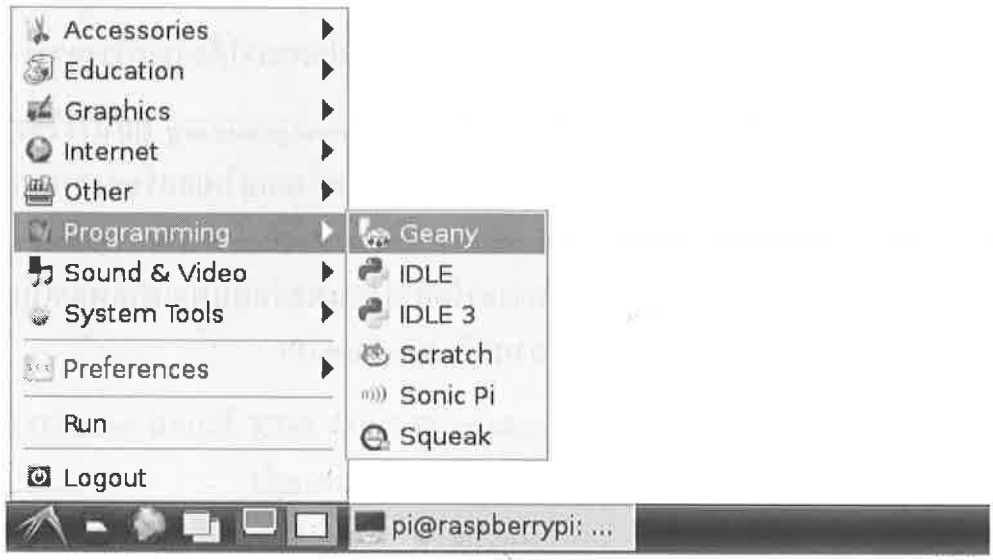
```
pi@raspberrypi ~/code $ sudo apt-get update
pi@raspberrypi ~/code $ sudo apt-get install geany
```

(4) จากนั้นรอจนกระทั่งการติดตั้งโปรแกรมเสร็จสิ้น

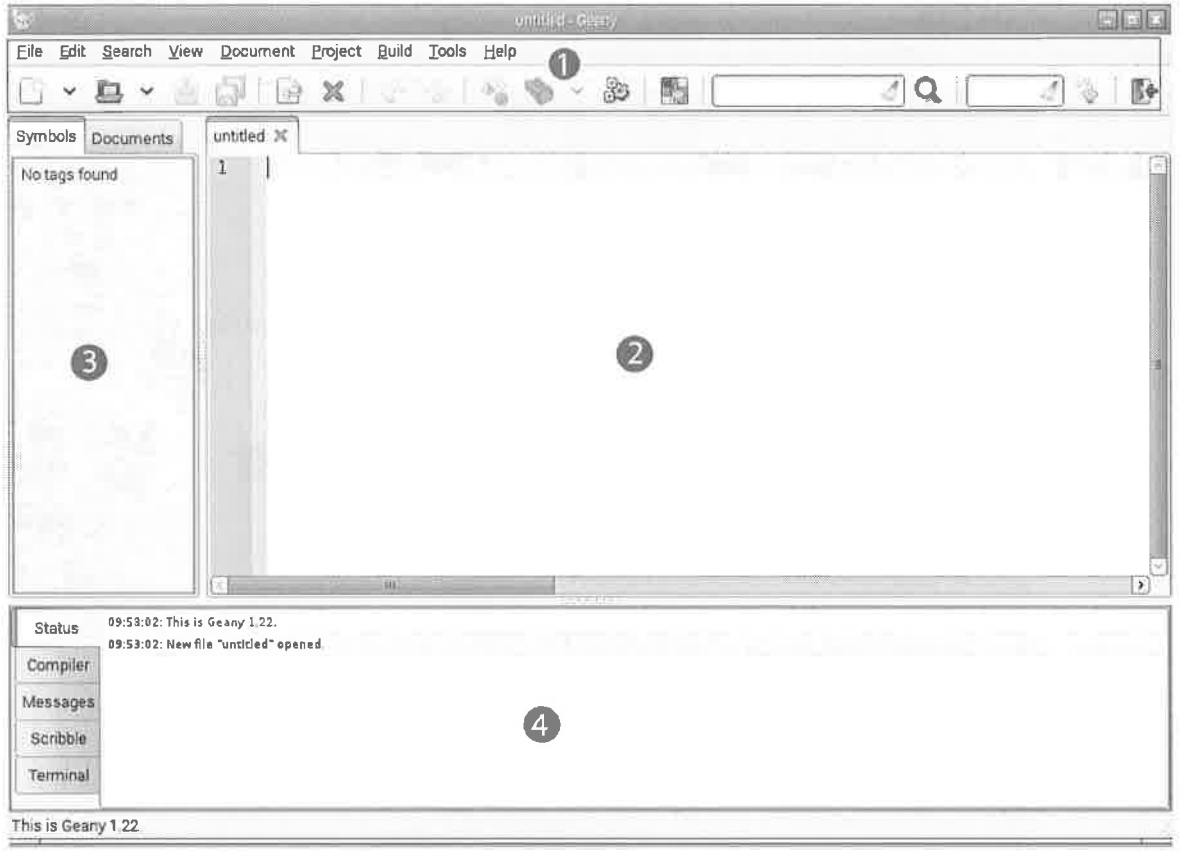
6.2 ส่วนประกอบของ Geany

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

หรือเลือกจากเมนู **Programming > Geany** ดังรูป



เมื่อเปิดโปรแกรมขึ้นมา หน้าต่างหลักของโปรแกรมเป็นดังรูปที่ 6-1



มีส่วนประกอบดังนี้

❶ แถบครื่องมือ ❷ ส่วนที่ใช้เขียนคำสั่ง

❸ ส่วนแสดงผลที่อยู่ของไฟล์เมื่อเลือก Documents และแสดงส่วนประกอบต่างๆ ของโปรแกรมที่เขียนเมื่อเลือก Symbols

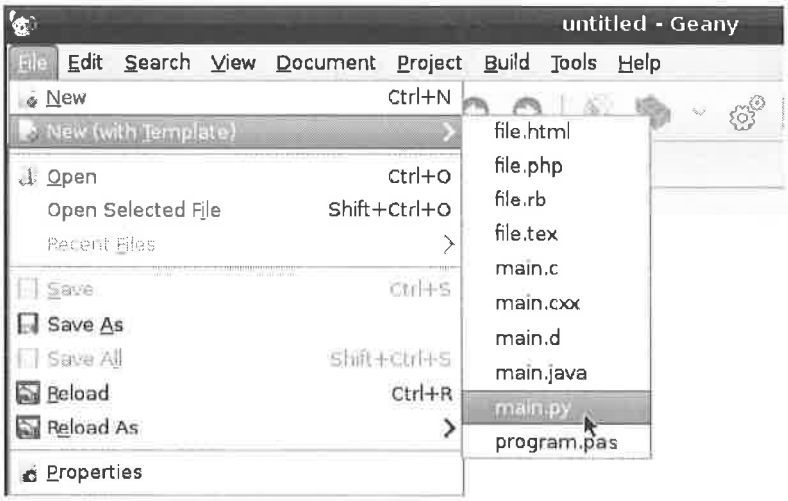
❹ ส่วนแสดงผลลัพธ์ต่างๆ เช่น ผลลัพธ์ของการคอมไพล์โปรแกรม

รูปที่ 6-1 ส่วนประกอบของหน้าต่างหลักของ Geany ซอฟต์แวร์เท็กซ์เอดิเตอร์สำหรับพัฒนาโปรแกรมให้แก่ Raspberry Pi

6.2.1 แถบเครื่องมือ

มีส่วนประกอบดังนี้

① **สร้างไฟล์ใหม่โดยไม่ระบุนามสกุล** ถ้าต้องการสร้างไฟล์ใหม่ที่ระบุนามสกุลด้วย ให้เลือกที่รูปลูกศรชี้ลงจะปรากฏเมนูให้เลือกดังรูป



② **เปิดไฟล์** ถ้าต้องการเปิดไฟล์ที่เคยเปิดมาแล้ว ให้เลือกที่รูปลูกศรชี้ลง จะปรากฏเมนูขึ้นมาให้เลือก

③ **บันทึกไฟล์ที่เลือกไว้**

④ **บันทึกไฟล์ทั้งหมดที่เปิด**

⑤ **เรียกคืนรูปแบบเดิมทั้งหมด** จะใช้ในกรณีที่ยังไม่ได้บันทึกไฟล์

⑥ **ปิดไฟล์ที่เลือกไว้**

⑦ **คอมไพล์ไฟล์สคริปต์**

⑧ **รันไฟล์สคริปต์**

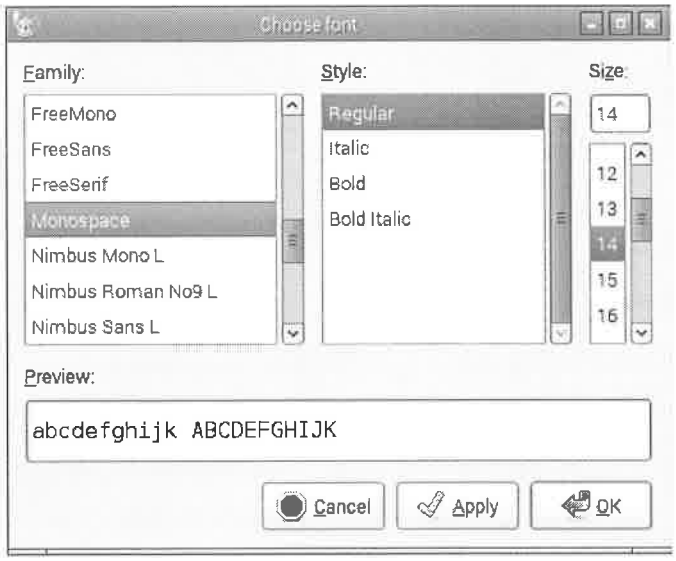
⑨ **ค้นหาคำที่อยู่ในไฟล์**

⑩ **เลือกหมายเลขบรรทัดที่ต้องการดูคำสั่ง**

⑪ **ออกจากโปรแกรม**

6.2.2 การเลือกรูปแบบตัวอักษร

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

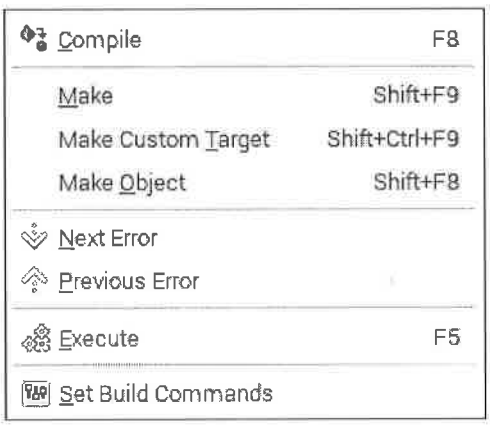


เมื่อตั้งค่าแล้ว คลิกปุ่ม **OK** เพื่อยืนยัน

6.2.3 ตั้งค่าการคอมไพล์และรันโปรแกรมหรือเอ็กซิคิวต์

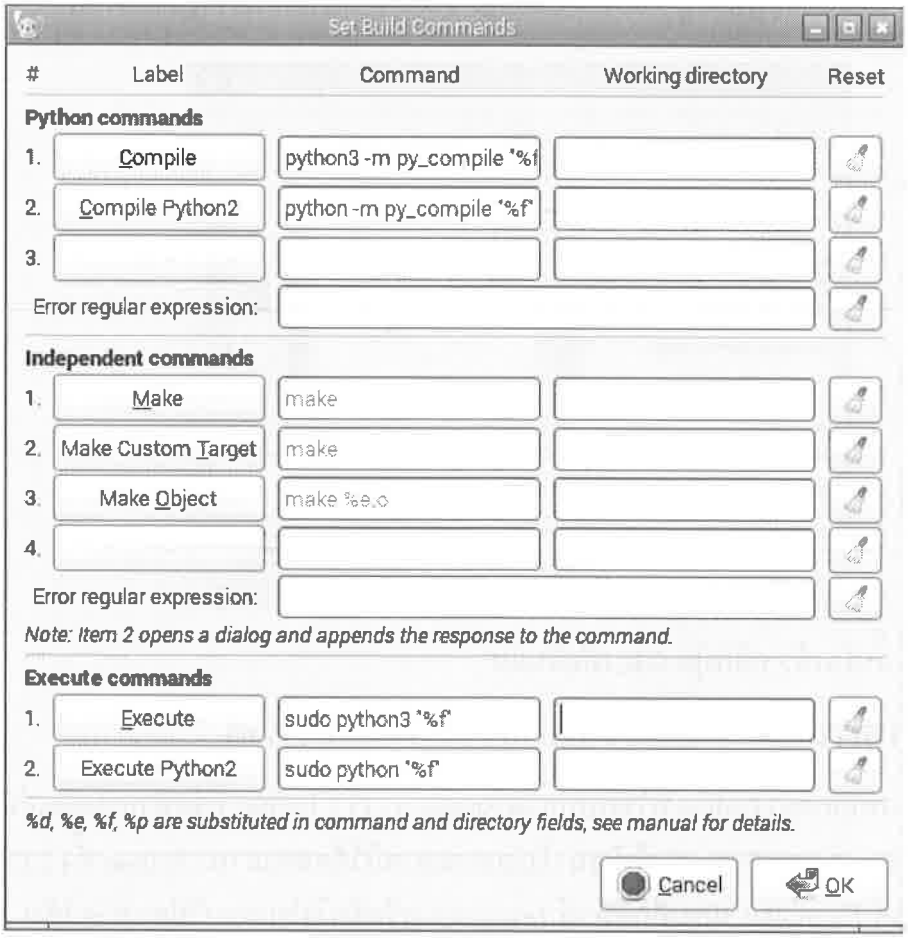
เนื่องจากภาษา Python ที่ใช้กับบอร์ด Raspberry Pi 2 ในขณะที่จัดทำหนังสือเล่มนี้มี 2 เวอร์ชัน หลักคือ Python2 และ Python3 ผู้พัฒนาโปรแกรมอาจมีโค้ดของภาษา Python ทั้ง 2 เวอร์ชัน หากทำการคอมไพล์ (compile) และเอ็กซิคิวต์ (execute) หรือรันโปรแกรมไม่ตรงเวอร์ชัน อาจทำให้ไฟล์สคริปต์ทำงานไม่ได้ หรือคอมไพล์ไม่ผ่าน จึงขอแนะนำการเพิ่มเมนู เพื่อให้ผู้ใช้งาน Geany ได้มีทางเลือกในการคอมไพล์และเอ็กซิคิวต์หรือรันโปรแกรมดังนี้

(1) ไปที่ **Build** จะปรากฏเมนูให้เลือกดังรูป

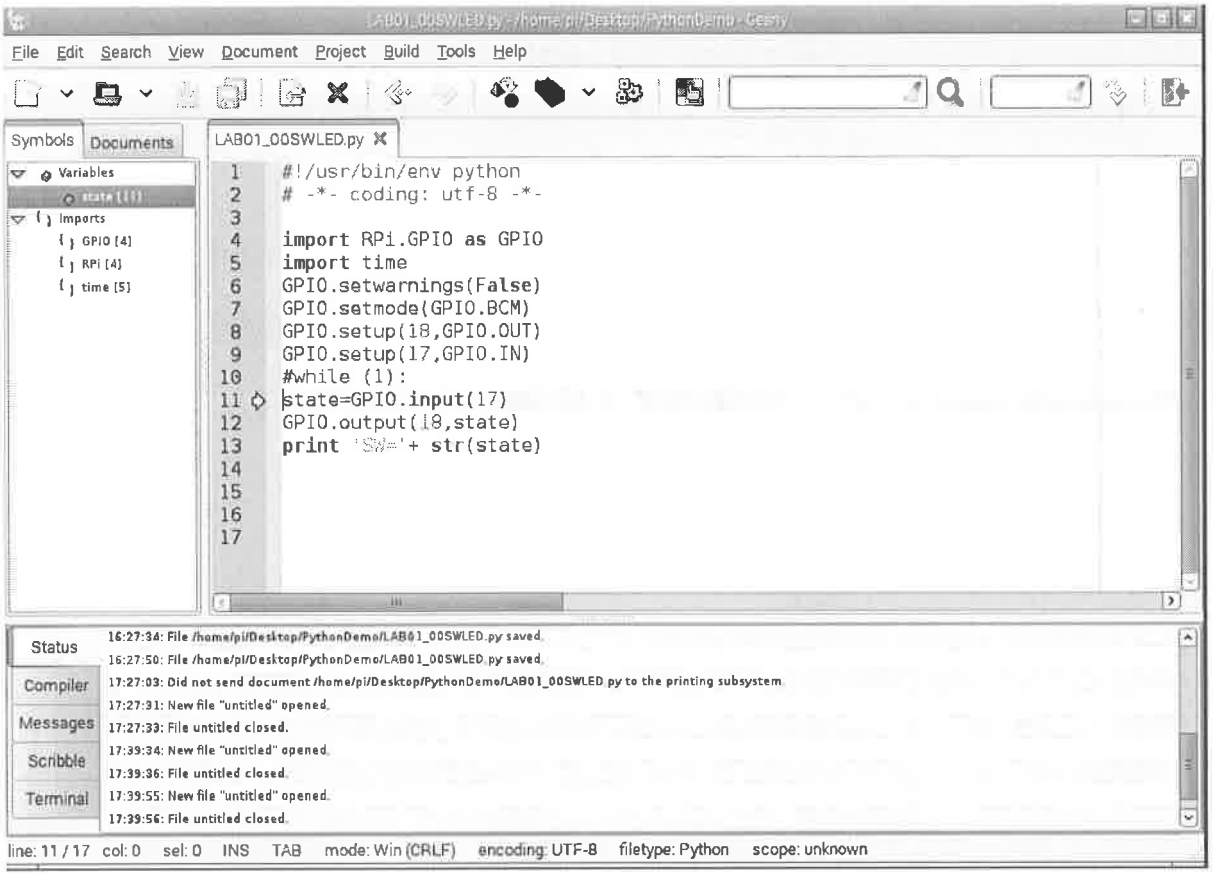


*จะเห็นว่ามีคำสั่ง **Complie** และ **Execute** ให้เลือก โดยทั้งสองคำสั่งใช้สำหรับ Python3*

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



(3) จากนั้นไปที่ **Build** จะเห็นว่า มีเมนูเพิ่มขึ้นมาจากเดิมดังรูป



รูปที่ 6-2 แสดงตัวอย่างหน้าต่างของ Geany เมื่อมีการเปิดไฟล์ของโปรแกรมภาษา Python ขึ้นมาใช้งาน

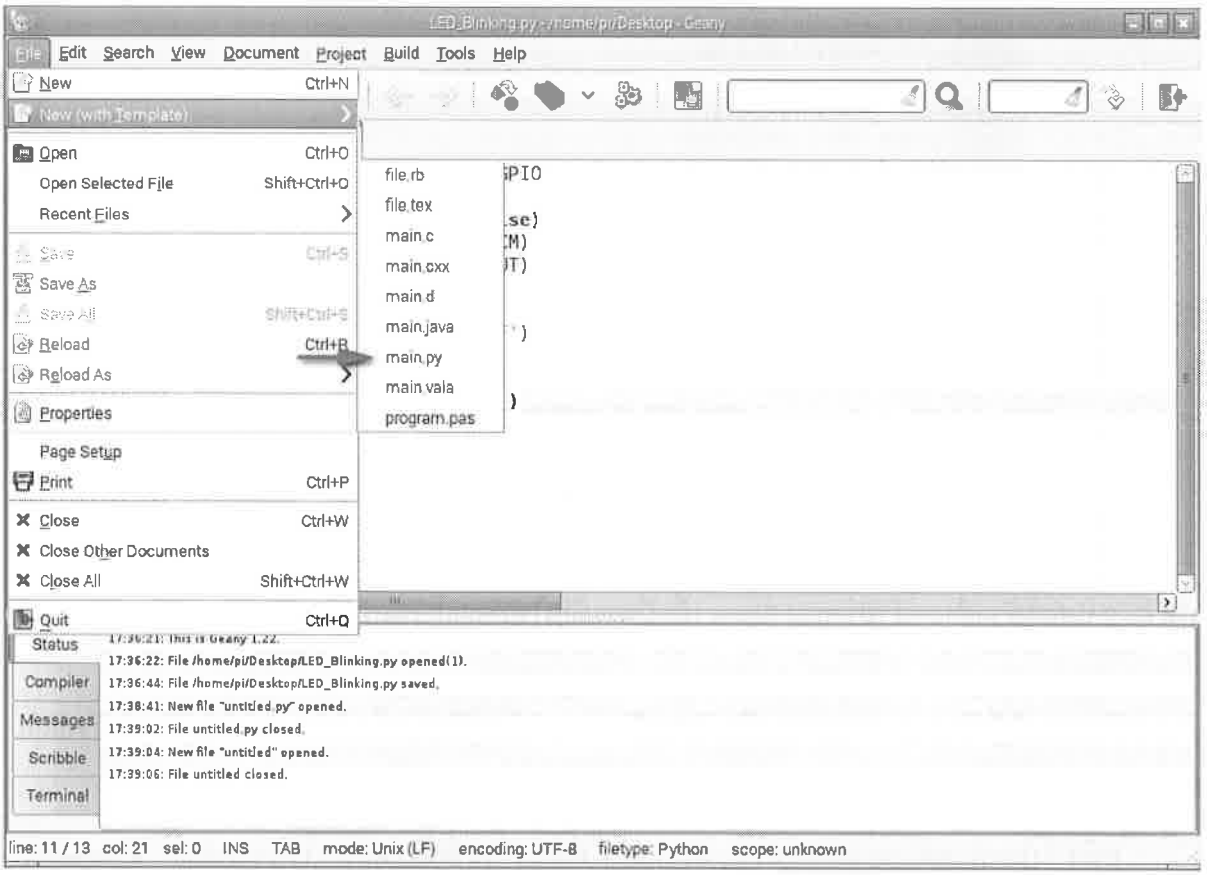
นับจากนี้ผู้พัฒนาจะสามารถเลือกคอมไพล์และเอ็กซิคิวต์หรือรันโปรแกรมภาษา Python ได้ทั้ง Python 2 และ Python 3 ผ่านทางเมนู Build ได้ตามต้องการ

อนึ่ง โปรแกรมตัวอย่างของภาษา Python ที่พัฒนาขึ้นเพื่อควบคุมการทำงานของ Raspberry Pi 2 ในหนังสือเล่มนี้นับจากบทนี้จะใช้ Geany เป็นเท็กซ์เอดิเตอร์หลักและใช้ Python3 ในการพัฒนาโปรแกรม

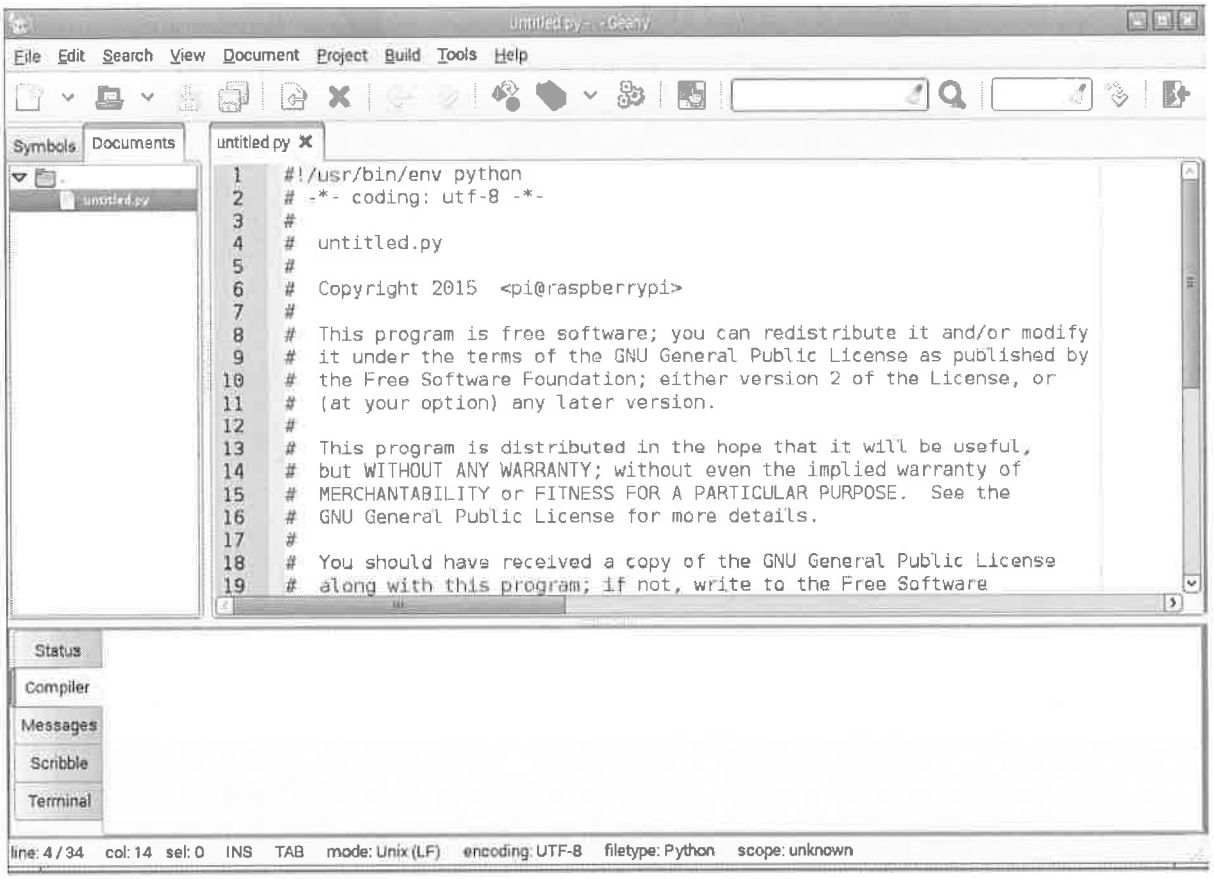
6.3 ตัวอย่างการทดสอบโปรแกรมภาษา Python ด้วย Geany

ในการเขียนโปรแกรมภาษา Python สำหรับ Raspberry Pi 2 ในที่นี้ ขอแนะนำให้ใช้ Python 3 เป็นหลัก สำหรับการพัฒนาโปรแกรมบน Geany เบื้องต้น มีขั้นตอนดังนี้

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



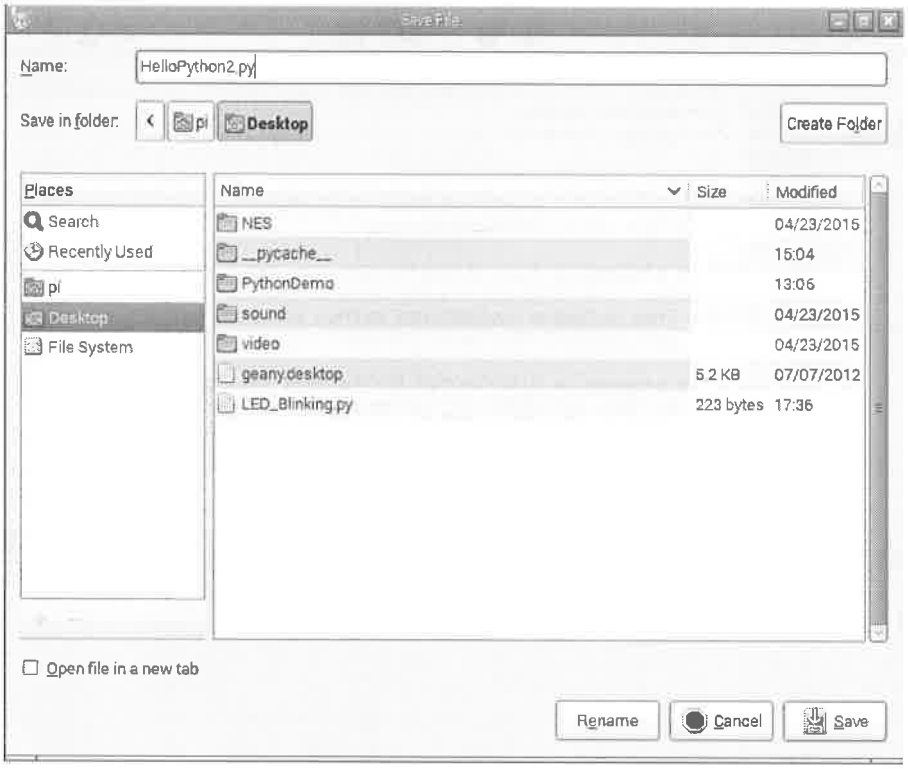
(2) จะปรากฏไฟล์สคริปต์ที่มีชื่อว่า **untitled.py** ดังรูป[20]



(3) พิมพ์คำสั่งต่อไปนี้

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

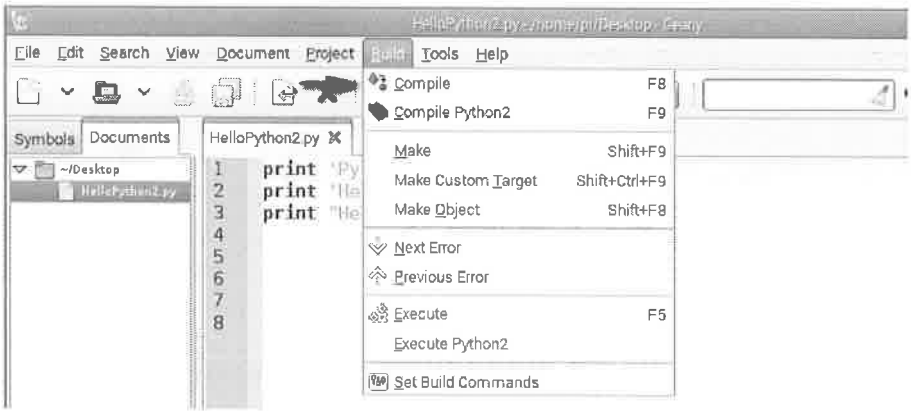
(4) บันทึกไฟล์สคริปต์ โดยไปที่ **File > Save** เลือกที่อยู่ของ File แล้วตั้งชื่อว่า **HelloPython3.py** ในที่นี้จะเก็บไฟล์ไว้ที่ **Desktop** แล้วคลิก **Save**



(5) ทดสอบคอมไพล์และเอ็กซิคิวต์ คำสั่ง `print` ที่พิมพ์ลงในไฟล์สคริปต์เป็นคำสั่งแสดงข้อความออกทางหน้าจอภาพ รูปแบบคำสั่ง `print` ของ Python3 และ 2 แตกต่างกัน ในที่นี้จะใช้ Python3 เป็นหลัก ดังนั้นหากผู้พัฒนาโปรแกรมมีตัวอย่างโปรแกรมเดิมของ Python2 เมื่อนำมาคอมไพล์หรือเอ็กซิคิวต์ด้วย Python3 จะต้องทำการแก้ไขคำสั่งให้ถูกต้องก่อน โดยศึกษาได้จาก help ของ Python3

(5.1) เริ่มต้นด้วยการคอมไพล์

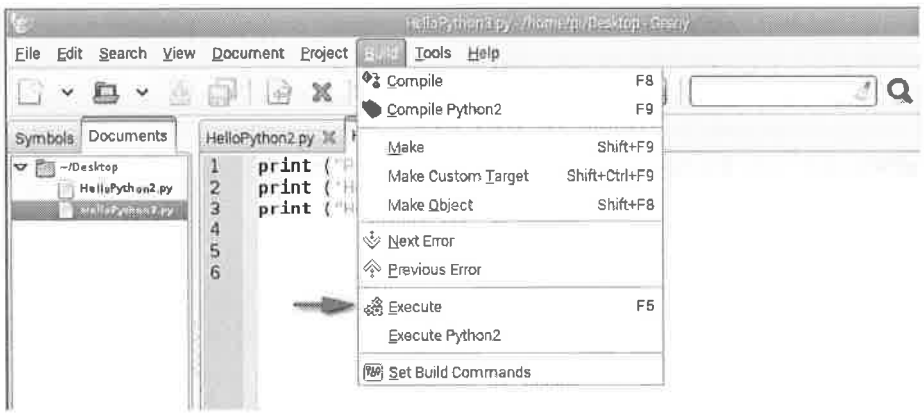
(5.1.1) ไปที่ **Build** จะปรากฏเมนูดังรูป



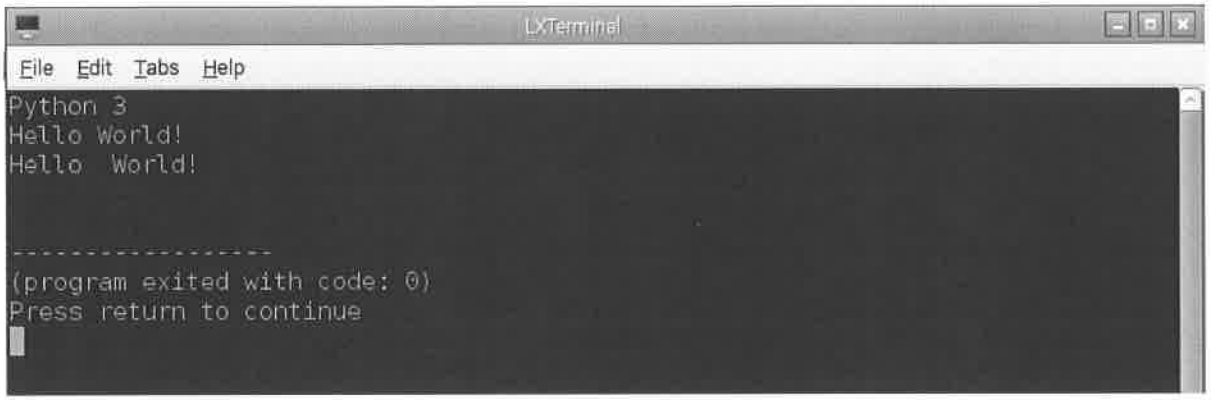
(5.1.2) คลิกที่ **Compile** เพื่อใช้ Python3 คอมไพเลอร์ในการคอมไพล์ จากนั้นผลลัพธ์ของการคอมไพล์จะปรากฏที่กรอบด้านล่างของหน้าต่างโปรแกรม โดยแสดงข้อความว่า **Compilation finished successfully** แสดงว่ารูปแบบคำสั่งถูกต้องพร้อมที่จะทำการเอ็กซิคิวต์หรือรันต่อไป

(5.2) การเอ็กซิคิวต์หรือสั่งให้รันโปรแกรม

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



(5.2.2) ผลลัพธ์ของการเอ็กซิคิวต์แสดงดังรูป



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

สำหรับการเรียนรู้เพื่อเขียนโปรแกรมควบคุมการทำงานของบอร์ด Raspberry Pi 2 ในหนังสือเล่มนี้เลือกใช้ภาษา Python และเป็นรุ่น Python3 โดยได้ทำการผนวกคอมไพเลอร์ไว้ใน Geany ตั้งแต่ขั้นตอนการสร้างอิมเมจไฟล์สำหรับระบบปฏิบัติการ Raspbian ที่ทาง INEX จัดทำขึ้น ทั้งนี้เพื่ออำนวยความสะดวกในการเรียนรู้และพัฒนาโปรแกรม

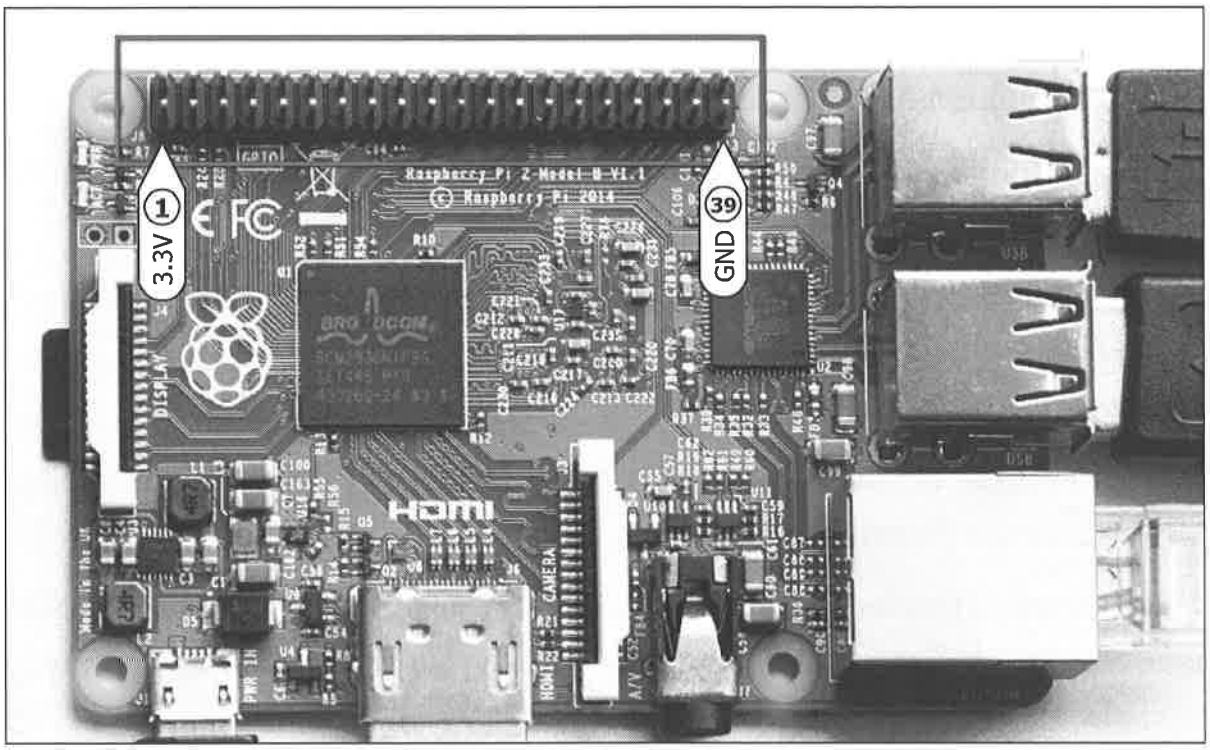
INNOVATIVE EXPERIMENT

บทที่ 7
การติดต่อพอร์ตอินพุตเอาต์พุตอเนกประสงค์ หรือ GPIO ของ Raspberry Pi 2

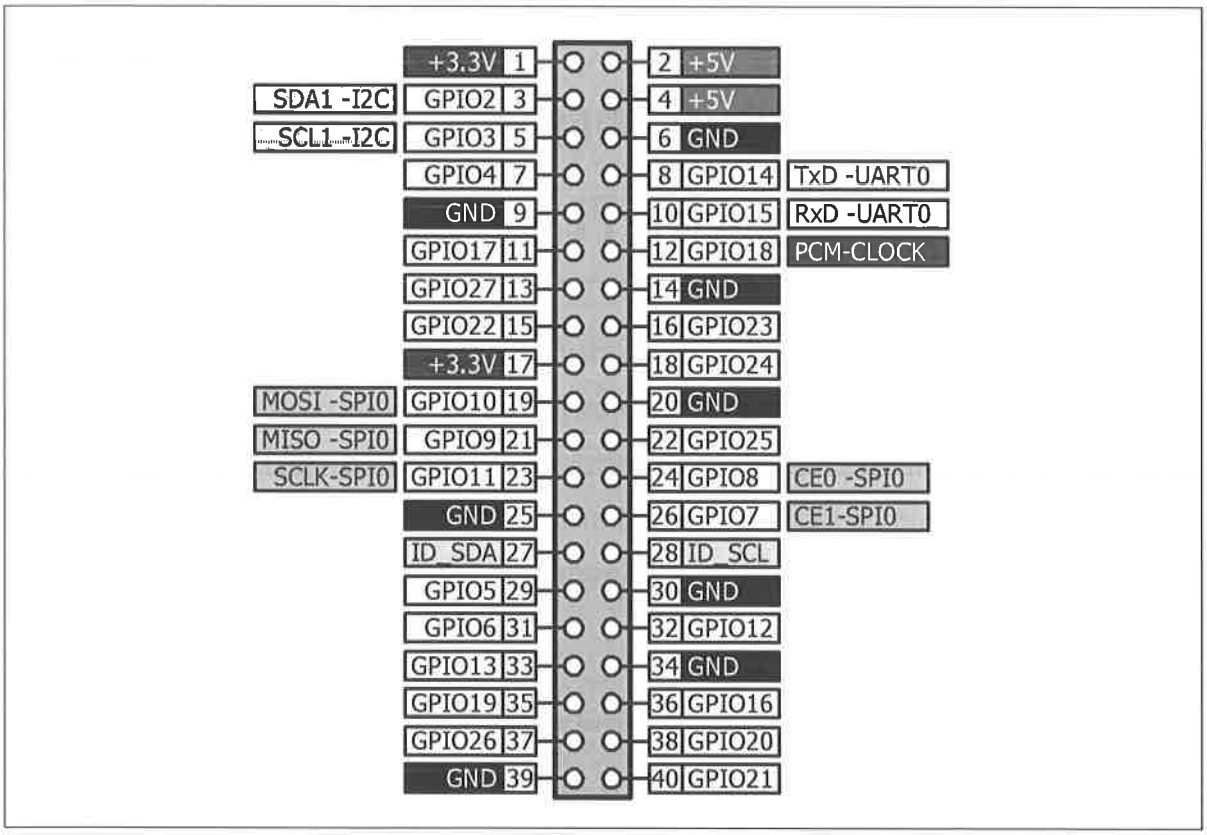
สิ่งที่ทำให้บอร์ด Raspberry Pi 2 เป็นที่นิยมนั้นไม่ได้มีเพียงเรื่องราคาและขนาด แต่บอร์ด Raspberry Pi 2 เป็นบอร์ดคอมพิวเตอร์ที่มีขาพอร์ตอินพุตเอาต์พุตอเนกประสงค์หรือ GPIO (General Purpose Input Output) สำหรับนำไปเชื่อมต่อกับวงจรอิเล็กทรอนิกส์ได้ จึงทำให้นำบอร์ด Raspberry Pi 2 ไปสร้างสรรค์โครงงานและสิ่งประดิษฐ์ได้อย่างมากมาย และนั่นคือจุดขายที่สำคัญที่นำโปรแกรมเมอร์มาพบกับนักทดลองวงจรอิเล็กทรอนิกส์ ส่งผลให้เกิดกลุ่มของผู้ใช้งานที่กว้างขึ้นอย่างมากมายมหาศาล

7.1 การจัดขา GPIO

บอร์ด Raspberry Pi 2 มีคอนเน็กเตอร์สำหรับขาพอร์ต GPIO 40 ขา แบ่งเป็น 2 แถว แถวละ 20 ขา โดยแต่ละขาจะมีหน้าที่ต่างกันดังแสดงในรูปที่ 7-2 จะเห็นได้ว่า มีขาพอร์ตอินพุตเอาต์พุตสำหรับใช้งานรวม 26 ขา มีระดับสัญญาณลอจิกเป็น 0 และ +3.3V จึงทำให้ Raspberry Pi 2 สามารถเชื่อมต่ออุปกรณ์อิเล็กทรอนิกส์ได้หลากหลาย ตั้งแต่ LED, สวิตช์ ไปจนถึงไอซีหน้าที่พิเศษหรือตัวตรวจจับต่างๆ ที่ใช้การเชื่อมต่อผ่านบัส I^2C และ SPI



รูปที่ 7-1 แสดงตำแหน่งของคอนเน็กเตอร์ GPIO 40 ขาบนบอร์ด Raspberry Pi 2



รูปที่ 7-2 การจัดขาคอนเน็กเตอร์ GPIO ของบอร์ด Raspberry Pi 2

ขาพอร์ตอินพุตเอาต์พุตของ Raspberry Pi 2 มีทั้งสิ้น 26 ขา มีการแบ่งปันหน้าที่การทำงานดังนี้

1. เป็นขาพอร์ตอินพุตเอาต์พุตดิจิตอล 26 ขา

2. ใช้งานเป็นขาเอาต์พุตสัญญาณ PWM ได้ 26 ขา

3. ใช้งานเป็นขาเชื่อมต่อบัส I^2C 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 2 ภายใต้มาตรฐาน **HAT** (Hardware Attached on Top)

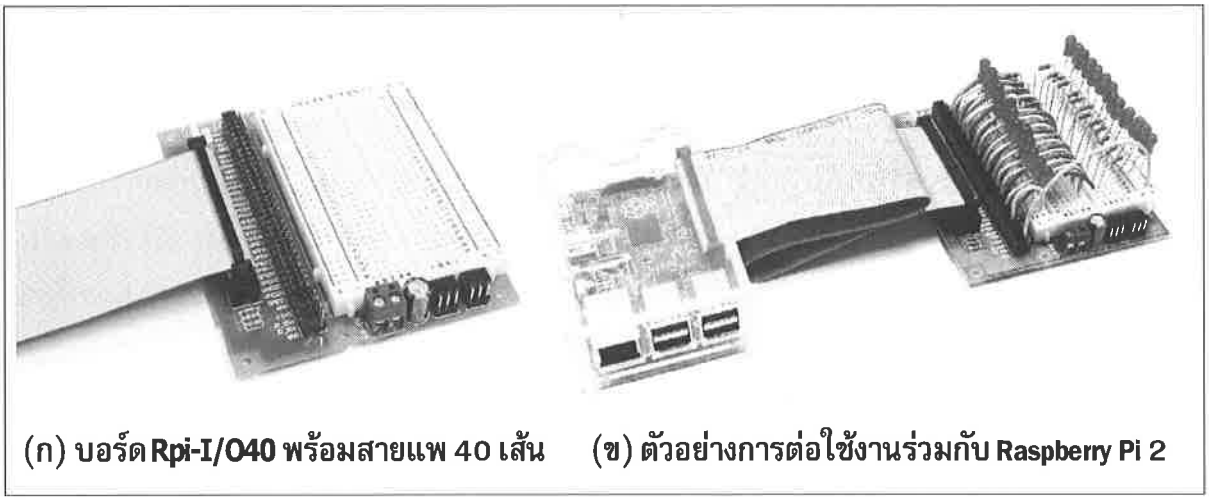
ขาพอร์ต GPIO ทั้งหมดรับและขับสัญญาณได้สูงสุด +3.3V ไม่สามารถรับแรงดัน +5V ได้เมื่อทำงานเป็นพอร์ตอินพุต ด้านความสามารถในการรับและจ่ายกระแสไฟฟ้าของพอร์ต GPIO แต่ละขาไม่เกิน 50mA แต่รวมทุกขาพอร์ตจะจ่ายกระแสไฟฟ้าได้ไม่เกิน 300mA เมื่อทำงานเป็นพอร์ตเอาต์พุต

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

7.2 Rpi-I/O40 บอร์ดอินพุตเอาต์พุตสำหรับ Raspberry Pi 2

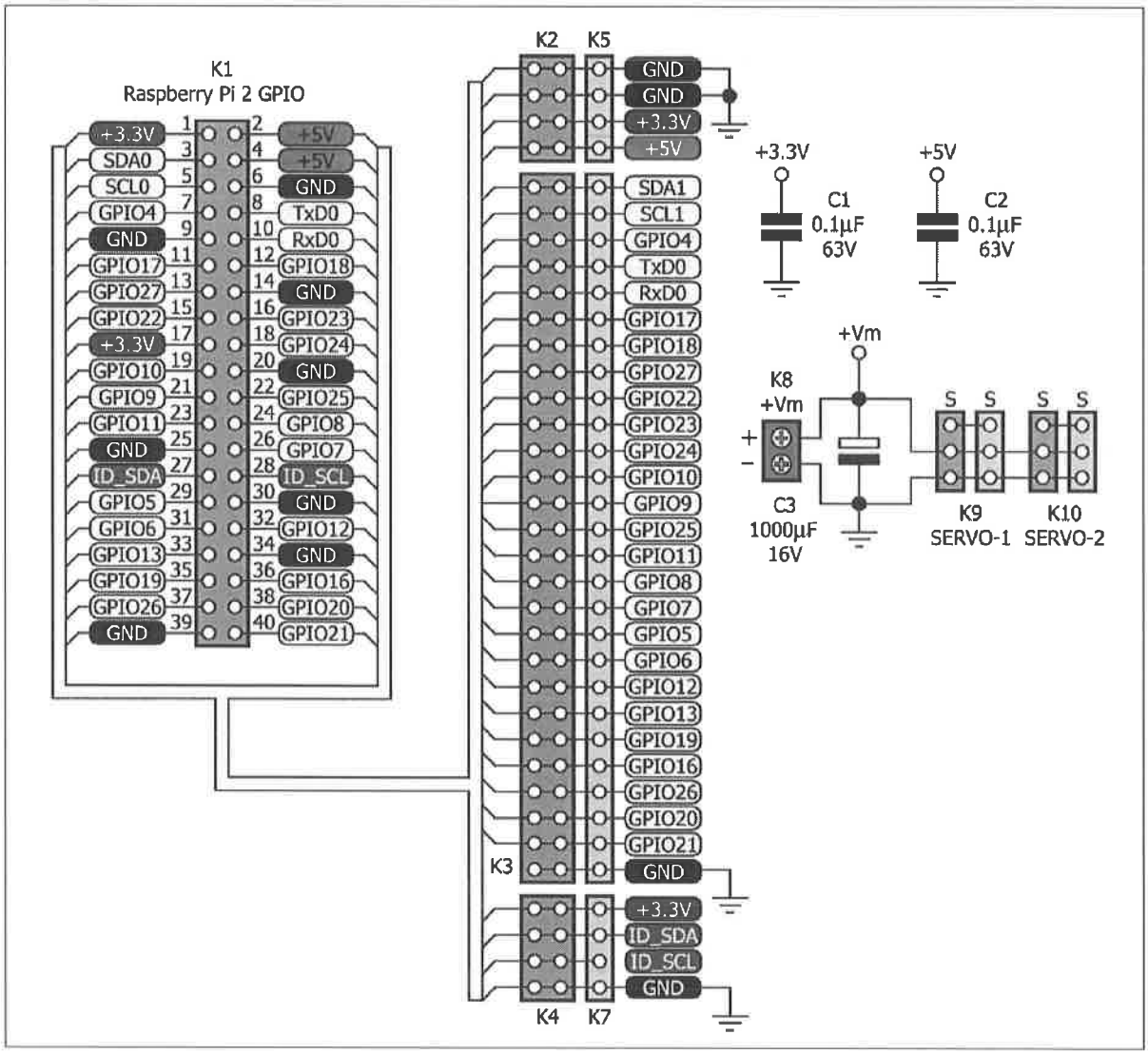
ในการใช้งานเพื่อเชื่อมต่อขาพอร์ต GPIO ของบอร์ด Raspberry Pi 2 เพื่อความปลอดภัยและสะดวกในการต่อวงจร จึงควรต่อสายออกมาแล้วทำการต่อวงจรทดลองบนเบรดบอร์ดหรือแผงต่อวงจรอุปกรณ์ที่แนะนำในบทนี้คือ **บอร์ด Rpi-I/O40** ที่มีพื้นที่สำหรับการสร้างวงจรหรือ Proto-Area ขนาดใหญ่ที่เทียบเท่ากับเบรดอร์ดขนาด 400 จุดต่อ รวมถึงมีการจัดสรรขาพอร์ตสำหรับการเชื่อมต่อเพื่อทดลองหรือเรียนรู้ได้สะดวกขึ้น รวมทั้งยังมีจุดต่อเซอร์โวมอเตอร์และจุดต่อไฟเลี้ยงสำหรับเซอร์โวมอเตอร์ไว้ให้ด้วย เพื่อช่วยเสริมในการนำแผงวงจรสำหรับทดลองตัวนี้ไปใช้ให้เกิดประโยชน์ที่น่าทึ่งต่อไป

ในรูปที่ 7-3 แสดงหน้าตาของบอร์ด Rpi-I/O40 พร้อมทั้งสายแพที่ใช้ในการเชื่อมต่อกับบอร์ด Raspberry Pi 2 และตัวอย่างการต่อใช้งานจริง



(ก) บอร์ด Rpi-I/O40 พร้อมสายแพ 40 เส้น (ข) ตัวอย่างการต่อใช้งานร่วมกับ Raspberry Pi 2

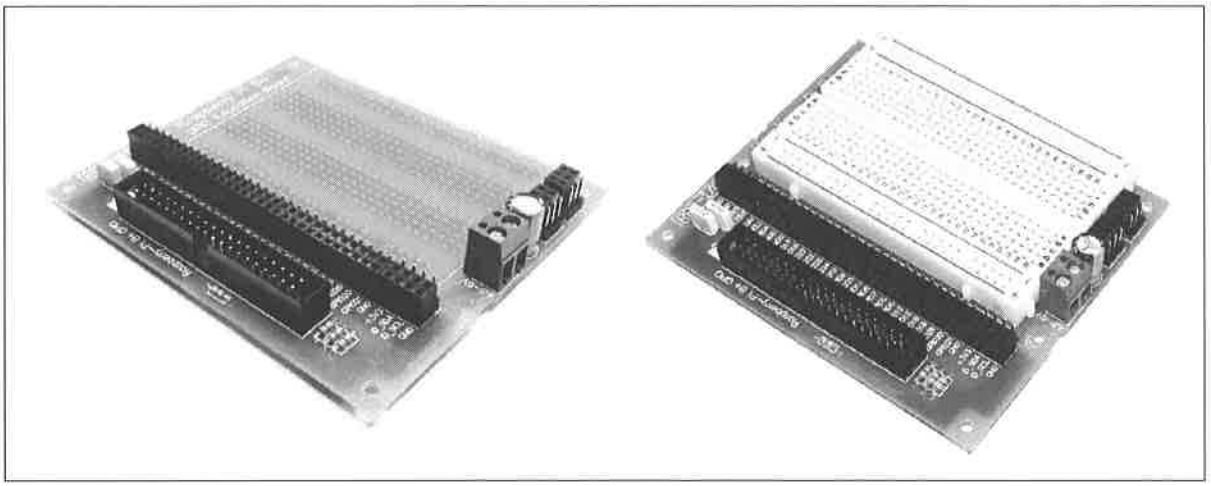
รูปที่ 7-3 บอร์ด Rpi-I/O40 พร้อมสายแพที่ใช้ในการเชื่อมต่อกับบอร์ด Raspberry Pi 2



รูปที่ 7-4 วงจรของ Rpi-I/O40 บอร์ดทดลองและพัฒนาโครงงานด้วย Raspberry Pi 2

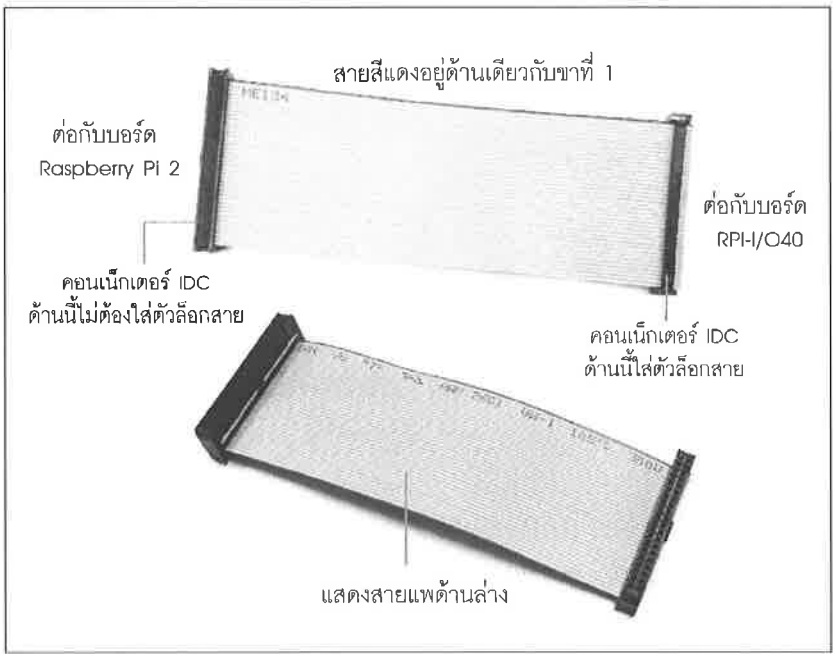
7.2.1 วงจรของบอร์ด RPi-I/O40

รูปที่ 7-4 แสดงวงจรของบอร์ด RPi-I/O40 สัญญาณจากคอนเน็กเตอร์ GPIO ของบอร์ด Raspberry Pi 2 จะถูกต่อเข้ามาที่คอนเน็กเตอร์ K1 จากนั้นจะถูกเชื่อมต่อมายังคอนเน็กเตอร์ K2 ถึง K7 โดยขาพอร์ตอินพุตเอาต์พุตหลักๆ จะถูกต่อมายังคอนเน็กเตอร์ K3 และ K6 โดย K3 เป็นคอนเน็กเตอร์ IDC ตัวเมีย 27 ขา แถวคู่ (รวม 54 ขา) และ K6 เป็นคอนเน็กเตอร์ IDC ตัวผู้ 27 ขา แถวเดี่ยว ส่วน K2 และ K5 เป็นจุดต่อไฟเลี้ยง +3.3V, +5V และกราวด์ โดย K2 เป็นคอนเน็กเตอร์ IDC ตัวเมีย 4 ขา แถวคู่ (รวม 8 ขา) และ K5 เป็นคอนเน็กเตอร์ IDC ตัวผู้ 4 ขา แถวเดี่ยว สำหรับ K4 และ K7 เป็นจุดต่อไอซีหน่วยความจำอีอีพรอมอนุกรม ซึ่งมีด้วยกัน 4 ขาคือ +3.3V, กราวด์, ID_SCL และ ID_SDA โดย K4 เป็นคอนเน็กเตอร์ IDC ตัวเมีย 4 ขา แถวคู่ (รวม 8 ขา) และ K7 เป็นคอนเน็กเตอร์ IDC ตัวผู้ 4 ขา แถวเดี่ยว



รูปที่ 7-5 แสดงการใช้งานพื้นที่สร้างวงจรของบอร์ด Rpi-I/O40
(ซ้าย) ปล่อยว่างไว้เพื่อรอรับการบัดกรีอุปกรณ์เพื่อสร้างวงจร
(ขวา) ติดตั้งเบรดบอร์ดเพื่อทดลองวงจร

นอกจากนั้นบนบอร์ด RPi-I/O40 ยังเตรียมวงจรเชื่อมต่อเซอร์โวมอเตอร์ไว้เพิ่มเติมอีก 2 ช่อง โดยไฟเลี้ยงเซอร์โวมอเตอร์ถูกต่อเข้าที่จุดต่อ K8 มีตัวเก็บประจุ C3 ค่า 470µF 10V เพื่อช่วยในการรักษาระดับแรงดันไฟเลี้ยงเซอร์โวมอเตอร์ให้มีความแน่นอน ในขณะที่เซอร์โวมอเตอร์ทำงาน ส่วนขาพอร์ตสำหรับขับเซอร์โวมอเตอร์จะถูกต่อมายังคอนเน็กเตอร์ K9 และ K10 โดยแต่ละตัวจะประกอบด้วยคอนเน็กเตอร์ IDC 3 ขาตัวผู้และตัวเมียอย่างละ 1 ตัว หากต้องการใช้ขาพอร์ตของ Raspberry Pi 2 เพื่อขับเซอร์โวมอเตอร์ ให้ต่อขาพอร์ตมายังจุดต่อ K9 หรือ K10



รูปที่ 7-6 สายแพที่ติดตั้งเข้ากับคอนเน็กเตอร์ IDC ตัวเมีย 40 ขา

7.2.2 การใช้งานบอร์ด Rpi-I/O40

บนบอร์ด Rpi-I/O40 พื้นที่ว่างสำหรับสร้างวงจรที่มีจุดบัดกรีและการเชื่อมต่อตรงกับแผงต่อวงจรหรือเบรดบอร์ด จึงอาจเว้นไว้เพื่อรองรับการบัดกรีสร้างวงจร หรือนำเบรดบอร์ดขนาด 400 จุดมาติดตั้งก็ได้ ดังรูปที่ 7-5

ในการใช้งานต้องนำสายแพ 40 เส้นมาเชื่อมต่อระหว่างบอร์ด RPi-I/O40 กับ Raspberry Pi 2 โดยสายแพที่ใช้จะมีหน้าตาและตำแหน่งการยึดคอนเน็กเตอร์เข้ากับสายแพดังรูปที่ 7-6

7.3 การควบคุม GPIO บนบอร์ด Raspberry Pi 2 ด้วยโปรแกรมภาษา Python เบื้องต้น

สำหรับการเขียนโปรแกรมควบคุม GPIO บนบอร์ด Raspberry Pi 2 ทำได้หลายวิธี ไม่ว่าจะเป็นการใช้โปรแกรมภาษา Python, C/C++, Java หรือแม้แต่ Javascript ก็ตาม

วิธีที่แนะนำในที่นี้เป็นการสั่งงานผ่านไฟล์สคริปต์ที่สร้างขึ้นด้วยภาษา Python โดยมีไลบรารีที่ชื่อว่า **RPi.GPIO** เป็นหัวใจหลัก ซึ่งในระบบปฏิบัติการ Raspbian เวอร์ชันใหม่ๆ จะติดตั้งไลบรารี RPi.GPIO มาให้แล้ว จึงไม่จำเป็นต้องติดตั้งเพิ่ม

แต่ถ้ายังไม่ได้ติดตั้ง ให้เข้าสู่หน้าต่างเทอร์มินอล แล้วพิมพ์คำสั่งที่เชลพร้อมพ์ ดังนี้

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

7.3.1 การกำหนดขา GPIO

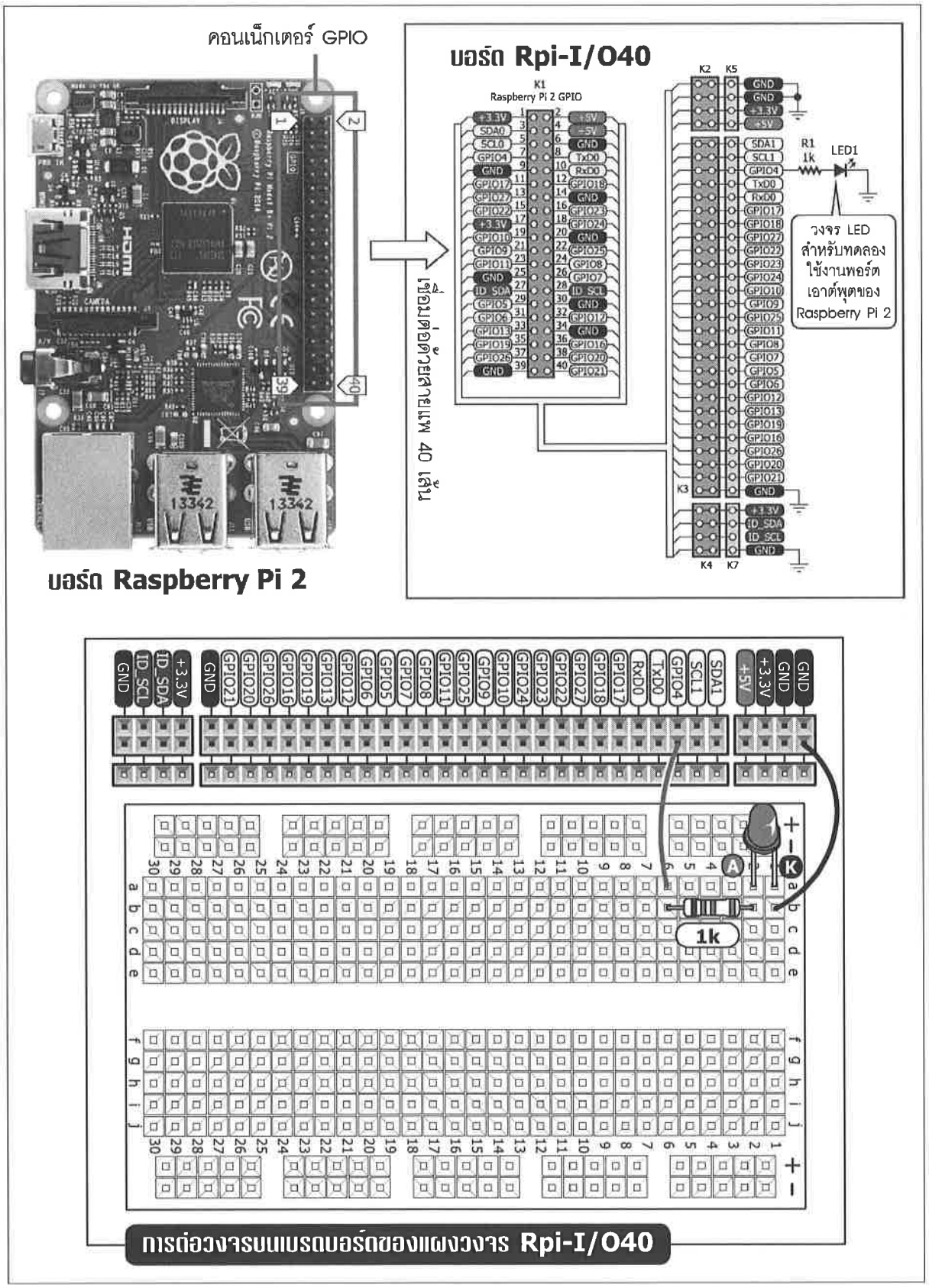
บอร์ด Raspberry Pi 2 จะมีการเรียกใช้งานขา GPIO ในภาษา Python อยู่ 2 แบบคือ อิงตามเลขขาบนบอร์ด (Board) และอิงตามขาของตัวชิป Broadcom (BCM) การเลือกรูปแบบการจัดขาใช้คำสั่ง

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

ตัวอย่างในหนังสือเล่มนี้ใช้การอ้างอิงขาพอร์ตตามขาของตัวชิป Broadcom (GPIO.BCM)

7.3.2 ติดต่อ GPIO ของบอร์ด Raspberry Pi 2 เพื่อสั่งงานขาพอร์ตเอาต์พุตเบื้องต้นด้วยภาษา Python

ตัวอย่างแรกของการใช้งานขา GPIO ของบอร์ด Raspberry Pi 2 เบื้องต้น เป็นการใช้งานขา GPIO4 ของ Raspberry Pi เป็นขาพอร์ตเอาต์พุตเพื่อขับ LED ให้ติดดับสลับกัน หรือทำงานเป็นไฟกะพริบ 1 ดวง



รูปที่ 7-7 วงจรทดลองใช้งานขาพอร์ต GPIO ของบอร์ด Raspberry Pi 2 อย่างง่ายด้วยการสร้างวงจรไฟกะพริบ

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(4,GPIO.OUT)
while True:
   GPIO.output(4,0)
   print('GPIO4=OFF')
   time.sleep(1)
   GPIO.output(4,1)
   print('GPIO4=ON')
   time.sleep(1)
```

โปรแกรมที่ 7-1 โค้ดภาษา Python สำหรับใช้งานขาพอร์ต GPIO4 ของ Rapberry Pi 2 ขับ LED (ใช้ Python3 คอมไพเลอร์)

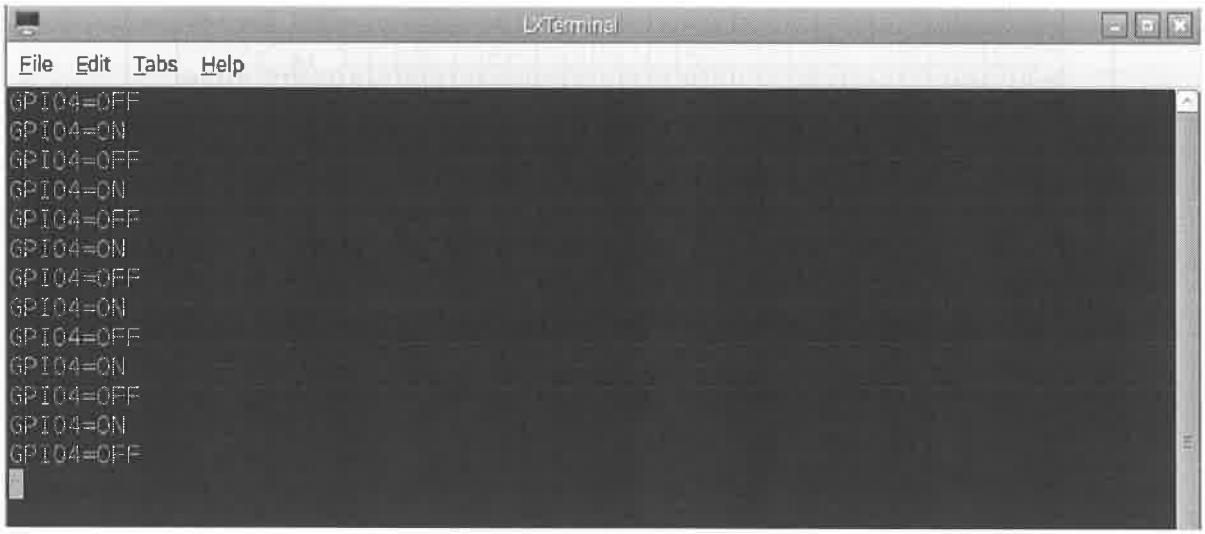
(1) ต่อวงจรเพื่อเชื่อมต่อบอร์ด Raspberry Pi 2 กับ LED ดังวงจรในรูปที่ 7-7

(2) เปิดซอฟต์แวร์ Geany เพื่อสร้างไฟล์สคริปต์ โดยเขียนโปรแกรมที่ 7-1

(3) บันทึกไฟล์สคริปต์ โดยตั้งชื่อว่า LED_Blinking.py

(4) เอ็กซิคิวต์ด้วย Python3 โดยไปที่เมนู **Bulid > Execute**

หน้าต่าง LX Terminal จะปรากฏขึ้นมาเพื่อแสดงสถานะของ LED ดังรูปที่ 7-8 และ LED ที่ต่อกับขาพอร์ต GPIO4 จะติดดับด้วยอัตรา 1 วินาที ถ้าต้องการหยุดการทำงานให้กด Ctrl และ C (กดคีย์ Ctrl ค้างไว้ แล้วกดคีย์ C ตาม)



รูปที่ 7-8 หน้าต่าง LX Terminal แสดงผลการทำงานของโปรแกรมที่ 7-1 ซึ่งเป็นสถานะของขาพอร์ต GPIO4 ของบอร์ด Raspberry Pi 2

7.4 ใช้งานพอร์ต GPIO ของ Raspberry Pi 2 เป็นพอร์ตอินพุตเอาต์พุตดิจิตอล

พอร์ตอินพุตดิจิตอลและเอาต์พุตดิจิตอลเป็นคุณสมบัติพื้นฐานของ GPIO ของ Raspberry Pi 2 เป็นการสั่งให้สถานะของขา GPIO เป็นลอจิกสูง (High) หรือลอจิกต่ำ (Low) ในการใช้งานพอร์ตอินพุตเอาต์พุตดิจิตอลทุกครั้งจะต้องกำหนดให้ขา GPIO นั้นๆ ทราบด้วยว่า ทำงานแบบใด เพราะ GPIO ของ Raspberry Pi 2 ไม่สามารถทำงานเป็นพอร์ตอินพุตและเอาต์พุตดิจิตอลได้พร้อมกัน

7.4.1 การกำหนดทิศทางของขาพอร์ต

ตัวอย่างการกำหนดการทำงานให้กับขา GPIO

```
GPIO.setup(4,  GPIO.OUT)
```

กำหนดให้ขา GPIO4 เป็นพอร์ตเอาต์พุตดิจิตอล

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

กำหนดให้ขา GPIO5 เป็นพอร์ตอินพุตดิจิตอล

สำหรับขาพอร์ตอินพุตดิจิตอลยังกำหนดได้ด้วยว่า จะให้มีการต่อพูลอัป (pull up -กำหนดให้เป็นลอจิกสูง) หรือพูลดาวน์ (pull down - กำหนดให้เป็นลอจิกต่ำ) ให้กับขาพอร์ตในขณะยังไม่มีการป้อนสัญญาณเข้ามา โดยกำหนดได้ดังนี้

สำหรับพูลอัป :

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

สำหรับพูลดาวน์ :

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

7.4.2 พอร์ตเอาต์พุตดิจิตอล

เมื่อต้องการให้ขา GPIO ขับสัญญาณตามที่กำหนด จะต้องกำหนดให้ทำงานเป็นพอร์ตเอาต์พุตดิจิตอล สัญญาณที่ขับได้ มี 2 สถานะคือ HIGH หรือลอจิกสูง หรือ "1" กับ LOW หรือลอจิกต่ำ หรือ "0"

รูปแบบคำสั่ง

GPIO.output(ขาพอร์ต, สถานะ)

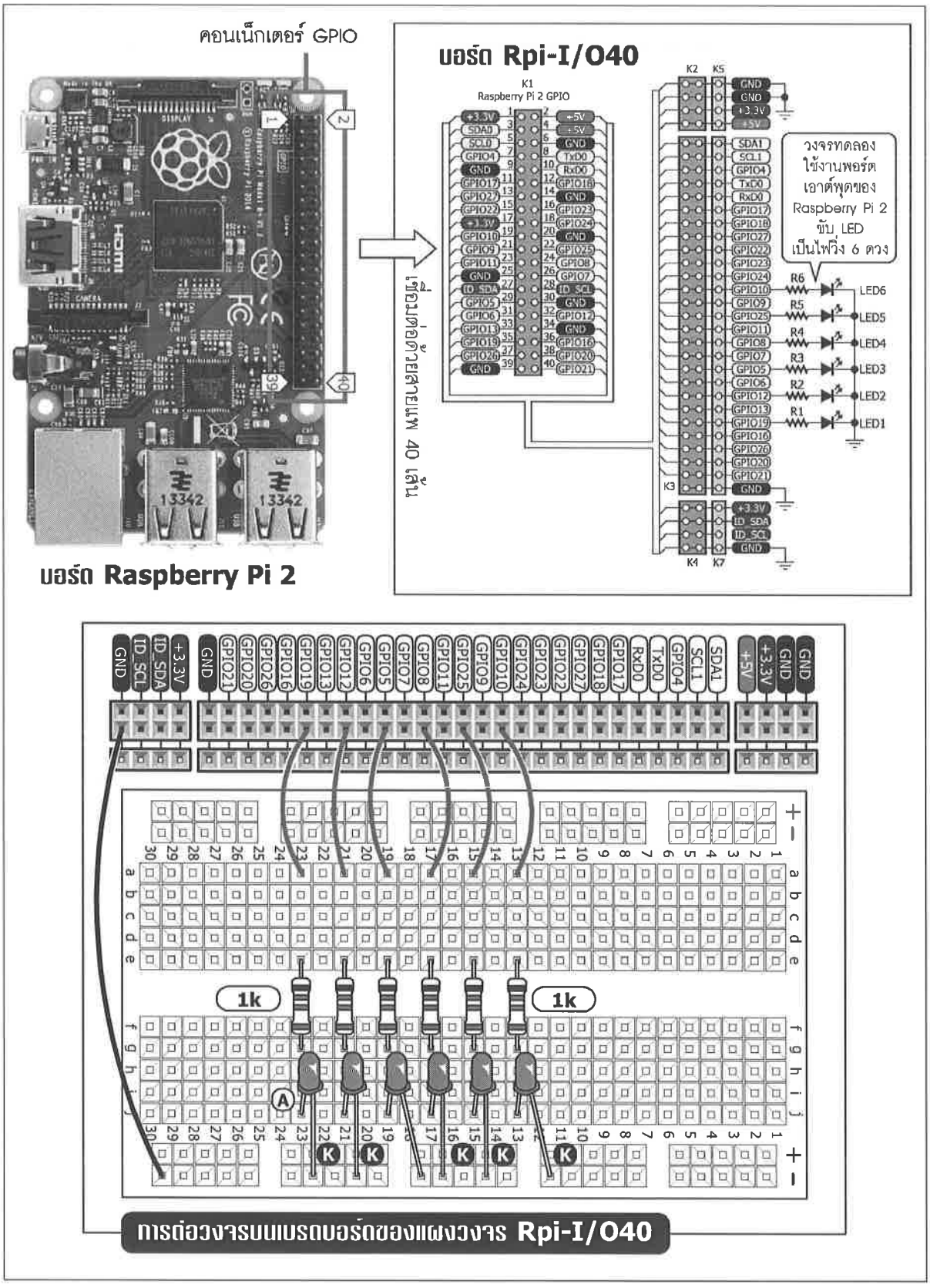
ตัวอย่างที่ 7-1

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

กำหนดให้ขา GPIO19 ขับสัญญาณลอจิกสูง

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

กำหนดให้ขา GPIO2 ขับสัญญาณลอจิกต่ำ



รูปที่ 7-9 วงจรไฟวิ่ง 6 ดวงอย่างง่ายควบคุมด้วยขา GPIO ของ Raspberry Pi 2

7.4.2.1 ทดลองสร้างไฟวิ่ง LED 6 ดวงอย่างง่าย

(1) ต่อวงจรตามรูปที่ 7-9

(2) เปิดซอฟต์แวร์ Geany แล้วสร้างไฟล์สคริปต์ จากนั้นพิมพ์คำสั่งตามโปรแกรมที่ 7-2

```
import RPi.GPIO as GPIO
import time

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

pin1 = 19
pin2 = 12
pin3 = 5
pin4 = 8
pin5 = 25
pin6 = 10

GPIO.setup(pin1, GPIO.OUT)
GPIO.setup(pin2, GPIO.OUT)
GPIO.setup(pin3, GPIO.OUT)
GPIO.setup(pin4, GPIO.OUT)
GPIO.setup(pin5, GPIO.OUT)
GPIO.setup(pin6, GPIO.OUT)

GPIO.output(pin1, False)
GPIO.output(pin2, False)
GPIO.output(pin3, False)
GPIO.output(pin4, False)
GPIO.output(pin5, False)
GPIO.output(pin6, 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)
        GPIO.output(pin4, GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(pin4, GPIO.LOW)
        GPIO.output(pin5, GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(pin5, GPIO.LOW)
        GPIO.output(pin6, GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(pin6, GPIO.LOW)
```

โปรแกรมที่ 7-2 โค้ดภาษา Python สำหรับใช้งานขาพอร์ต GPIO ของบอร์ด Raspberry Pi 2 เป็นขาพอร์ตเอาต์พุตดิจิตอลเพื่อขับ LED 6 ดวง

(3) ทำการบันทึกไฟล์ LED6Scrolling.py

(4) เอ็กซิคิวต์ด้วย Python3 โดยไปที่เมนู **Bulid > Execute** หรือรันด้วยคำสั่ง **sudo python3 LED6Scrolling.py**

เมื่อโปรแกรมเริ่มทำงาน LED ทั้ง 6 ดวงจะติดดับในลักษณะไฟวิ่ง ถ้าต้องการให้หยุดทำงาน กด Ctrl และ C (กดคีย์ Ctrl ค้างไว้ แล้วกดคีย์ C ตาม)

7.4.2.2 สร้างไฟวิ่ง LED 6 ดวงแบบกระชับ

(1) ใช้วงจรในรูปที่ 7-9 ในการทดลอง

(2) เปิดซอฟต์แวร์ Geany แล้วสร้างไฟล์สคริปต์ จากนั้นพิมพ์คำสั่งตามโปรแกรมที่ 7-3

(3) ทำการบันทึกไฟล์ชื่อ LED6Scrolling-Short.py

(4) เอ็กซิคิวต์ด้วย Python3 โดยไปที่เมนู **Bulid > Execute** หรือรันผ่านเทอร์มินอลด้วยคำสั่ง **sudo python3 LED6Scrolling-Short.py**

เมื่อโปรแกรมเริ่มทำงาน LED ทั้ง 6 ดวงจะติดดับในลักษณะไฟวิ่งเหมือนกับโปรแกรม LED6Scrolling แต่ถ้าพิจารณาจากโค้ดแล้ว จะพบว่า โปรแกรม LED6Scrolling-Short จะสั้นและกระชับกว่า

```
import RPi.GPIO as GPIO
import time

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

pin_set = [19, 12, 5, 8, 25, 10]

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 7-3 โค้ดภาษา Python สำหรับใช้งานขาพอร์ต GPIO ของบอร์ด Raspberry Pi 2 เป็นขาพอร์ตเอาต์พุตดิจิตอลเพื่อขับ LED 6 ดวง ที่ปรับปรุงจากโปรแกรมที่ 7-2 ให้มีความกระชับมากขึ้น

7.4.3 พอร์ตอินพุตดิจิตอล

เมื่อต้องการสั่งงานให้ขา GPIO รับสัญญาณดิจิตอลจากภายนอก ต้องกำหนดให้ GPIO ทำงานเป็นพอร์ตออินพุตสัญญาณที่รับได้มี 2 สถานะคือ HIGH หรือลอจิกสูง หรือ "1" กับ LOW หรือลอจิกต่ำ หรือ "0" โดยพอร์ตอินพุตดิจิตอลมีวิธีรับสัญญาณได้ 2 แบบใหญ่ๆ คือ **แบบซิงโครนัส (synchronous) และอะซิงโครนัส (asynchronous)**

7.4.3.1 พอร์ตอินพุตดิจิตอลแบบซิงโครนัส

การใช้งานพอร์ตอินพุตดิจิตอลแบบซิงโครนัสมี 2 วิธีคือ

วิธีที่ 1 เป็นการอ่านค่าสถานะ ณ เวลานั้นของขา GPIO เมื่ออ่านค่าได้แล้ว จึงกระทำคำสั่งต่อไป

วิธีที่ 2 เป็นการหยุดรอจนกว่าสัญญาณจะตรงกับที่กำหนดไว้ จึงกระทำคำสั่งต่อไป

(ก) ใช้งานพอร์ตอินพุตดิจิตอลแบบซิงโครนัสด้วยวิธีที่ 1

ในวิธีนี้จะเป็นการอ่านค่าสถานะในปัจจุบันรของขา GPIO เมื่ออ่านค่าได้แล้ว จึงกระทำคำสั่งต่อไป ต้องใช้คำสั่งต่อไปนี้ในการกำหนดลักษณะการทำงาน

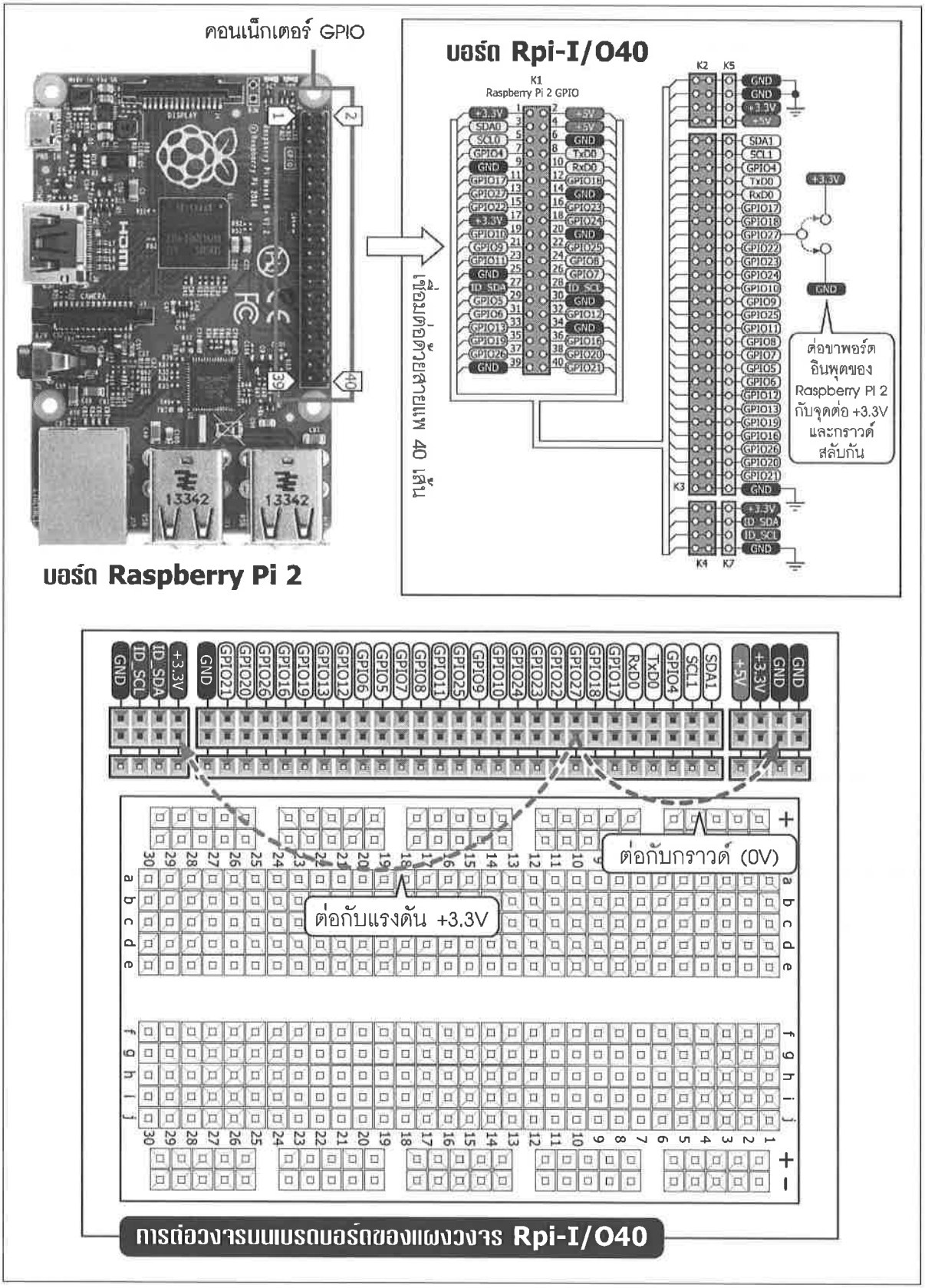
```
state = GPIO.input(ขาพอร์ต)
```

ตัวอย่างที่ 7-2

(1) ต่อวงจรตามรูปที่ 7-10

```
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 7-4 โค้ดภาษา Python ไฟล์ SynchronousInput1.py สำหรับอ่านค่าสถานะของขาพอร์ต GPIO ของบอร์ด Raspberry Pi 2 เมื่อทำงานเป็นขาพอร์ตอินพุต

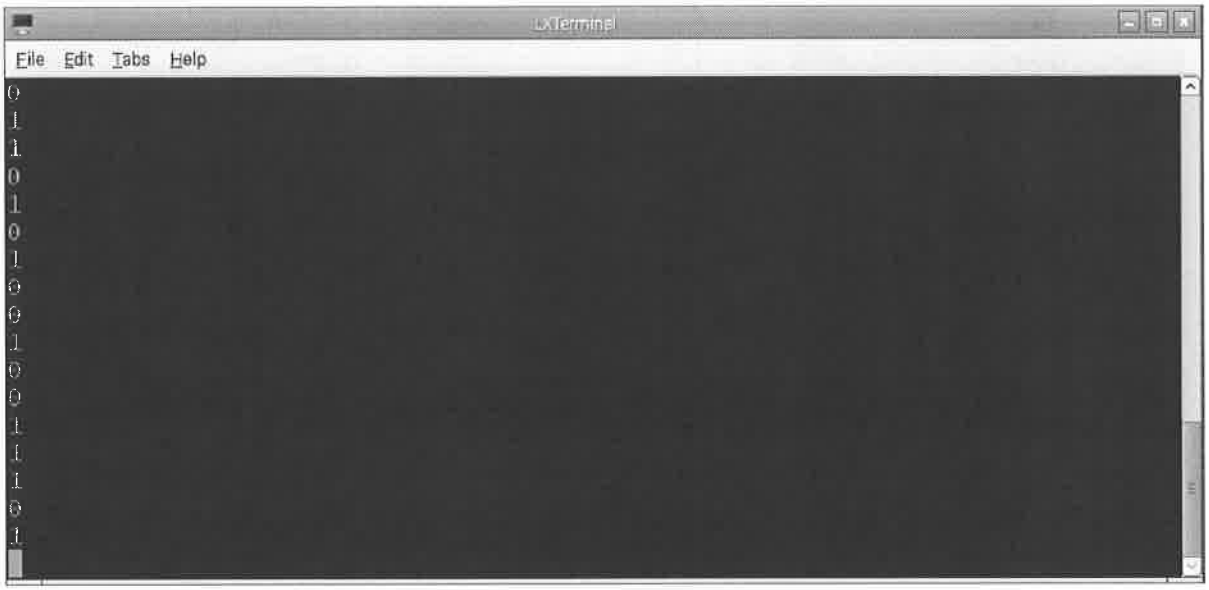


รูปที่ 7-10 การทดลองป้อนแรงดันเข้าที่ขาพอร์ต GPIO27 ของ Raspberry Pi 2 เพื่อทดสอบการทำงานของพอร์ตอินพุต

(2) เข้าสู่โปรแกรม Geany แล้วเขียนโปรแกรมที่ 7-4

(3) บันทึกไฟล์ SynchronousInput1.py ทำการคอมไพล์และเอ็กซิคิวต์ เพื่อทดสอบการทำงาน

โปรแกรมก็จะคอยอ่านค่าสถานะของขา GPIO27 ว่า มีสถานะเป็นอย่างไร ถ้ามีสถานะเป็น High จะแสดงค่าออกมาเป็นเลข 1 และถ้าเป็น Low ก็จะแสดงเป็น 0 โดยจะอ่านค่าทุกๆ 0.5 วินาที แล้วแสดงบนหน้าต่างเทอร์มินอล อาจทดลองง่ายๆ ด้วยการต่อสายจากขา GPIO27 ไปยัง +3.3V สลับกับกราวด์ ดังรูปที่ 7-10 เมื่อขาพอร์ตต่อไฟเลี้ยง +3.3V จะอ่านค่าสถานะเป็น "1" หากต่อลงกราวด์จะแสดงสถานะ "0"

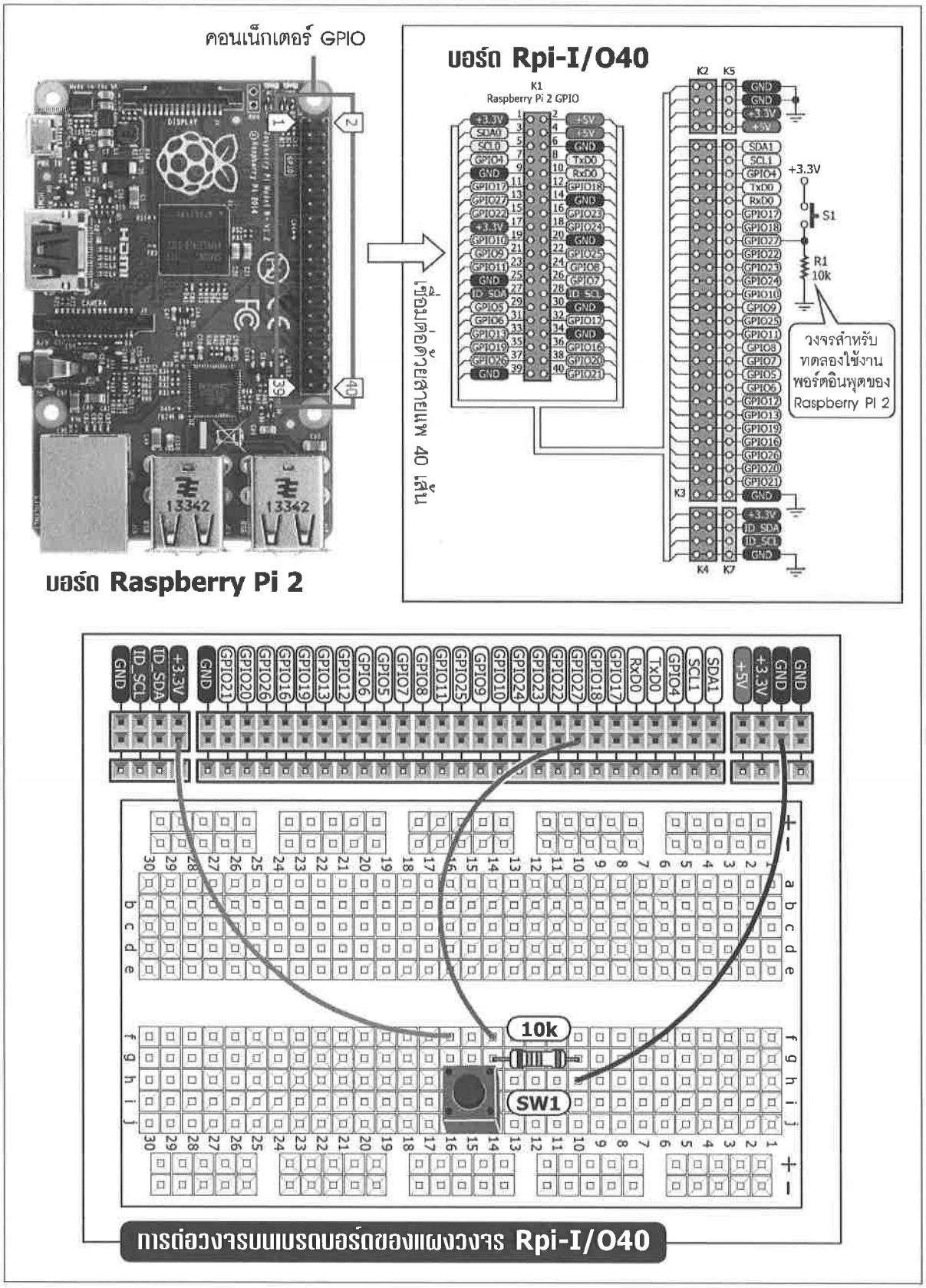


รูปที่ 7-11 ผลการทำงานของโปรแกรมที่ 7-4 เป็นการแสดงสถานะลอจิกของขา 27 ของพอร์ต GPIO ของบอร์ด Raspberry Pi 2 ที่ใช้งานเป็นขาพอร์ตอินพุต

(4) การแสดงผลลัพธ์เป็นเลข 0 และ 1 อาจจะดูเข้าใจยาก ดังนั้นจึงควรปรับเปลี่ยนการแสดงค่าให้เหมาะสมมากขึ้นด้วยการแสดงเป็นข้อความบอกว่า ปุ่มถูกกดหรือไม่ถูกกด จึงต้องต่อวงจรเพิ่มเติมดังรูปที่ 7-12

(5) เขียนโปรแกรมทดสอบเพิ่มเติมดังโปรแกรมที่ 7-5

(6) บันทึกไฟล์ชื่อ SynchronousInput2.py ทำการคอมไพล์และเอ็กซิคิวต์ เพื่อทดสอบการทำงาน



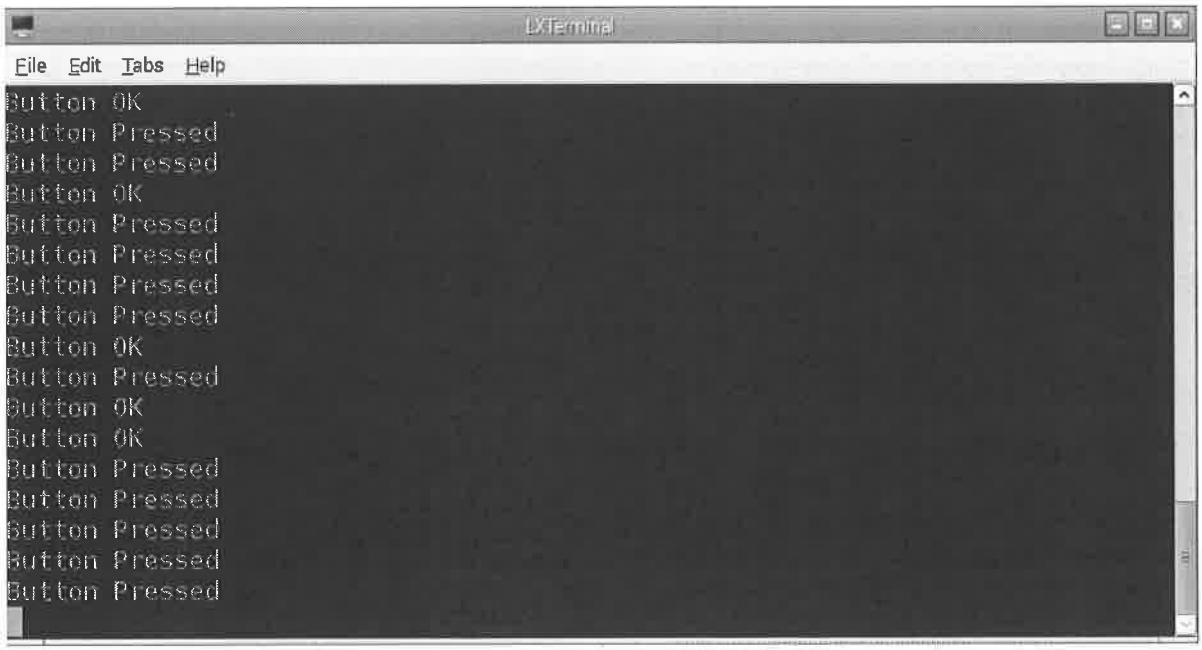
รูปที่ 7-12 วงจรทดสอบการทำงานพอร์ตอินพุตดิจิตอลของ GPIO ของ Raspberry Pi 2 โดยวงจรนี้เมื่อกดสวิตช์จะให้สถานะเป็น 1 ปล่อยสวิตช์จะมีสถานะเป็น 0

```
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 7-5 โค้ดภาษา Python ไฟล์ SynchronousInput2.py สำหรับใช้งานพอร์ต GPIO ของบอร์ด Raspberry Pi 2 เป็นอินพุตทำงานแบบซิงโครนัสในลักษณะที่เมื่ออ่านค่าได้แล้ว ให้กระทำคำสั่งต่อไป

(7) ทดลองกดสวิตช์แล้วปล่อยสลับกัน สังเกตผลการทำงานที่หน้าต่างเทอร์มินอล

เมื่อปุ่มหรือสวิตช์ไม่ถูกกดจะแสดงข้อความว่า Button OK และเมื่อปุ่มหรือสวิตช์ถูกกดจะแสดงข้อความว่า Button Pressed ที่จอแสดงผล ดังรูปที่ 7-13



รูปที่ 7-13 ผลการทำงานของโปรแกรมที่ 7-5 เป็นการแสดงสถานะลอจิกของขา GPIO27 ของพอร์ต GPIO ของบอร์ด Raspberry Pi 2 ที่ใช้งานเป็นขาพอร์ตอินพุตเมื่อมีการกดและปล่อยสวิตช์

(ข) ใช้งานพอร์ตอินพุตดิจิตอลแบบซิงโครนัสด้วยวิธีที่ 2

เป็นการรอจนกว่าค่าสัญญาณจะตรงกับที่กำหนดไว้ 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 7-3

(1) ใช้วงจรจากรูปที่ 7-12 ในการทดลอง เข้าสู่โปรแกรม Geany แล้วเขียนโปรแกรมที่ 7-6

(2) บันทึกไฟล์ SynchronousInputEdge.py ทำการคอมไพล์และเอ็กซิคิวต์โดยไปที่เมนู Bulid > Execute หรือรันผ่านเทอร์มินอลด้วยคำสั่ง **sudo python3 SynchronousInputEdge.py**

โปรแกรมนี้เป็นการรอสัญญาณขอบขาขึ้นที่ขา GPIO27 โดยจะหยุดรอที่คำสั่ง `GPIO.wait_for_edge` *เมื่อกดสวิตช์ สถานะที่ขาพอร์ตจะเป็น "1" หรือเกิดสัญญาณขอบขาขึ้น ทำให้เงื่อนไขเป็นจริง แสดงสถานะออกทางจอแสดงผล แล้วรอ 0.2 วินาทีก่อนวนกลับไปอ่านค่าใหม่อีกครั้ง*

```
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)` เป็นการกำหนดให้ขาพอร์ตที่เลือก ในที่นี้คือขา 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 7-6 โค้ดภาษา Python ไฟล์ SynchronousInputEdge.py สำหรับใช้งาน GPIO ของ Raspberry Pi 2 เป็นอินพุตทำงานแบบซิงโครนัสในลักษณะรอจนกว่าสถานะของขาพอร์ตตรงกับที่กำหนดไว้จึงกระทำคำสั่งต่อไป

7.4.3.2 พอร์ตอินพุตดิจิตอลแบบอะซิงโครนัส

การใช้งานพอร์ตอินพุตดิจิตอลแบบนี้เป็นการใช้อีเวนต์ (Event) หรือเหตุการณ์เข้ามาช่วย โดยไม่จำเป็นต้องคอยเขียนคำสั่งวนตรวจสอบค่าตลอดเวลา เพียงกำหนดอีเวนต์หรือเหตุการณ์ที่จะเกิดขึ้น และเมื่อเกิดอีเวนต์ตามที่กำหนดไว้ โปรแกรมจะทำงานตามชุดคำสั่งในส่วนนั้นให้ทันที

ตัวอย่างที่ 7-4

(1) ยังคงใช้วงจรในรูปที่ 7-12 ในการทดลอง เข้าสู่โปรแกรม Geany แล้วเขียนโปรแกรมที่ 7-7

(2) บันทึกไฟล์ AsynchronousInput.py ทำการคอมไพล์และเอ็กซิคิวต์ เพื่อทดสอบการทำงาน

ผลการทำงานจะเหมือนกับโปรแกรมที่ 7-6 หากแต่ลักษณะของโปรแกรมแตกต่างกัน โดยในโปรแกรมที่ 7-7 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

ในโปรแกรมนี้สร้างฟังก์ชั่นที่ชื่อว่า `button_pressed` เตรียมไว้ จากนั้นกำหนดขา GPIO27 ให้เป็นพอร์ตอินพุตดิจิตอล และกำหนดให้มีการต่อพูลดาวน์ไว้ จากนั้นกำหนดเงื่อนไขให้กับขา GPIO27 ว่า ถ้ามีสัญญาณที่เป็นขอบขาขึ้น (Rising) ก็จะเรียกให้ฟังก์ชั่น button_pressed ทำงาน และกำหนดระยะเวลาสำหรับดีเบาซ์ (debounce) หรือการแก้สัญญาณรบกวนจากการกดสวิตช์ไว้ที่ 300 มิลลิวินาที

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

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

7.4.4 ตัวอย่างการใช้งานพอร์ตอินพุตเอาต์พุต

7.4.4.1 สวิตช์ควบคุม LED

ในตัวอย่างนี้เป็นการกำหนดให้ Raspberry Pi 2 อ่านค่าจากขา GPIO27 ที่ต่อกับสวิตช์เพื่อนำค่ามาขับ LED ที่ต่อกับขา GPIO4

(1) ต่อวงจรตามรูปที่ 7-14

(2) สร้างไฟล์สคริปต์ด้วย Geanny หรือ nano เท็กซ์เอดิเอตร์

(3) พิมพ์คำสั่งตามโปรแกรมที่ 7-8

(4) ทำการบันทึกไฟล์ชื่อ SwitchLED.py

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

(6) สังเกตการทำงานของ LED และหน้าจอแสดงผลเมื่อไม่ได้กดสวิตช์ และหลังจากกดสวิตช์

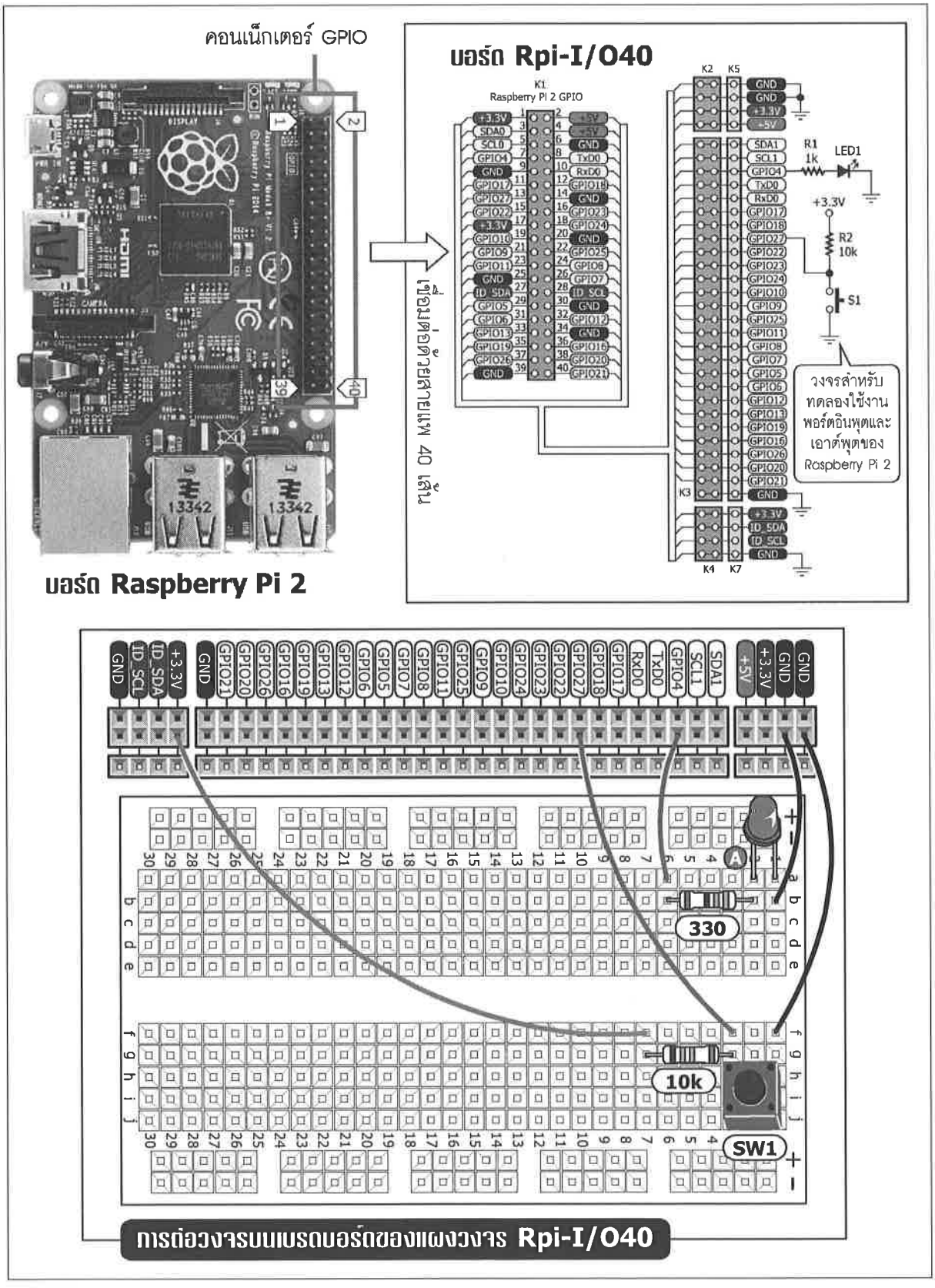
เมื่อยังไม่ได้กดสวิตช์ ที่ขา GPIO27 จะมีค่าเป็น 1 และส่งผลให้ ขา GPIO4 ที่กำหนดไว้เป็นเอาต์พุต มีสะถานะเป็น 1 ทำให้ LED ที่ต่อกับขา GPIO4 สว่าง และเมื่อกดสวิตช์ สถานะที่ขา GPIO27 จะมีค่าเป็น 0 ส่งผลให้ LED ที่ต่อกับขา GPIO4 ดับ และที่จอแสดงผลจะแสดงข้อความดังนี้

`SW=1` *หมายความว่าไม่ได้กดสวิตช์*

`SW=0` *หมายความว่า กดสวิตช์*

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(4,GPIO.OUT)
GPIO.setup(27,GPIO.IN)

while (1):
  state=GPIO.input(27)
  GPIO.output(4,state)
  print("SW=",state)
```

โปรแกรมที่ 7-8 โค้ดภาษา Python ไฟล์ SwitchLED.py สำหรับใช้งาน GPIO ของบอร์ด Raspberry Pi 2 ทั้งอินพุตและเอาต์พุตให้ทำงานร่วมกัน โดยอ่านค่าสถานะของสวิตช์จากพอร์ตอินพุตมาควบคุมการทำงานของ LED ที่ต่อกับขาพอร์ตเอาต์พุต



รูปที่ 7–14 วงจรทดสอบการทำงานของพอร์ต GPIO เบื้องต้นในลักษณะทำงานร่วมกันทั้งอินพุตและเอาต์พุต

7.4.4.2 สวิตช์ควบคุมไฟวิ่ง LED

ในตัวอย่างนี้เป็นการกำหนดให้ Raspberry Pi 2 อ่านค่าจากขา GPIO27 ที่ต่อกับสวิตช์เพื่อนำสถานะมาควบคุมการทำงานของไฟวิ่ง LED ที่ต่อกับขา GPIO4

(1) ต่อวงจรตามรูปที่ 7-15 จากนั้นสร้างไฟล์สคริปต์ด้วย Geanny หรือ Nano เท็กซ์เอดิเตอร์

(2) พิมพ์คำสั่งตามโปรแกรมที่ 7-9

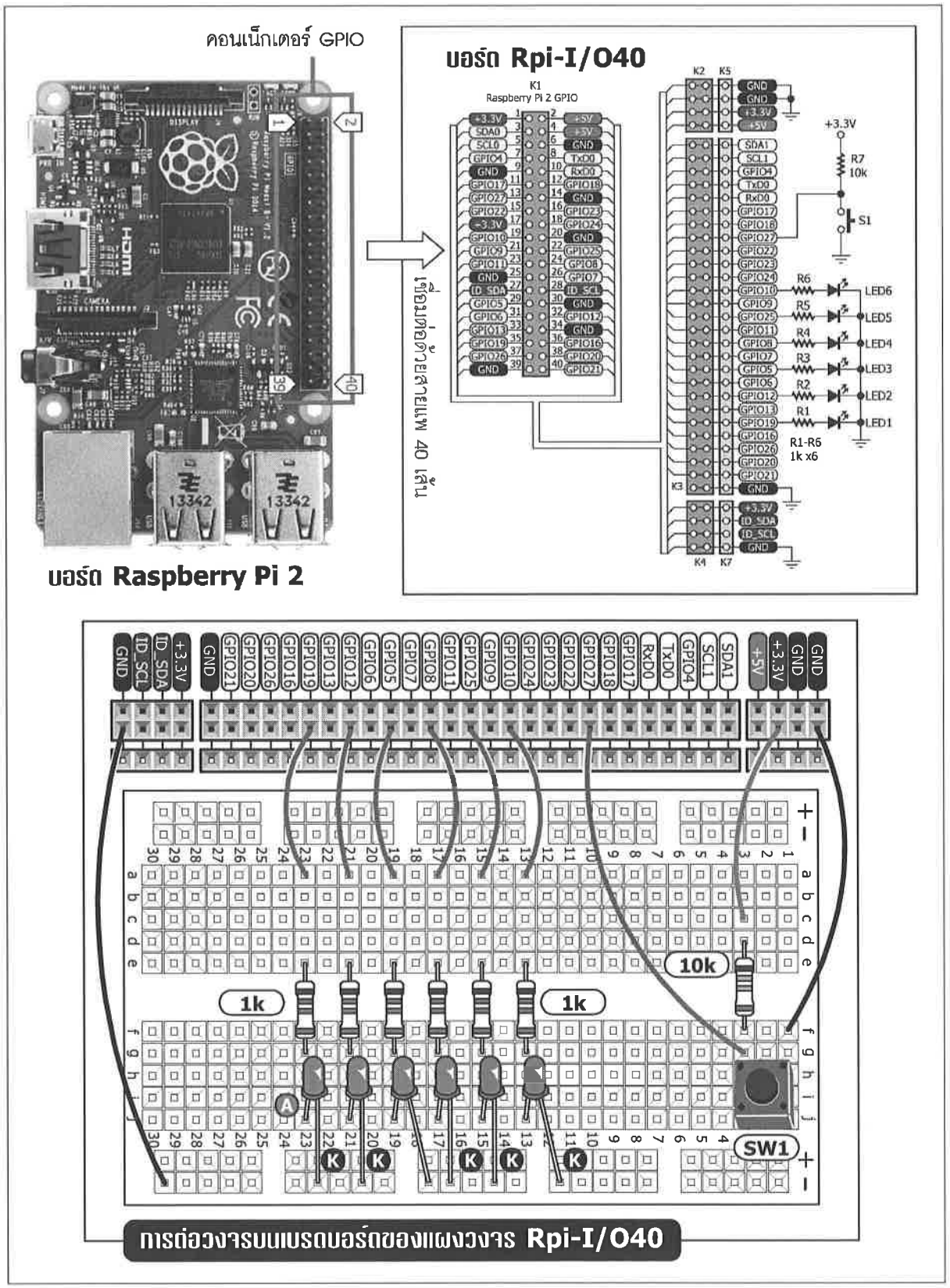
```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
my_pin=[19,12,5,8,25,10]
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 7-9 โค้ดภาษา Python ไฟล์ SwitchLED6Running.py สำหรับใช้งานพอร์ต GPIO ของบอร์ด Raspberry Pi 2 ทั้งอินพุตและเอาต์พุตให้ทำงานร่วมกัน โดยอ่านค่าสถานะของสวิตช์จากพอร์ตอินพุตมาควบคุมการทำงานของไฟวิ่ง LED 6 ดวงที่ต่อกับขาพอร์ตเอาต์พุต 6 ขา



รูปที่ 7–15 วงจรสวิตช์ควบคุมไฟวิ่ง 6 ดวง เป็นวงจรทดสอบการทำงานของพอร์ต GPIO เบื้องต้นในลักษณะทำงานร่วมกันทั้งอินพุตและเอาต์พุต

(3) ทำการบันทึกไฟล์ชื่อ SwitchLED6Running.py

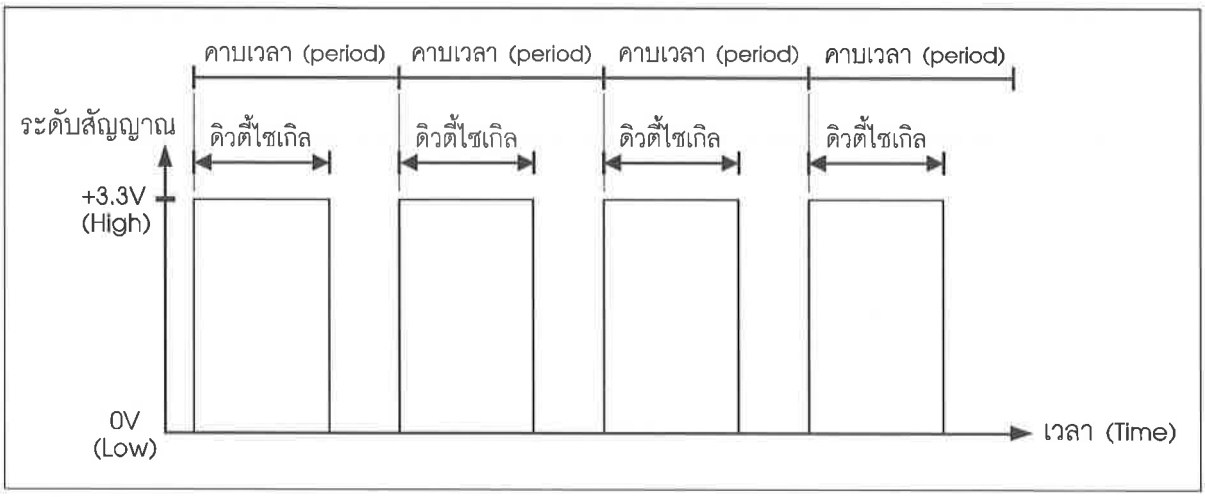
(4) สั่งให้ไฟล์สคริปต์ทำงาน โดยไปที่เมนู **Bulid > Execute** ในหน้าต่างของ Geany หรือรันไฟล์สคริปต์ด้วยเทอร์มินอล โดยใช้คำสั่ง `sudo python3 SwitchLED6Running.py`

(5) สังเกตการทำงานของ LED และหน้าจอแสดงผลเมื่อไม่ได้กดสวิตช์และหลังจากกดสวิตช์ค้างไว้

เมื่อกดสวิตช์ วงจรไฟวิ่ง LED ยังไม่ทำงาน ทันทีที่ปล่อยสวิตช์ วงจรไฟวิ่ง LED จะเริ่มทำงาน โดย LED ที่ต่อกับขา GPIO19 จะติดเป็นดวงแรกนาน 0.5 วินาทีแล้วดับลง LED ที่ต่อกับขา GPIO12 จะติดขึ้นมาแทน ไล่ไปตามลำดับ จนถึง LED ที่ต่อกับขา GPIO 10 จะหยุดทำงาน เพื่อรอให้กดและปล่อยสวิตช์ที่ต่อกับขา GPIO18 อีกครั้ง

7.5 การสร้างสัญญาณ PWM สำหรับ GPIO ของ Raspberry Pi 2

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



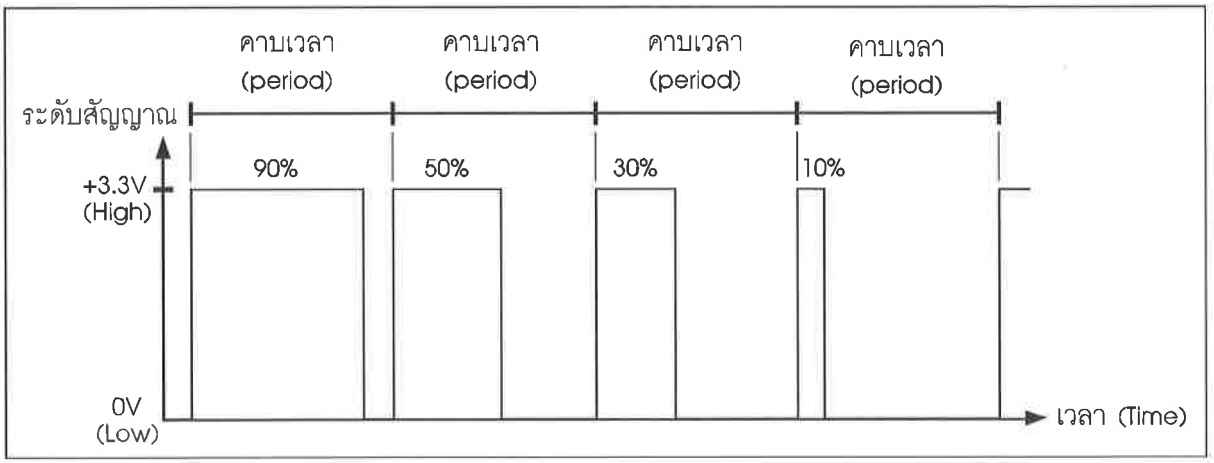
รูปที่ 7-16 ตัวอย่างของสัญญาณ PWM

7.5.1 แนะนำสัญญาณ PWM

ในรูปที่ 7-16 แสดงตัวอย่างสัญญาณ PWM ซึ่งมีลักษณะเป็นสัญญาณพัลส์ที่มีความกว้างค่าหนึ่งที่สามารถกำหนดได้ โดยมีค่าของคาบเวลาที่คงที่ เนื่องจากสัญญาณ PWM มีทั้งช่วงเวลาที่เป็นลอจิกสูงและต่ำสลับกัน มากน้อยขึ้นกับการกำหนดความกว้างของพัลส์ ซึ่งเรียกสัดส่วนของสัญญาณพัลส์บวก (ช่วงที่เป็นลอจิกสูง) กับคาบเวลาทั้งหมดของสัญญาณพัลส์ว่า **ดิวตี้ไซเกิล (duty cycle)** โดยคิดค่าดิวตี้ไซเกิลเป็นเปอร์เซ็นต์ของค่าความกว้างพัลส์ ทั้งหมด

เมื่อสัญญาณมีลักษณะเป็นพัลส์ดังรูปที่ 7-16 ก็จะทำให้แรงดันไฟตรงของสัญญาณ PWM นี้มีถูกเฉลี่ยตามอัตราส่วนระหว่างความกว้างพัลส์กับคาบเวลาของสัญญาณหรือดิวตี้ไซเกิล หากค่าของดิวตี้ไซเกิลมาก แรงดันไฟตรงเฉลี่ยของสัญญาณ PWM จะสูง ในทางตรงข้าม หากค่าดิวตี้ไซเกิลต่ำ แรงดันเฉลี่ยก็จะต่ำ

ในรูปที่ 7-17 แสดงให้เห็นถึงสัญญาณ PWM ที่มีค่าดิวตี้ไซเกิลต่างๆ กัน จะเห็นว่า คาบเวลาของสัญญาณจะคงที่ แต่ความกว้างของสัญญาณพัลส์บวกแตกต่างกัน ตัวอย่างจากรูปที่ 7-17 พัลส์ลูกซ้ายสุดมีค่าดิวตี้ไซเกิล 90% หมายถึง ความกว้างของพัลส์ช่วงบวกมีความกว้างเป็น 90% ของความกว้างทั้งหมด ดังนั้นแรงดันเฉลี่ยที่ได้เท่ากับ (90 x 3.3) /100 = 2.97V (ที่ไฟเลี้ยงวงจร +3.3V) และถ้าหากสัญญาณ PWM มีดิวตี้ไซเกิล 50% แรงดันเฉลี่ยจะเท่ากับ 1.65V ลดลงเหลือ 0.99V ที่ดิวตี้ไซเกิล 30% และ 0.33V ที่ดิวตี้ไซเกิล 10%



รูปที่ 7-17 แสดงตัวอย่างสัญญาณ PWM ที่มีค่าดิวตี้ไซเกิลต่าง ๆ กัน

ผู้พัฒนาสามารถปรับค่าดิวตี้ไซเกิล, คาบเวลา และความถี่ของสัญญาณ PWM ได้ตามจุดประสงค์ในการใช้งาน โดยมีการคำนวณเบื้องต้นดังนี้

ตัวอย่างที่ 7-5

สัญญาณ PWM มีคาบเวลา 20 มิลลิวินาที มีความกว้างพัลส์ 5 มิลลิวินาที จะมีดิวตี้ไซเกิลเท่ากับ 5 / 2 x 100% = 25%

ตัวอย่างที่ 7-6

สัญญาณ PWM ความถี่ 4kHz และมีดิวตี้ไซเกิล 18% คำนวณหาคาบเวลาและความกว้างของพัลส์ได้ดังนี้

คาบเวลา = 1/4000Hz = 0.25 มิลลิวินาทีหรือ 250 ไมโครวินาที

ความกว้างพัลส์ = 250 x 18% / 100% = 45 ไมโครวินาที

7.5.2 Raspberry Pi 2 กับการสร้างสัญญาณ PWM

ในการสร้างสัญญาณ PWM บน Raspberry Pi 2 นั้น ทำได้ง่ายๆ ด้วยคำสั่ง PWM ที่มีอยู่แล้วใน RPi.GPIO สิ่งที่ต้องทำคือ กำหนดขาที่ต้องการขับสัญญาณ PWM, ความถี่ และค่าดิวตี้ไซเกิล

ในซีพียู Broadcom BCM2836 ของ Raspberry Pi 2 มีฮาร์ดแวร์สร้างสัญญาณ PWM แล้ว แต่ในไลบรารี RPi.GPIO ไม่ได้เรียกใช้ โดยเลือกให้กำเนิดสัญญาณ PWM ด้วยกระบวนการทางซอฟต์แวร์แทน หรือเรียกว่า **PWM ทางซอฟต์แวร์ (Software PWM)** ซึ่งมีข้อดีคือ ทำให้ขาพอร์ต GPIO ทุกขาสร้างสัญญาณ PWM ได้

การกำหนดให้ขาพอร์ต GPIO ของ Raspberry Pi 2 เป็นขาเอาต์พุต PWM จะต้องเรียกใช้โมดูล RPi.GPIO และกำหนดให้ขาพอร์ตที่ต้องการใช้งานเป็นเอาต์พุตก่อน ด้วยคำสั่ง

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

ส่วนคำสั่งสำหรับการสร้างสัญญาณ PWM ประกอบด้วย

7.5.2.1 คำสั่งเลือกขาพอร์ตและความถี่

รูปแบบคำสั่ง

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

พารามิเตอร์

p คือ ตัวแปรใดๆ

pin คือ ขาพอร์ต GPIO

frequency คือ ความถี่ของสัญญาณ PWM ในหน่วย Hz

ตัวอย่างที่ 7-7

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

7.5.2.2 คำสั่งเริ่มต้นการสร้างสัญญาณ PWM ด้วยการกำหนดค่าดิวตี้ไซเกิล

รูปแบบคำสั่ง

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

พารามิเตอร์

p คือ ตัวแปรใดๆ

dutycycle คือ ค่าดิวตี้ไซเกิล มีค่า 0.0 ถึง 100.0

ตัวอย่างที่ 7-8

```
p.start = (0)
# ไม่มีสัญญาณออกจากขาพอร์ต เนื่องจากค่าดิวตี้ไซเกิล = 0%
p.start = (50)
# กำเนิดสัญญาณ PWM ที่มีค่าดิวตี้ไซเกิล 50%
```

7.5.2.3 คำสั่งเปลี่ยนค่าความถี่

รูปแบบคำสั่ง

```
p.ChangeFrequrency(frequency)
```

พารามิเตอร์

p คือ ตัวแปรใดๆ

frequency คือ ความถี่ของสัญญาณ PWM ในหน่วย Hz

ตัวอย่างที่ 7-9

```
p.ChangeFrequency(100)
# เปลี่ยนความถี่ของสัญญาณ PWM เป็น 100Hz
```

7.5.2.4 คำสั่งเปลี่ยนค่าดิวตี้ไซเกิล

รูปแบบคำสั่ง

```
p.ChangeDutyCycle(dutycycle)
```

พารามิเตอร์

p คือ ตัวแปรใดๆ

dutycycle คือ ค่าดิวตี้ไซเกิล มีค่า 0.0 ถึง 100.0

ตัวอย่างที่ 7-10

```
p.ChangeDutyCycle(90)
# เปลี่ยนค่าดิวตี้ไซเกิลเป็น 90%
```

7.5.2.5 คำสั่งหยุดการสร้างสัญญาณ PWM

รูปแบบคำสั่ง

```
p.stop()
```

พารามิเตอร์

p คือ ตัวแปรใดๆ

7.5.3 ตัวอย่างการสร้างสัญญาณ PWM ของ Raspberry Pi 2

7.5.3.1 ขับ LED กะพริบด้วยสัญญาณ PWM

(1) ต่อวงจรตามรูปที่ 7-18 โดยต่อ LED เข้าที่ขา GPIO25 โดยมีตัวต้านทานต่อจำกัดกระแสไฟฟ้าไม่ให้ไหลผ่าน LED มากเกินไป ค่าของตัวต้านทานใช้ได้ตั้งแต่ 300Ω จนถึง 1kΩ

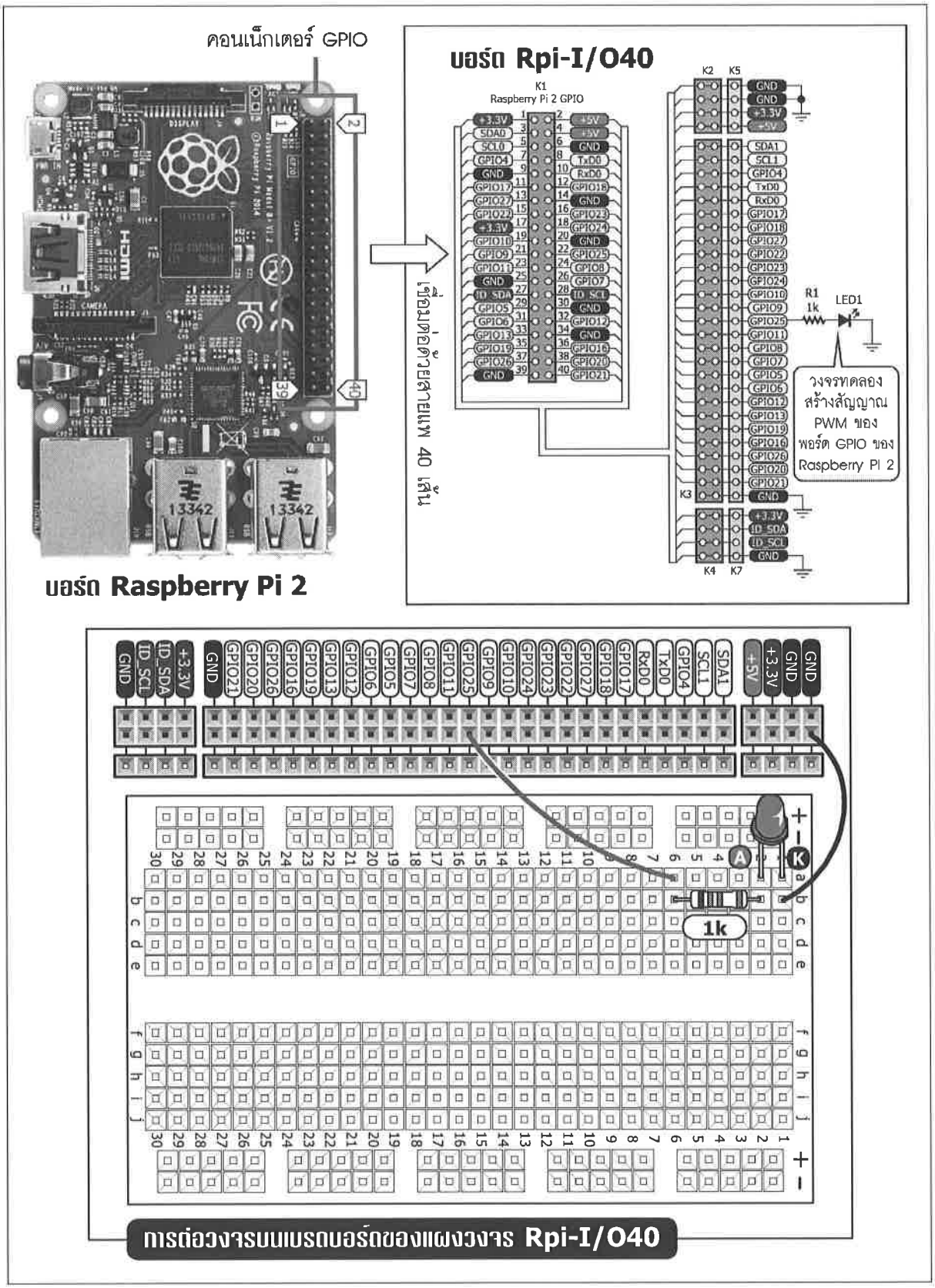
(2) สร้างไฟล์สคริปต์ด้วย Geanny หรือ nano เท็กซ์เอดิเตอร์ พิมพ์คำสั่งตามโปรแกรมที่ 7-10

(3) บันทึกไฟล์ชื่อ PWMblink.py แล้วทำการคอมไพล์และเอ็กซิคิวต์เพื่อทดสอบการทำงาน

โปรแกรมนี้จะสร้างสัญญาณ PWM ความถี่ 1Hz มีค่าดิวตี้ไซเกิล 50% ส่งออกไปยังขา GPIO25 โดยสัญญาณ PWM ที่ขับออกมานี้จะทำให้ LED ที่ต่อกับขา GPIO25 กะพริบทุกๆ 0.5 วินาที

```
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 7-10 โค้ดภาษา Python สำหรับใช้งานพอร์ต GPIO ของบอร์ด Raspberry Pi 2 สร้างสัญญาณ PWM



รูปที่ 7-18 วงจรทดลองสร้างสัญญาณ PWM ของบอร์ด Raspberry Pi 2 ผ่านทางขา GPIO25

7.5.3.2 ปรับความสว่างของ LED ด้วยสัญญาณ PWM

(1) ยังคงใช้วงจรตามรูปที่ 7-18 ในการทดลอง

(2) เข้าสู่โปรแกรม Geany หรือ nano เท็กซ์เอดิเตอร์ แล้วเขียนโปรแกรมที่ 7-11

(3) บันทึกไฟล์ชื่อ PWMLED.py แล้วทำการคอมไพล์และเอ็กซิคิวต์เพื่อทดสอบการทำงาน

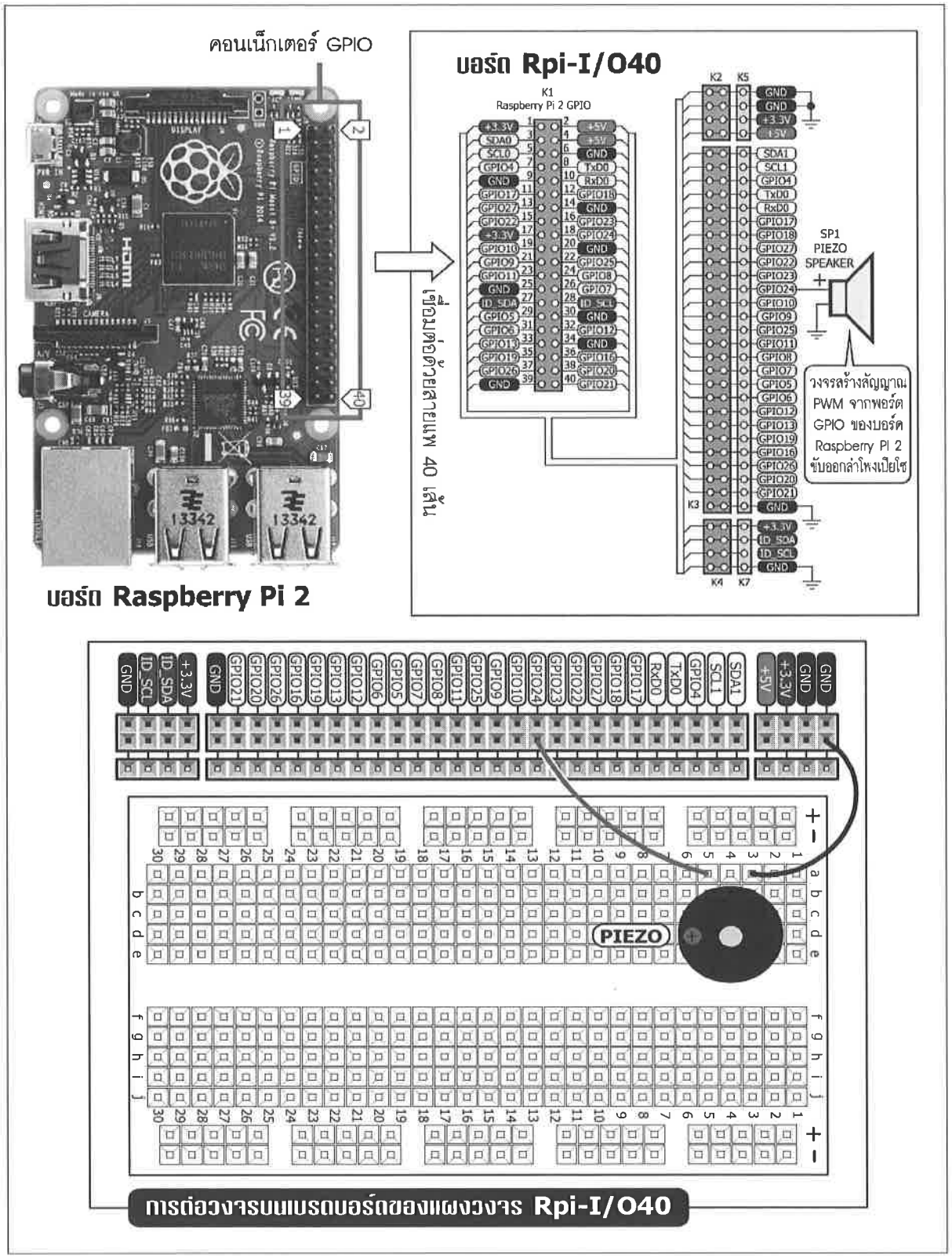
โปรแกรมนี้จะส่งสัญญาณ PWM ออกไปทางขา GPIO25 เพื่อขับให้ LED สว่างเพิ่มขึ้นเรื่อยๆ จนสว่างที่สุดที่ค่าดิวตี้ไซเกิล100% และเมื่อสว่างเต็มที่แล้ว สัญญาณ PWM จะลดค่าดิวตี้ไซเกิลลง ทำให้ LED ค่อยหรี่ๆ ลงจนดับ แล้ววนทำงานในลักษณะนี้ตลอดเวลา หากต้องการให้หยุดทำงานให้กด Ctrl และ C (กดคีย์ Ctrl ค้างไว้ แล้วกดคีย์ 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 7-11 โค้ดภาษา Python ไฟล์ PWMLED.py สำหรับใช้งาน GPIO ของบอร์ด Raspberry Pi 2 เพื่อสร้างสัญญาณ PWM สำหรับขับและควบคุมความสว่างของ LED

7.5.3.3 สร้างเสียงเตือนด้วย PWM

(1) ต่อวงจรตามรูปที่ 7-19



รูปที่ 7-19 วงจรทดลองสร้างสัญญาณ PWM จากพอร์ต GPIO ของ Raspberry Pi 2 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 7-12 โค้ดภาษา Python ไฟล์ PWMpiezo.py สำหรับใช้งานพอร์ต GPIO ของบอร์ด Raspberry Pi 2 เพื่อสร้างสัญญาณ PWM สำหรับขับเสียงออกทางลำโพงเปียโซ

(2) เข้าสู่โปรแกรม Geany หรือ nano เท็กซ์เอดิเตอร์ แล้วเขียนโปรแกรมที่ 7-12

(3) บันทึกไฟล์ชื่อ PWMpiezo.py

(4) สั่งให้ไฟล์สคริปต์ทำงาน โดยไปที่เมนู Bulid > Execute หรือรันไฟล์สคริปต์ด้วยเทอร์มินอลโดยใช้คำสั่ง sudo python3 PWMpiezo.py

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

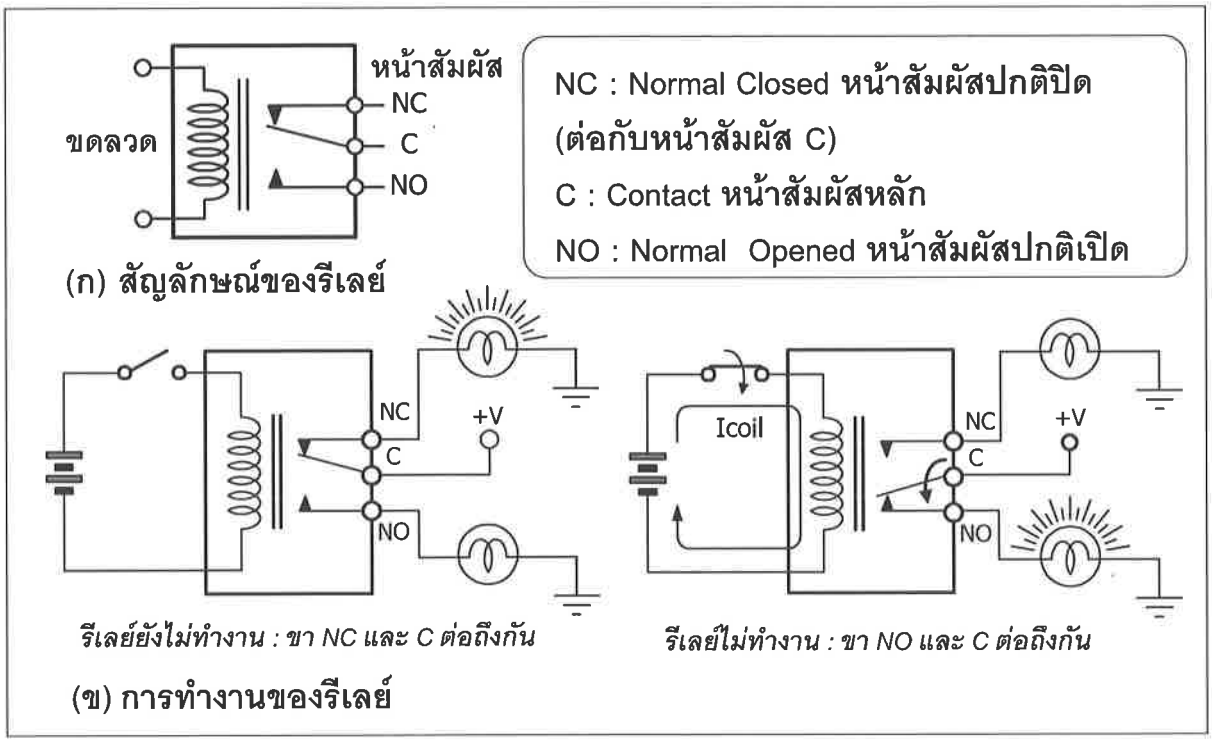
โดยค่าเวลานี้มาจากคำสั่ง `da=datetime.now()` *เมื่อค่าเวลาของพารามิเตอร์* `microsecond` *น้อยกว่า 700,000 จะทำให้คำสั่ง* `blink.ChangeDutyCycle(0)` *ได้ค่าของดิวตี้ไซเกิลเท่ากับ 0 จึงไม่มีการขับสัญญาณเสียงเกิดขึ้น เมื่อ ค่าเวลาของพารามิเตอร์* `microsecond` *มากกว่า 700,000 คำสั่ง* `blink.ChangeDutyCycle(50)`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

บทที่ 8

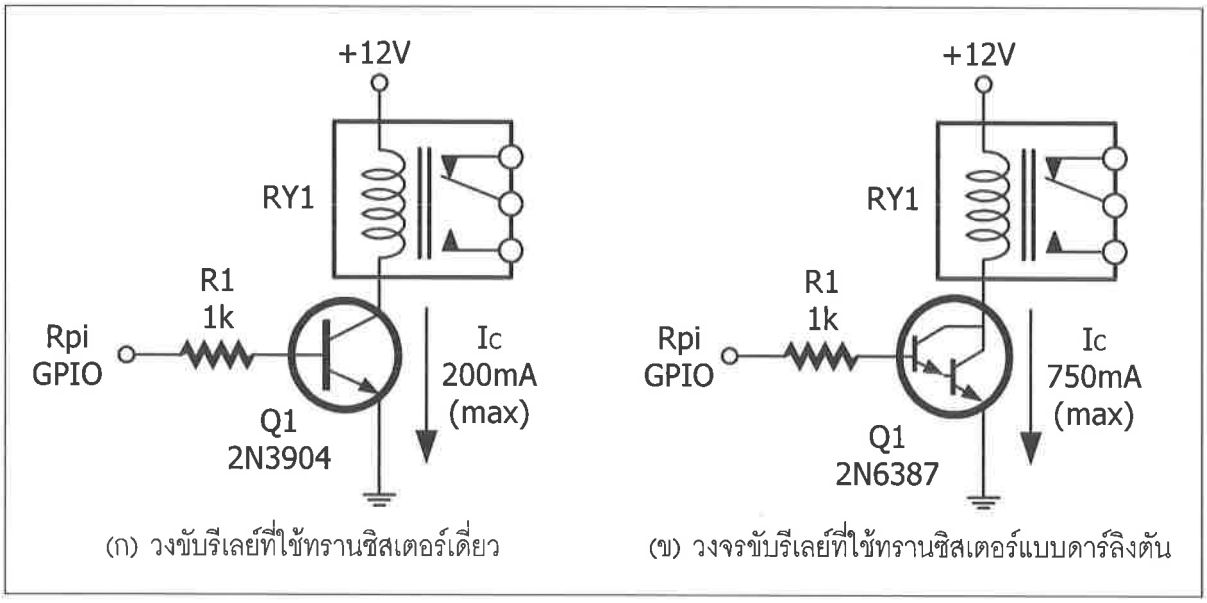
Raspberry Pi 2 กับการขับโหลดกระแสไฟฟ้าสูงด้วยรีเลย์

พอร์ต GPIO ของ Raspberry Pi 2 และรุ่นอื่นๆ มีความสามารถในการขับกระแสไฟฟ้าเมื่อทำงานเป็นพอร์ตเอาต์พุตได้สูงสุด 50mA ที่ +3.3V ซึ่งเพียงพอต่อกับโหลดกำลังไฟฟ้าต่ำ อาทิ LED หรือโมดูล LCD หากต้องการนำไปขับโหลดกำลังไฟฟ้าสูง เช่น หลอดไฟ หรือเครื่องใช้ไฟฟ้า จะต้องกระทำผ่านวงจรขับโหลดกระแสไฟฟ้าสูง ซึ่งมีอุปกรณ์หลักเป็นตัวช่วยคือ **รีเลย์**

ในบทนี้จะนำเสนอถึงข้อมูลเบื้องต้นและแนวทาง ตลอดจนตัวอย่างการเขียนโปรแกรมเพื่อควบคุมให้บอร์ด Raspberry Pi 2 ทำงานกับอุปกรณ์และวงจรขับโหลดกระแสไฟฟ้าสูงที่ใช้รีเลย์แบบกลไก (mechanical relay)



รูปที่ 9-1 แสดงสัญลักษณ์และการทำงานเบื้องต้นของรีเลย์



(ก) วงขับรีเลย์ที่ใช้ทรานซิสเตอร์เดี่ยว (ข) วงจรขับรีเลย์ที่ใช้ทรานซิสเตอร์แบบดาร์ลิงตัน

รูปที่ 8-2 วงจรขับรีเลย์โดยใช้ทรานซิสเตอร์

8.1 ความรู้เบื้องต้นเกี่ยวกับรีเลย์แบบกลไก

เป็นอุปกรณ์แม่เหล็กไฟฟ้าแบบหนึ่งที่ทำหน้าที่เป็นสวิตช์ตัดต่อหนึ่งชุดหรือมากกว่า ขึ้นอยู่กับจำนวนหน้าสัมผัสที่รีเลย์ตัวหนึ่ง ๆ บรรจุอยู่ รีเลย์มีสัญลักษณ์ตามรูปที่ 8-1 (ก) จะเห็นว่ารีเลย์ประกอบด้วยส่วนสำคัญ 2 ส่วนคือ ขดลวด (coil) และหน้าสัมผัส (contact) แบ่งเป็นหน้าสัมผัสปกติ (Normally Closed :NC) และปกติเปิดวงจรหรือไม่ต่อ (Normally Opened :NC)

การกระตุ้นให้รีเลย์ทำงานทำได้ง่ายมาก เพียงจ่ายแรงดันให้แก่ขดลวดในปริมาณที่ขดลวดนั้นต้องการ ก็จะทำให้แม่เหล็กไฟฟ้าเกิดขึ้นที่หน้าสัมผัส เกิดการดูดหน้าสัมผัสจากจุด NC มายังจุด NO ดังนั้นเมื่อรีเลย์ทำงานหน้าสัมผัส NO จะต่อวงจร ในขณะที่ NC จะเปิดวงจรแทน ในลักษณะนี้ทำงานเหมือนเป็นสวิตช์ 2 ทางที่ควบคุมด้วยแม่เหล็กไฟฟ้า ดังแสดงการทำงานในรูปที่ 8-1 (ข)

คุณสมบัติที่สำคัญของรีเลย์ได้แก่

1. แรงดันตกคร่อมขดลวดที่ทำให้รีเลย์ทำงาน (Vcoil หรือ Coil Voltage)

2. ค่าความต้านทานของขดลวด (Coil resistance) ปกติมีค่าประมาณ 100 ถึง 600W

3. อัตราทนได้สูงสุดทั้งแรงดันและกระแสไฟฟ้าของหน้าสัมผัส (Contact rating)

4. อายุการใช้งาน (Operating time)

5. ตำแหน่งขาของหน้าสัมผัส NO, NC และ C รวมทั้งขาต่อใช้งานของขดลวด

8.2 วงจรขับรีเลย์

ในการทำงานปกติ พอร์ตเอาต์พุตของไมโครคอนโทรลเลอร์ไม่สามารถนำไปขับอุปกรณ์เอาต์พุตกระแสไฟฟ้าสูงได้โดยตรงได้ เนื่องจากข้อจำกัดด้านความสามารถในการจ่ายกระแสไฟฟ้า ดังนั้น ถ้าต้องการนำไมโครคอนโทรลเลอร์ไปขับโหลดกระแสไฟฟ้าสูงจะต้องมีอุปกรณ์ที่ทำหน้าที่จ่ายแรงดันและกระแสสูงโดยเฉพาะ เรียกอุปกรณ์เหล่านี้ว่า **อุปกรณ์ขับ** หรือ **ไดรเวอร์ (driver)**

8.2.1 การใช้ทรานซิสเตอร์ขับรีเลย์

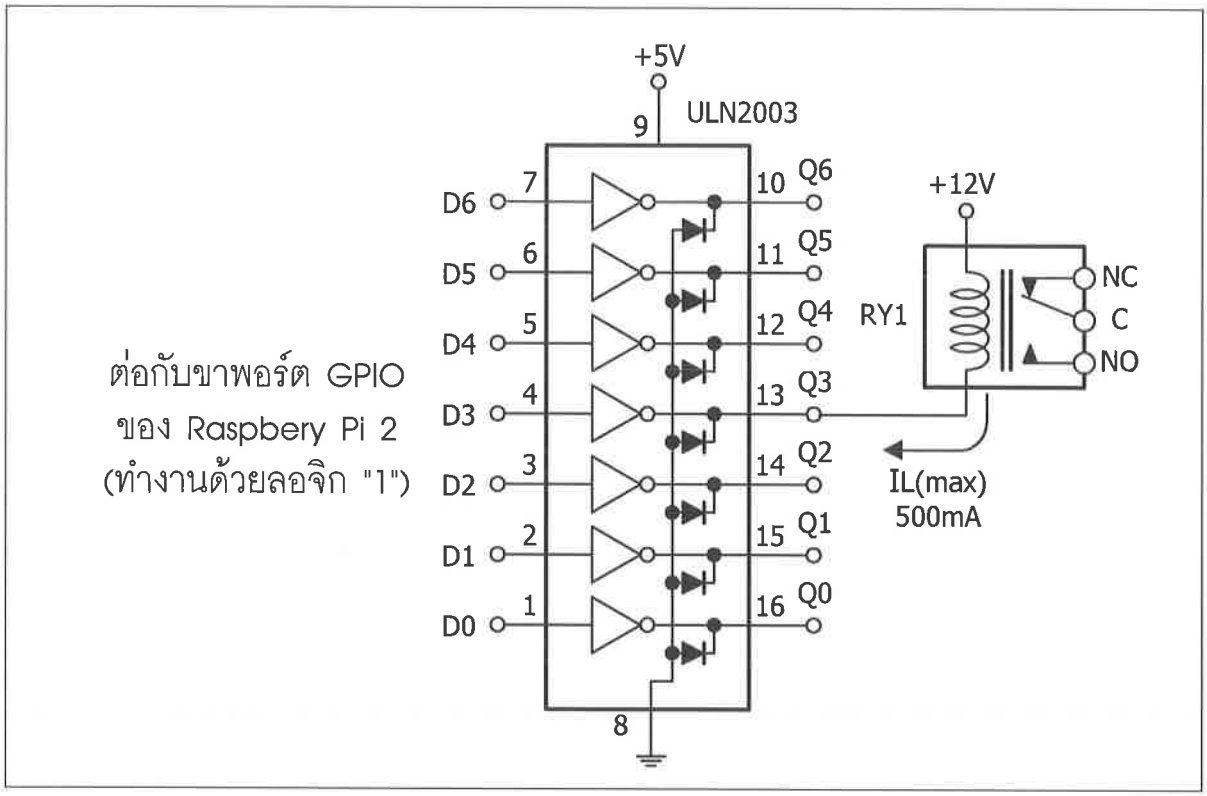
การขับโดยวิธีนี้เหมาะสมสำหรับโหลดที่มีความต้องการกระแสไฟฟ้าปานกลาง ตั้งแต่ 30 ถึง 200mA อาทิ รีเลย์กำลังต่ำไปจนถึงปานกลางที่มีค่าความต้านทานของขดลวดภายในรีเลย์ไม่ต่ำกว่า 100?, หลอดไฟกำลังต่ำ และมอเตอร์ไฟตรงขนาดเล็ก มีวงจรตัวอย่างตามรูปที่ 8-2

ในรูปที่ 8-2 (ก) เป็นการต่อทรานซิสเตอร์เข้ากับขาพอร์ต GPIO ของ Rapberry Pi โดยมีตัวต้านทาน R1 ทำหน้าที่จำกัดกระแสไฟฟ้าที่ไหลเข้าขาเบสของทรานซิสเตอร์ Q1 ซึ่งจะทำงานก็ต่อเมื่อขาพอร์ตจของไมโครคอนโทรลเลอร์มีสถานะลอจิกเป็น "1" เมื่อ Q1 ทำงาน ก็จะเกิดกระแสไฟฟ้าไหลผ่าน RL ซึ่งเป็นโหลดต่ออยู่ทางเอาต์พุตที่ขาคอลเล็กเตอร์ของ Q1 กระแสโหลดสูงสุด (ILmax) มีค่าเท่ากับ 12V/300??= 40mA ถึงแม้ว่า Q1 เบอร์ 2N3904 มีค่ากระแสคอลเล็กเตอร์สูงสุดถึง 100mA แต่ในทางปฏิบัติจริงไม่ควรออกแบบให้ทรานซิสเตอร์ทำงานถึงพิกัดสูงสุด ย่านปลอดภัยของทรานซิสเตอร์ควรอยู่ไม่เกินครึ่งหนึ่งของอัตราการทนได้สูงสุด ด้วยการจัดวงจรตามรูปที่ 9-2 สามารถใช้สัญญาณจากพอร์ตเอาต์พุตกระตุ้นให้ทรานซิสเตอร์ทำงานเพื่อขับรีเลย์ขนาดเล็กได้อย่างปลอดภัย

ในรูปที่ 8-2 (ข) เป็นวงจรขับรีเลย์ที่ใช้ทรานซิสเตอร์ดาร์ลิงตัน ที่บรรจุทรานซิสเตอร์ 2 ตัวต่อกันแบบดาร์ลิงตันภายใต้ตัวถังเดียวกัน เพื่อขับกระแสไฟฟ้าทางเอาต์พุตให้สูงขึ้น และมีความเร็วในการทำงานสูงด้วย ด้วยการใช้อุปกรณ์เพียงตัวเดียว ส่งผลให้ขนาดของวงจรเล็กลง จากวงจร Q1 ซึ่งเป็นทรานซิสเตอร์แบบดาร์ลิงตันสามารถขับกระแสไฟฟ้าทางเอาต์พุตได้สูงถึง 750mA โดยผ่านตัวต้านทานจำกัดกระแสเพียงตัวเดียวและไม่ต้องต่อทรานซิสเตอร์แบบคาสเคด ทำให้มีความเร็วในการทำงานสูง ตลอดจนสามารถขับกระแสไฟฟ้าทางเอาต์พุตได้สูงพอสมควร

8.2.2 การใช้ไอซีขับ

ไอซีที่ใช้ในการขับโหลดกระแสสูงมักจะมีวงจรทางเอาต์พุตเป็นแบบคอลเล็กเตอร์เปิด ทำให้ใช้กับแรงดันไฟฟ้าที่สูงได้ สำหรับไอซีขับหรือไอซีไดรเวอร์ที่ยกมาอธิบายคือเบอร์ ULN2003 เป็นไอซีที่ภายในบรรจุอินเวอร์เตอร์เกต 7 ตัว มีรูปแบบการจัดขาและวงจรภายในแสดงในรูปที่ 8-3 ใช้กับแรงดันได้สูงสุด +30V กระแสไฟฟ้าเอาต์พุตสูงสุดในแต่ละขาเท่ากับ 500mA ทั้งนี้ขึ้นอยู่กับความสามารถในการจ่ายกระแสไฟฟ้าของแหล่งจ่ายไฟด้วย



รูปที่ 8-3 วงจรขับรีเลย์โดยใช้ไอซีขับโหลดกระแสไฟฟ้าสูงเบอร์ ULN2003

นอกจากนั้น ULN2003 ยังมีการต่อไดโอดป้องกันแรงดันย้อนกลับจากอุปกรณ์เอาต์พุตที่มีโครงสร้างเป็นขดลวดไว้ที่ทุกขาเอาต์พุต ทำให้ใช้ขับโหลดที่เป็นขดลวด อาทิ รีเลย์ หรือมอเตอร์ไฟตรงขนาดเล็กถึงขนาดกลางได้

การกระตุ้นให้ทำงาน ทำได้ง่ายๆ โดยต่อขาพอร์ต GPIO ของ Rapberry Pi เข้าที่อินพุตของ ULN2003 โดยตรง หากต้องการให้วงจรขับชุดนั่นๆ ทำงาน เพียงส่งสัญญาณลอจิกสูง หรือ “1” มายังขาอินพุตของ ULN2003 วงจรขับที่ได้รับสัญญาณลอจิก “1” จะทำงาน กลับลอจิกที่เอาต์พุตเป็น “0” ทำให้เกิดกระแสไฟฟ้าไหลผ่านโหลดที่เป็นขดลวงรีเลย์ได้ รีเลย์ก็จะทำงาน และจะหยุดทำงานเมื่ออินพุตของ ULN2003 ได้รับลอจิกต่ำหรือ “0”

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

8.3 แนะนำ RELAY4i บอร์ดขับรีเลย์ 4 ช่อง

RELAY4i เป็นแผงวงจรขับรีเลย์ 4 ช่อง ราคาประหยัด มีคุณสมบัติทางเทคนิคโดยสรุปและหน้าตาของบอร์ดแสดงในรูปที่ 8-4 ส่วนในรูปที่ 8-5 แสดงวงจรสมบูรณ์ของบอร์ด RELAY4i

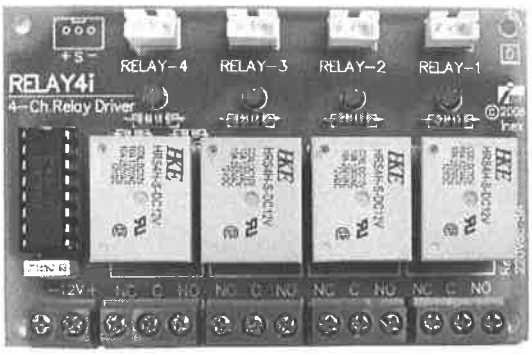
การใช้งานบอร์ดขับรีเลย์ RELAY4i สรุปเป็นขั้นตอนได้ดังนี้ (ใช้รูปที่ 8-6 อ้างอิง)

(1) ต่อโหลดที่ต้องการควบคุมเข้าที่จุดต่อหน้าสัมผัสของรีเลย์ ซึ่งมี 4 ช่อง แต่ละช่องเลือกให้ทำงานแบบต่อหรือตัดวงจรก็ได้ ปกติแล้วจะเลือกใช้งานแบบต่อวงจรมากกว่า นั่นคือ เมื่อรีเลย์ทำงานจะเป็นการต่อวงจรเพื่อจ่ายไฟเลี้ยงไปยังโหลดหรืออุปกรณ์ไฟฟ้าเพื่อให้ทำงานต่อไป จากรูปที่ 8-6 จะเห็นว่า ผู้ใช้งานสามารถต่อหน้าสัมผัสรีเลย์เข้ากับเครื่องใช้ไฟฟ้าได้สูงสุด 220Vac 600W (วัตต์) โดยต่อผ่านเต้าเสียบ ในขณะที่อีกช่องหนึ่งนั้นจะต่อกับหลอดไฟ 12V ในแต่ละช่องของหน้าสัมผัสรีเลย์ต่อกับโหลดได้ทั้งแบบไฟฟ้ากระแสตรงหรือกระแสสลับ รวมถึงการต่อวงจรเพื่อทำหน้าที่เป็นเหมือนสวิตช์ธรรมดาก็สามารถทำได้

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

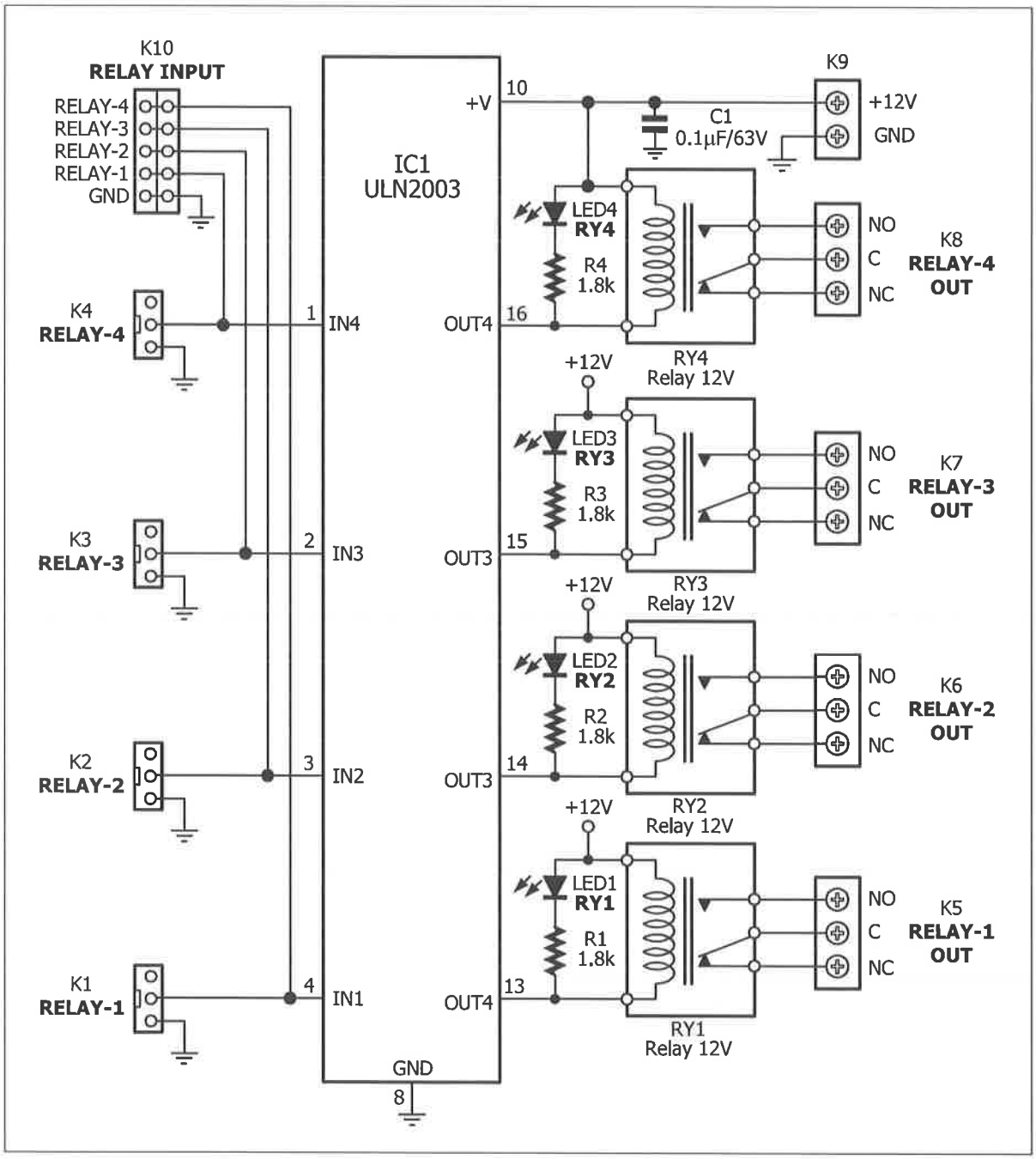
(3) หน้าสัมผัส NC หมายถึง ปกติปิดวงจร (Normally Close) เมื่อรีเลย์ทำงานจะเปิดวงจรออกจากกับขา C หากต้องการใช้งานแบบตัดวงจร ต้องเลือกต่อใช้งานหน้าสัมผัส NC และ C

(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 8-4 คุณสมบัติทางเทคนิคของ RELAY4i บอร์ดขับรีเลย์ 4 ช่อง



รูปที่ 8-5 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 จะได้ยินเสียงหน้าสัมผัสรีเลย์ตัดต่อ พร้อมกับไฟแสดงการทำงานของรีเลย์ติดสว่าง หากต้องการหยุดการทำงานให้ส่งสัญญาณลอจิก "0" เข้ามาที่อินพุตของวงจร

8.4 ตัวอย่างการทดลองใช้งาน Raspberry Pi 2 ขับโหลดกระแสไฟฟ้าสูงด้วยรีเลย์

8.4.1 เตรียมอุปกรณ์

ประกอบด้วย

1. บอร์ด Raspberry Pi 2

2. บอร์ด Rpi-I/O40 ที่ติดตั้งเบรดบอร์ดหรือแผงต่อวงจร พร้อมกับสายแพ 40 เส้นสำหรับต่อกับบอร์ด Raspberry Pi 2

3. บอร์ด RELAY4i

4. อะแดปเตอร์ 12V 1A

5. สายต่อวงจร IDC1MF หรือ IDC1FF 5 เส้น

6. โหลดกำลังไฟฟ้าสูง พร้อมสายต่อไฟสลับ 220Vac

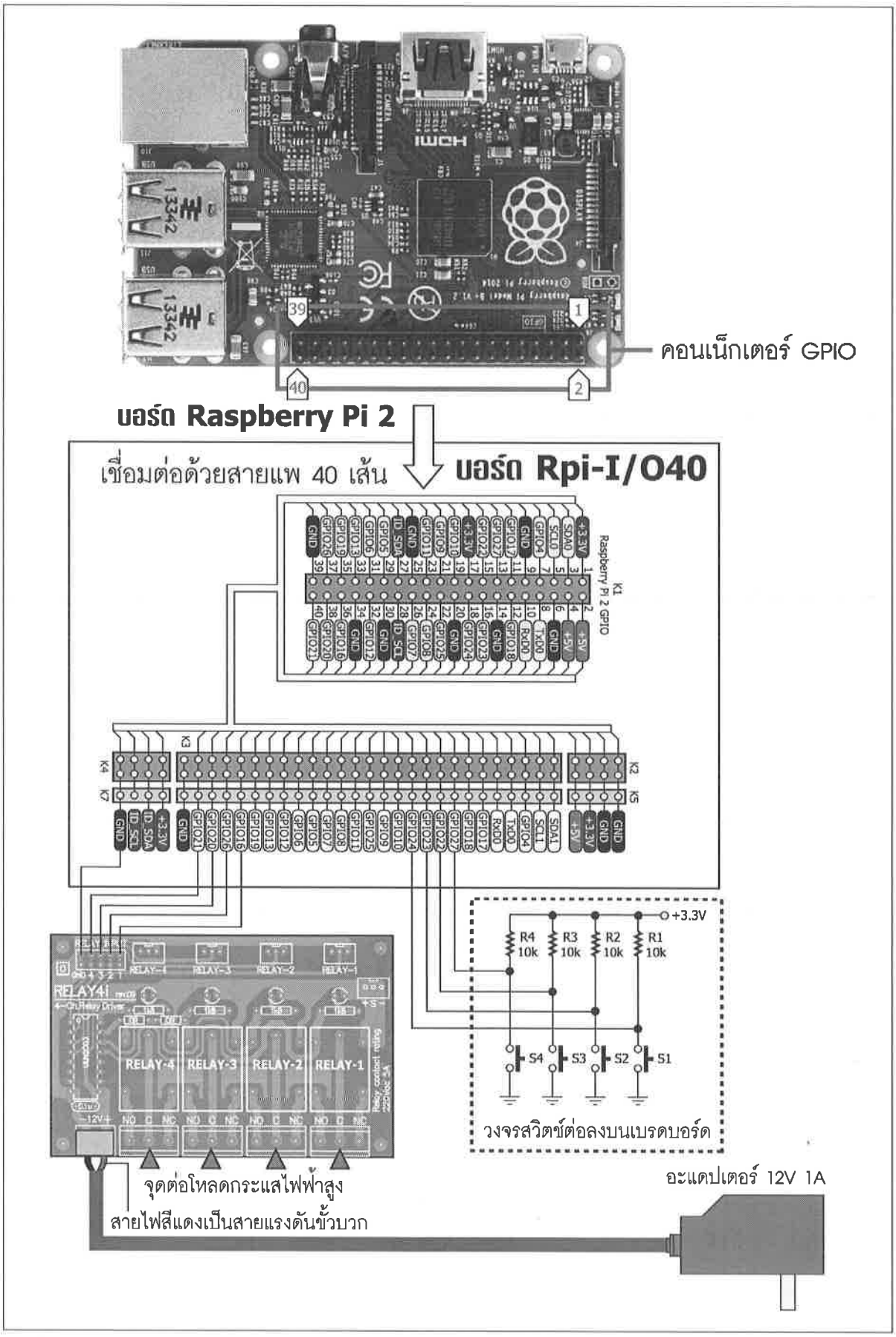
8.4.2 วงจรสวิตช์ควบคุมรีเลย์ด้วย Raspberry Pi 2

มีขั้นตอนดังนี้

(1) เชื่อมต่อบอร์ด Raspberry Pi 2 กับอุปกรณ์ทั้งหมดเพื่อสร้างเป็นระบบควบคุมโหลดกระแสไฟฟ้าสูงอย่างง่าย ดังรูปที่ 8-7

(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) พิมพ์คำสั่งตามโปรแกรมที่ 8-1 แล้วบันทึกชื่อไฟล์เป็น RelaySW2.py



รูปที่ 8-7 วงจรทดลองการเชื่อมต่อ Raspberry Pi 2 กับวงจรขับโหลดกระแสไฟฟ้าสูงด้วยรีเลย์

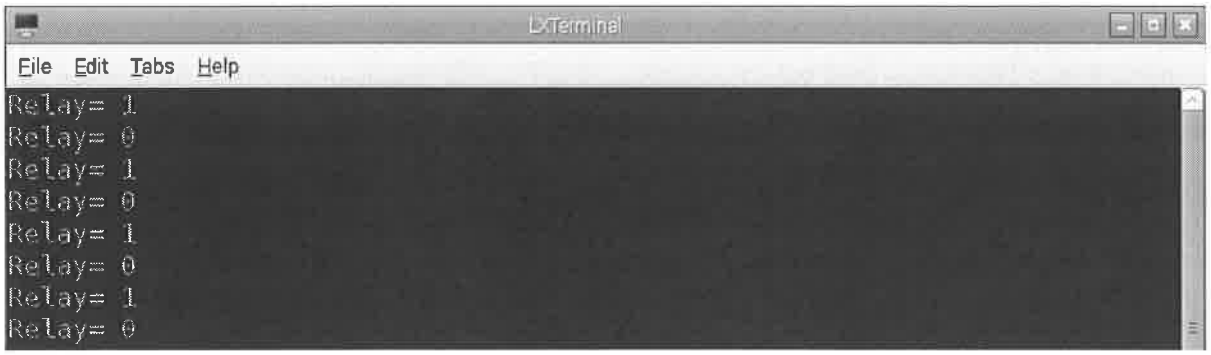
```
import RPi.GPIO as GPIOimport time
GPIO.setmode(GPIO.BCM)
Rel_1=16
sw_on=27
sw_off=22
GPIO.setup(Rel_1,GPIO.OUT)
GPIO.setup(sw_on,GPIO.IN)
GPIO.setup(sw_off,GPIO.IN)
state=0;
while (1):
  if(GPIO.input(sw_on)==0):
     state=1
  if(GPIO.input(sw_off)==0):
     state=0
  GPIO.output(Rel_1,state)
  print("Relay=",state)
```

โปรแกรมที่ 8-1 โค้ด Python สำหรับบอร์ด Raspberry Pi 2 เพื่อใช้งานขาพอร์ต GPIO ในการรับค่าจากสวิตช์ 2 ตัวเพื่อนำมาขับโหลดกระแสไฟฟ้าสูงผ่านวงจรขับรีเลย์ 1 ช่องโดยใช้ RELAY4i บอร์ดขับรีเลย์ 4 ช่อง

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

ในตัวอย่างนี้ จะใช้สวิตช์ 2 ตัวในการควบคุมการทำงานของรีเลย์ 1 ช่อง เมื่อเริ่มต้นการทำงาน รีเลย์ไม่ทำงานจนกว่าจะมีการกดสวิตช์ที่ต่อกับขา GPIO27 เมื่อมีการกดสวิตช์เกิดขึ้น โปรแกรมจะส่งสัญญาณลอจิก "1" ไปยังขาพอร์ต GPIO16 เพื่อกระตุ้นให้วงจรขับรีเลย์ทำงาน สังเกตได้จาก LED ของวงจรขับรีเลย์ที่ทำงานจะติดสว่าง

หากต้องการให้รีเลย์หยุดทำงาน ทำได้ด้วยการกดสวิตช์ที่ต่อกับขา GPIO22 เมื่อมีการกดสวิตช์เกิดขึ้น โปรแกรมจะส่งสัญญาณลอจิก "0" ไปยังขาพอร์ต GPIO16 เพื่อหยุดการทำงานของรีเลย์ทำงาน พร้อมกันนั้น หน้าต่าง LX Terminal จะปรากฏขึ้น เพื่อแสดงสถานะการทำงานของรีเลย์ด้วย โดย Relay=1 หมายถึง รีเลย์ทำงาน และ Relay=0 หมายถึงรีเลย์หยุดทำงาน



8.4.3 วงจรสวิตช์ควบคุมรีเลย์แบบกดเปิดกดปิด

ในตัวอย่างนี้พัฒนาต่อจากหัวข้อ 8.4.2 ซึ่งใช้สวิตช์ 2 ตัวสำหรับเลือกให้รีเลย์ทำงานและหยุดทำงาน โดยในตัวอย่างนี้จะลดสวิตช์เหลือ 1 ตัว เพื่อควบคุมการทำงานของรีเลย์ มีขั้นตอนการทดลองดังนี้

(1) ยังคงใช้วงจรในรูปที่ 8-7 ในการทดลอง โดยเลือกใช้สวิตช์ 1 ตัวที่ต่อกับขาพอร์ต GPIO27 และวงจรขับรีเลย์ 1 ช่องซึ่งต่อกับขาพอร์ต GPIO16

(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) พิมพ์คำสั่งตามโปรแกรมที่ 8-2 แล้วบันทึกชื่อไฟล์เป็น RelayToggleSW.py

```
import RPi.GPIO as GPIOimport time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
sw_onoff = 27
Rel_1=16
GPIO.setup(Rel_1,GPIO.OUT)
GPIO.setup(sw_onoff,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
def sw_pressed(channel):
  GPIO.output(16,not(GPIO.input(16)))
  print("Relay=",GPIO.input(16))
GPIO.add_event_detect(sw_onoff,GPIO.RISING,callback=sw_pressed,bouncetime=300)
while(1):
  pass
```

คำอธิบายโปรแกรมเพิ่มเติม

ในโปรแกรมนี้ หัวใจสำคัญคือ คำสั่ง `GPIO.add_event_detect(sw_onoff,GPIO.RISING, callback=sw_pressed,bouncetime=300)` ซึ่งใช้ในการรักษาค่าสถานะของขาพอร์ตก่อนหน้าไว้ เมื่อมีการเปลี่ยนแปลงที่ขาพอร์ต GPIO27 ซึ่งต่อกับสวิตช์อีกครั้ง ก็จะทำการกลับสถานะของขาพอร์ตเอาต์พุต จาก "1" เป็น "0" ทำให้วงจรขับรีเลย์ที่ต่อกับขาพอร์ต GPIO16 หยุดทำงาน

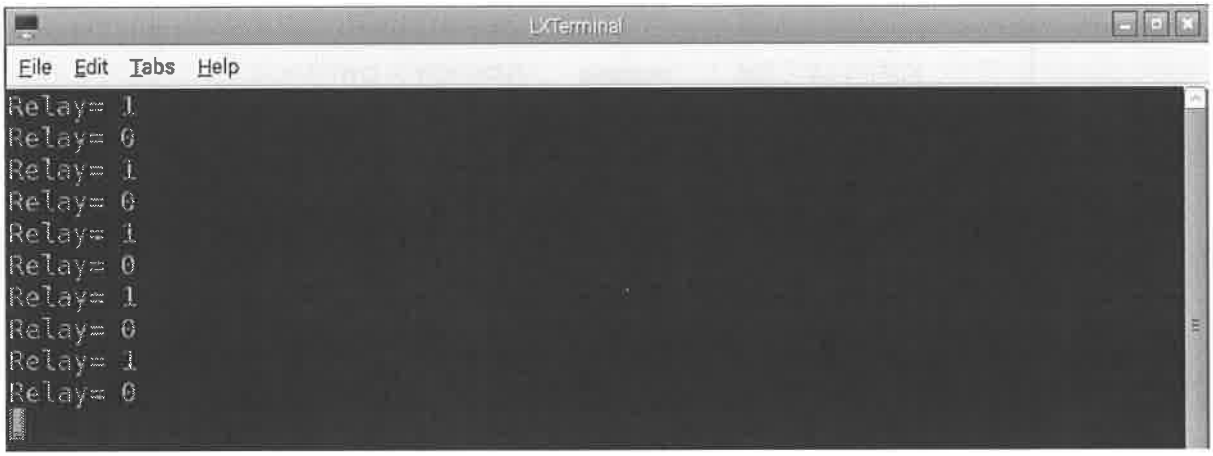
โปรแกรมที่ 8-2 โค้ด Python ไฟล์ RelayToggleSW.py สำหรับบอร์ด Raspberry Pi 2 เพื่อใช้งานขาพอร์ต GPIO 1 ขาในการรับค่าจากการกดสวิตช์ เพื่อนำมาขับโหลดกระแสไฟฟ้าสูงผ่านวงจรขับรีเลย์ 1 ช่องโดยใช้ RELAY4i บอร์ดขับรีเลย์ 4 ช่อง

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

ในตัวอย่างนี้ จะใช้สวิตช์ 1 ตัวในการควบคุมการทำงานของรีเลย์ 1 ช่อง เมื่อเริ่มต้นการทำงาน รีเลย์ไม่ทำงานจนกว่าจะมีการกดสวิตช์ที่ต่อกับขา GPIO27 เมื่อมีการกดสวิตช์เกิดขึ้น โปรแกรมจะส่งสัญญาณลอจิก "1" ไปยังขาพอร์ต GPIO16 เพื่อกระตุ้นให้วงจรขับรีเลย์ทำงาน สังเกตได้จาก LED ของวงจรขับรีเลย์ที่ทำงานจะติดสว่าง

จากนั้นทดลองกดสวิตช์ซ้ำ โปรแกรมจะทำการส่งสัญญาณลอจิก "0" ไปยังขาพอร์ต GPIO16 เพื่อหยุดการทำงานของวงจรขับรีเลย์

นอกจากนั้น ในขณะทำงาน หน้าต่าง LX Terminal จะปรากฏขึ้นเพื่อแสดงสถานะการทำงานของรีเลย์ด้วย จาก **`Relay=1`** *หมายถึง รีเลย์ทำงาน และ* **`Relay=0`** *หมายถึง รีเลย์หยุดทำงาน*



8.4.4 วงจรสวิตช์ควบคุมรีเลย์แบบกดเปิดกดปิด 4 ช่อง

ในตัวอย่างนี้พัฒนาต่อจากหัวข้อ 8.4.3 โดยจะทำการควบคุมรีเลย์ 4 ตัว โดยใช้สวิตช์ 4 ตัวแยกควบคุมกันอย่างอิสระ มีขั้นตอนการทดลองดังนี้

(1) ยังคงใช้วงจรในรูปที่ 8-7 ในการทดลอง โดยในตัวอย่างนี้ใช้สวิตช์ครบ 4 ตัวที่ต่อกับขาพอร์ต GPIO27, 22, 23 และ 24 ร่วมกับการใช้วงจรขับรีเลย์ 4 ช่องซึ่งต่อกับขาพอร์ต GPIO16, 26, 20 และ 21 เรียงตามลำดับ

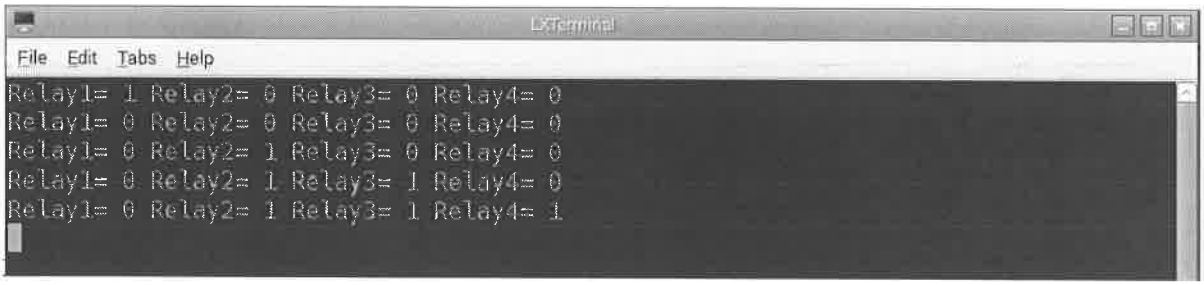
ขาพอร์ตที่ต่อกับสวิตช์	ขาพอร์ตที่เชื่อมต่อกับบอร์ด RELAY4i
GPIO27 : S1	GPIO16 : RELAY-1
GPIO22 : S2	GPIO26 : RELAY-2
GPIO23 : S3	GPIO20 : RELAY-3
GPIO24 : S4	GPIO21 : RELAY-4

(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) พิมพ์คำสั่งตามโปรแกรมที่ 8-3 แล้วบันทึกชื่อไฟล์เป็น RelayToggleSW4.py

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **`sudo python3 RelayToggleSW4.py`**

ในตัวอย่างนี้ จะใช้สวิตช์ 1 ตัวในการควบคุมการทำงานของรีเลย์ 1 ช่อง แต่จะบรรจุครบ 4 ช่องตามความสามารถของบอร์ดขับรีเลย์ โดยสวิตช์ทุกตัวจะทำงานในลักษณะกดหนึ่งครั้งส่งลอจิก "1" เมื่อกดอีกครั้ง จะส่งลอจิก "0" ออกไปแทน นอกจากนั้น ในขณะทำงาน หน้าต่าง LX Terminal จะปรากฏขึ้นเพื่อแสดงสถานะการทำงานของรีเลย์ทั้ง 4 ช่องด้วย



```
import RPi.GPIO as GPIOimport time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

sw_1 = 27
sw_2 = 22
sw_3 = 23
sw_4 = 24
Rel_1=16
Rel_2=26
Rel_3=20
Rel_4=21

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)
GPIO.setup(sw_2,GPIO.IN)
GPIO.setup(sw_3,GPIO.IN)
GPIO.setup(sw_4,GPIO.IN)
state=0

def  sw_pressed(channel):
  if(channel==sw_1):
    GPIO.output(Rel_1,not(GPIO.input(Rel_1)))
  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)))
print("Relay1=",GPIO.input(Rel_1),"Relay2=",GPIO.input(Rel_2),"Relay3=",
GPIO.input(Rel_3),"Relay4=",GPIO.input(Rel_4))

GPIO.add_event_detect(sw_1,GPIO.RISING,callback=sw_pressed,bouncetime=300)
GPIO.add_event_detect(sw_2,GPIO.RISING,callback=sw_pressed,bouncetime=300)
GPIO.add_event_detect(sw_3,GPIO.RISING,callback=sw_pressed,bouncetime=300)
GPIO.add_event_detect(sw_4,GPIO.RISING,callback=sw_pressed,bouncetime=300)

while(1):
  pass
```

คำอธิบายโปรแกรมเพิ่มเติม

ในโปรแกรมนี้ ที่บรรทัด `print("Relay1=",GPIO.input(Rel_1),"Relay2=", GPIO.input(Rel_2),"Relay3=",GPIO.input(Rel_3),"Relay4=",GPIO. input(Rel_4))` เป็นการกำหนดให้ Rapberry Pi 2 ทำการแสดงสถานะของวงจรขับรีเลย์ทั้ง 4 ช่องบนบรรทัดเดียวที่หน้าต่างเทอร์มินอล โดยการพิมพ์คำสั่งต้องพิมพ์ต่อเนื่อง ไม่ต้องขึ้นบรรทัดใหม่ (ในหน้านี้ต้องขึ้นบรรทัดใหม่เนื่องจากข้อจำกัดด้านพื้นที่ของหน้ากระดาษ)

โปรแกรมที่ 8-3 โค้ด Python ไฟล์ RelayToggleSW4.py สำหรับบอร์ด Raspberry Pi 2 เพื่อใช้งานขาพอร์ต GPIO 4 ขาในการรับค่าจากการกดสวิตช์ เพื่อนำมาขับโหลดกระแสไฟฟ้าสูง 4 ช่องโดยใช้บอร์ด RELAY4i

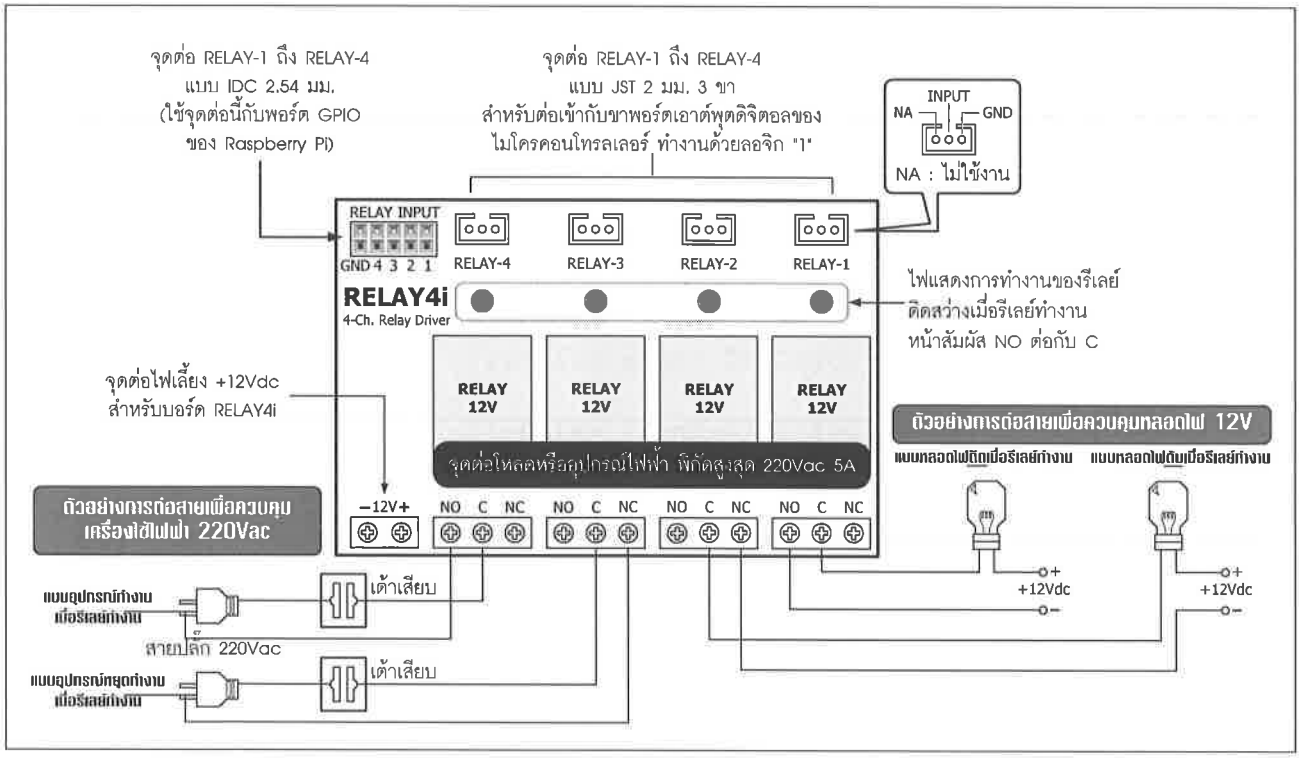
8.5 เกี่ยวกับการต่อใช้งานหน้าสัมผัสของรีเลย์แบบกลไก

เนื่องจากรีเลย์แบบกลไกมีหน้าสัมผัสสำหรับต่อกับโหลดและแรงดันไฟเลี้ยงของโหลดในแบบปกติปิดหรือ NC และแบบปกติเปิดหรือ NO ดังนั้นการใช้งานจึงมี 2 แนวทางหลักๆ คือ

1. กำหนดให้โหลดหรือเครื่องใช้ไฟฟ้าทำงานเมื่อรีเลย์ทำงาน ในแนวทางนี้จะเลือกต่อใช้งานหน้าสัมผัส NO และ C

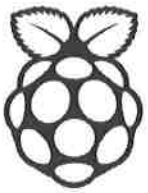
2. กำหนดให้โหลดหรือเครื่องใช้ไฟฟ้าหยุดทำงานเมื่อรีเลย์ทำงาน ในแนวทางนี้จะเลือกต่อใช้งานหน้าสัมผัส NC และ C แทน

รูปที่ 8-8 แสดงแนวทางการใช้งานบอร์ดขับรีเลย์เพื่อขับโหลดในลักษณะต่างๆ



รูปที่ 8-8 แนวทางการใช้งานบอร์ดขับรีเลย์ RELAY4i เพื่อขับโหลดในแบบทำงานเมื่อรีเลย์ทำงานและแบบหยุดทำงานเมื่อรีเลย์ทำงาน โดยแสดงการใช้งานกับโหลดที่ใช้ไฟฟ้ากระแสตรงอย่างหลอดไฟและเครื่องใช้ไฟฟ้า 220Vac ผ่านเต้าเสียบ

บทที่ 9
เชื่อมต่อไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลผ่านบัส SPI



การเพิ่มเติมความสามารถสำหรับบอร์ดคอมพิวเตอร์อย่าง Raspberry Pi 2 ด้วยอุปกรณ์ภายนอกสมัยใหม่เป็นความจำเป็น เนื่องจากความสามารถพิเศษบางอย่างจะมีการทำงานที่แน่นอนและช่วยลดภาระในการทำงานแก่ซีพียูลงได้อย่างมาก โดยเฉพาะอย่างยิ่ง ความสามารถในการการแปลงสัญญาณอะนาลอกเป็นดิจิตอลที่มีความละเอียดสูง และมีจำนวนช่องอินพุตมาก เป็นสิ่งบรรจุรวมลงไปในบอร์ดคอมพิวเตอร์ได้ไม่ง่าย อาจทำให้ราคาโดยรวมของฮาร์ดแวร์ทั้งระบบสูงเกินไป ด้วยการใช้ไอซีที่ทำหน้าที่นี้โดยเฉพาะจะช่วยให้เสถียรภาพการทำงาน, ต้นทุนรวมของระบบ และการพัฒนาโปรแกรมในภาพรวมเป็นไปอย่างมีประสิทธิภาพมากขึ้น

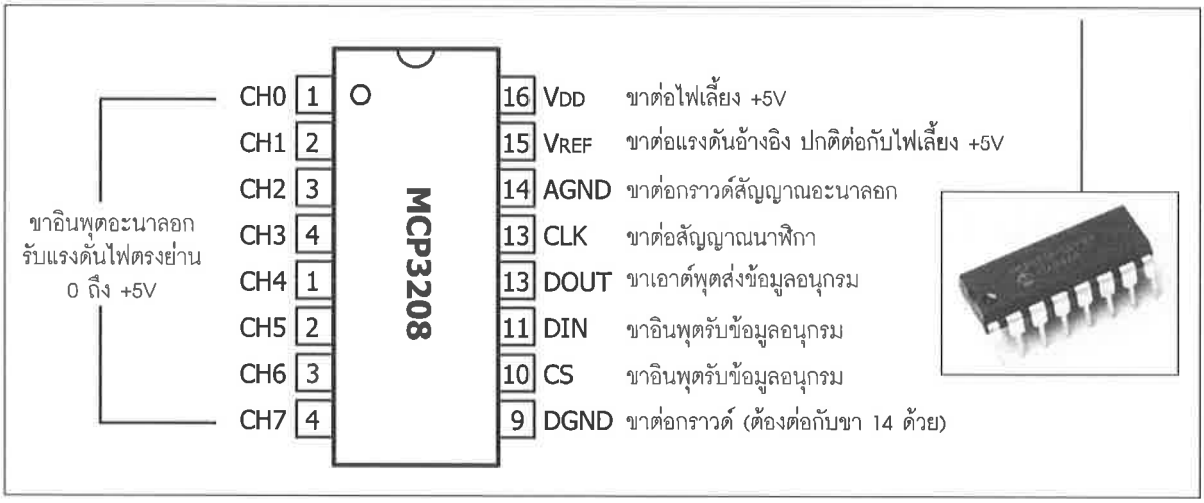
แม้ว่าบอร์ด Raspberry Pi 2 จะไม่ได้รวมโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลเอาไว้ ทว่านั่นไม่ใช่ปัญหาเพราะในปัจจุบันมีไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลความละเอียดสูงที่ติดต่อกับไมโครคอนโทรลเลอร์หรือบอร์ดคอมพิวเตอร์ด้วยการใช้สายสัญญาณไม่กี่เส้นให้เลือกใช้มากมาย ที่นำมาแนะนำในที่นี้คือ **MCP3208** อันเป็นไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลหรือ ADC ที่มีความละเอียดในการแปลงสัญญาณสูงถึง 12 บิต นั่นคือให้ข้อมูลดิจิตอลในช่วง 0 ถึง 4,095 รวม 4,096 ค่า สำหรับแรงดันไฟตรงในย่าน 0 ถึง +5V และใช้งานได้กับไฟเลี้ยง +3.3V ด้วย เท่ากับว่ามันสามารถวัดระดับการเปลี่ยนแปลงของสัญญาณไฟฟ้าได้ละเอียดถึง 0.001220 V (ที่ +5V) เลยทีเดียว ส่วนด้านการติดต่อเป็นแบบอนุกรม 4 สายที่เรียกว่า ***ระบบบัส SPI*** หรือ ***Serial Peripheral Interface***

การเชื่อมต่อผ่านระบบบัส SPI เป็นการสื่อสารข้อมูลอนุกรมแบบซิงโครนัส (Synchronous) ซึ่งมีสัญญาณนาฬิกาเป็นตัวกำหนดจังหวะในการถ่ายทอดข้อมูลระหว่างอุปกรณ์

9.1 MCP3208 ไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลความละเอียดสูง

9.1.1 ข้อมูลทางเทคนิคเบื้องต้น

MCP3208 เป็นแปลงสัญญาณอะนาลอกเป็นดิจิตอลที่มีความละเอียดในการแปลงสัญญาณ 12 บิต มีอินพุตอะนาลอก 8 ช่อง รับแรงดันได้ 0 ถึง +5V ติดต่อกับไมโครคอนโทรลเลอร์หรือบอร์ดคอมพิวเตอร์ผ่านระบบบัส SPI โดยใช้ขาสัญญาณ 3 หรือ 4 ขา ในรูปที่ 9-1 แสดงการจัดขาของไอซี MCP3208



รูปที่ 9-1 การจัดขาของไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลเบอร์ MCP3208

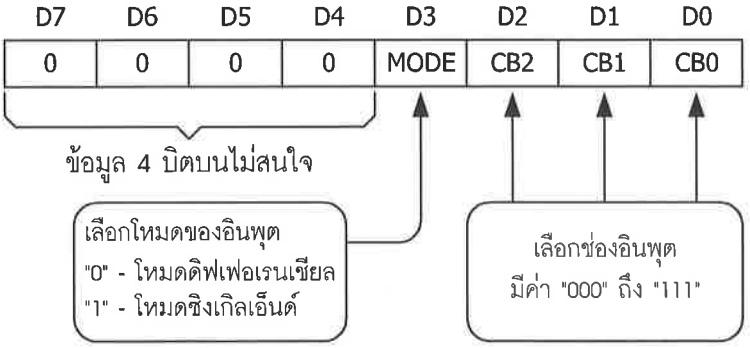
บิตควบคุม				โหมดของอินพุต	ช่องอินพุตที่เลือก	
D3 Single/$\overline{\text{Diff}}$	**D2**	**D1**	**D0**			
1	0	0	0	ซิงเกิลเอ็นด์	CH0	
1	0	0	1	ซิงเกิลเอ็นด์	CH1	
1	0	1	0	ซิงเกิลเอ็นด์	CH2	
1	0	1	1	ซิงเกิลเอ็นด์	CH3	
1	1	0	0	ซิงเกิลเอ็นด์	CH4	
1	1	0	1	ซิงเกิลเอ็นด์	CH5	
1	1	1	0	ซิงเกิลเอ็นด์	CH6	
1	1	1	1	ซิงเกิลเอ็นด์	CH7	
0	0	0	0	ดิฟเฟอเรนเชียล	CH0=IN+	CH1=IN-
0	0	0	1	ดิฟเฟอเรนเชียล	CH0=IN-	CH1=IN+
0	0	1	0	ดิฟเฟอเรนเชียล	CH2=IN+	CH3=IN-
0	0	1	1	ดิฟเฟอเรนเชียล	CH2=IN-	CH3=IN+
0	1	0	0	ดิฟเฟอเรนเชียล	CH4=IN+	CH5=IN-
0	1	0	1	ดิฟเฟอเรนเชียล	CH4=IN-	CH5=IN+
0	1	1	0	ดิฟเฟอเรนเชียล	CH6=IN+	CH7=IN-
0	1	1	1	ดิฟเฟอเรนเชียล	CH6=IN-	CH7=IN+

ตารางที่ 9-1 ข้อมูลสำหรับเลือกโหมดการทำงานและช่องอินพุตอะนาลอกที่ต้องการอ่านค่าของไอซี MCP3208

ไอซี MCP3208 มีอินพุตรับสัญญาณอะนาลอกโดยตรง 8 ช่องในแบบเทียบกับกราวด์หรือเรียกว่า ช่องอินพุตเดี่ยว หรือ ซิงเกิลเอ็นด์ (single-ended) หากมีความต้องการให้ใช้งานเป็นอินพุตวัดแตกต่างหรือดิฟเฟอเรนเชียล (differential) ก็ทำได้ จะทำให้เหลืออินพุตแบบวัดความแตกต่าง 4 ชุด ในการกำหนดลักษณะการทำงานของอินพุตนั้นจะต้องเขียนข้อมูล 4 บิตเพื่อตั้งค่าการทำงานและเลือกช่องอินพุตที่ต้องการอ่านค่าไปยัง MCP3208 ดังมีข้อมูลโดยสรุปแสดงดังตารางที่ 9-1

9.1.2 การอ่านข้อมูลจาก MCP3208

(1) การอ่านข้อมูลจากแต่ละช่องอินพุตนั้น จะต้องส่งข้อมูลควบคุมจำนวน 4 บิตไปยัง MCP3208 ก่อน โดยมีรูปแบบดังนี้



(2) บิตที่สนใจคือ 4 บิตล่าง ประกอบด้วย **บิตเลือกโหมด MODE** และ**บิตเลือกช่องสัญญาณอีก 3 บิต** เมื่อไมโครคอนโทรลเลอร์ส่งข้อมูลตั้งค่าออกไปแบบ 8 บิต จะพบว่า ข้อมูล 4 บิตแรกนั้น MCP3208 จะไม่สนใจ และจะสนใจเริ่มตอบสนองต่อข้อมูล 4 บิตล่างเท่านั้น

ตัวอย่าง : ข้อมูลควบคุมต่อไปนี้ เป็นการเลือกให้อินพุตช่อง 0 ของ MCP3208 ทำงานในแบบซิงเกิลเอ็นด์

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	0	0

ดังนั้นการอ่านค่าจากอินพุตของ MCP3208 จึงสามารถส่งข้อมูลควบคุมได้หลายค่าเพื่อเลือกอินพุตช่องเดียวกันดังตัวอย่างบางส่วนในตารางที่ 9-2 และ 9-3

	ข้อมูลควบคุม #1		ข้อมูลควบคุม #2	
ช่องอินพุต	เลขฐานสอง	เลขฐานสิบ	เลขฐานสอง	เลขฐานสิบ
CH0	0001 1000	24	0000 1000	8
CH1	0001 1001	25	0000 1001	9
CH2	0001 1010	26	0000 1010	10
CH3	0001 1011	27	0000 1011	11
CH4	0001 1100	28	0000 1100	12
CH5	0001 1101	29	0000 1101	13
CH6	0001 1110	30	0000 1110	14
CH7	0001 1111	31	0000 1111	15

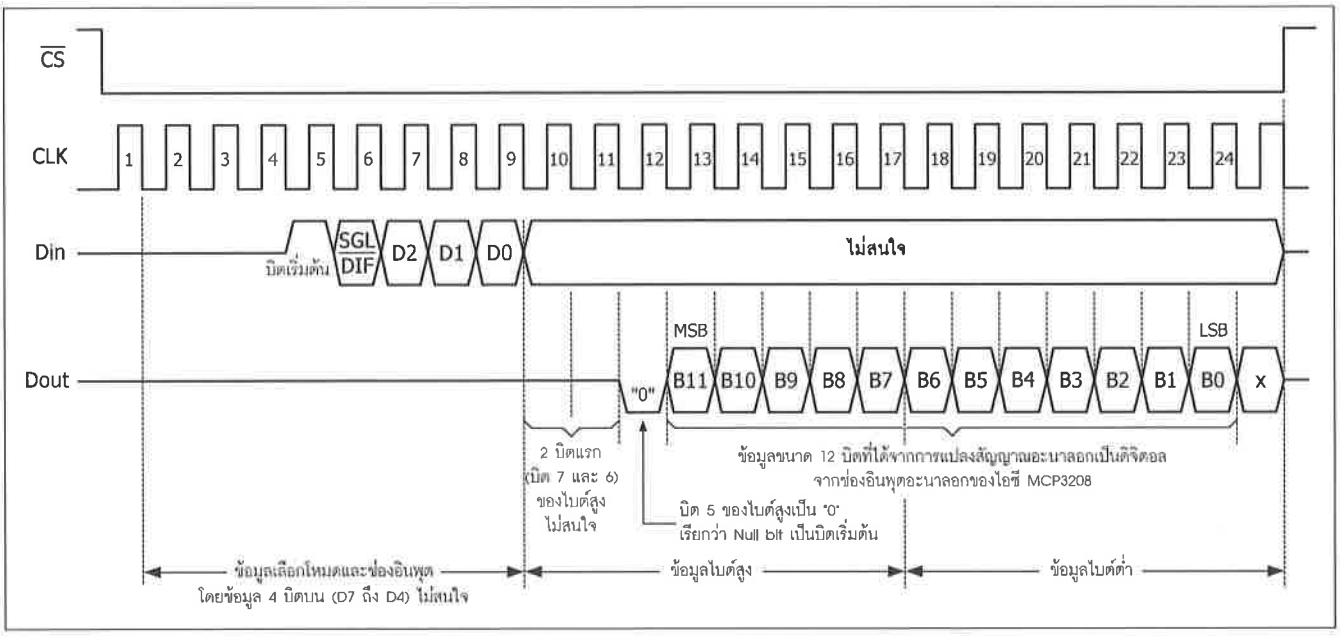
หากต้องการอ่านค่าจากอินพุตช่อง 0 ในโหมดซิงเกิลเอ็นด์
ตัวควบคุม/ซีพียูส่งข้อมูลควบคุมเป็น 0x08 (00001000 ฐานสองหรือ 8 ฐานสิบ) หรือ 0x18, 0x28, 0x38, 0x48, 0x58, 0x68, 0x78, 0x88, 0x98, 0xA8, 0xB8, 0xC8, 0xD8, 0xE8 และ 0xF8 (ฐานสิบหก) เพราะ 4 บิตบนเป็นค่าใดๆ ก็ได้

ตารางที่ 9-2 ตัวอย่างข้อมูลสำหรับอ่านค่าจาก MCP3208 ในโหมดซิงเกิลเอ็นด์

	ข้อมูลควบคุม #1		ข้อมูลควบคุม #2	
ช่องอินพุต	เลขฐานสอง	เลขฐานสิบ	เลขฐานสอง	เลขฐานสิบ
CH0	0001 0000	16	0000 0000	0
CH1	0001 0001	17	0000 0001	1
CH2	0001 0010	18	0000 0010	2
CH3	0001 0011	19	0000 0011	3
CH4	0001 0100	20	0000 0100	4
CH5	0001 0101	21	0000 0101	5
CH6	0001 0110	22	0000 0110	6
CH7	0001 0111	23	0000 0111	7

หากต้องการอ่านค่าจากอินพุตช่อง 0 ในโหมดดิฟเฟอเรนเชียล
ตัวควบคุม/ซีพียูส่งข้อมูลควบคุมเป็น 0x00 (00000000 ฐานสองหรือ 0 ฐานสิบ) หรือ 0x10, 0x20, 0x30, 0x40, 0x50, 0x60, 0x70, 0x80, 0x90, 0xA0, 0xB0, 0xC0, 0xD0, 0xE0 และ 0xF0 (ฐานสิบหก) เพราะ 4 บิตบนเป็นค่าใดๆ ก็ได้

ตารางที่ 9-3 ตัวอย่างข้อมูลสำหรับอ่านค่าจาก MCP3208 ในโหมดดิฟเฟอเรนเชียล



รูปที่ 9-2 ไดอะแกรมเวลาของการทำงานของซี MCP3208

(3) เมื่อส่งคำสั่งไปอ่านข้อมูลจาก MCP3208 แล้ว ลำดับต่อไป ซีพียูจะต้องรอรับข้อมูลจาก MCP3208 ขอให้พิจารณาไดอะแกรมเวลาในรูปที่ 9-2 ประกอบด้วย จำนวนไบต์ข้อมูลที่ต้องการรอรับในที่นี้คือ 2 ไบต์

(4) จากไดอะแกรมเวลาของการอ่านข้อมูลจาก MCP3208 ในรูปที่ 9-2 พบว่า หลังจากส่งข้อูลเพื่อเลือกโหมดและช่องอินพุตอะนาลอกให้แก่ไอซี MCP3208 แล้ว ไอซีจะใช้เวลาในการแปลงสัญญาณ ทำให้ข้อมูล 3 บิตแรกของข้อมูลไบต์สูงที่อ่านกลับมาจากขา Dout จึงไม่ต้องสนใจ จนกระทั่งถึงบิตถัดมา (บิต 4 ของไบต์สูง) จะมีค่าเป็น "0" เพื่อแจ้งว่า จะเริ่มต้นส่งข้อมูลที่แปลงเสร็จแล้วออกมา จากนั้นข้อมูลที่ถูกต้องจะเริ่มเกิดขึ้นที่บิต 4 ของไบต์สูง ซึ่งมันคือบิต 11 หรือบิต MSB ของข้อมูลที่ได้จากการแปลงสัญญาณอะนาลอกเป็นดิจิตอลด้วยไอซี MCP3208 รับต่อเนื่องกันไปจนครบ 12 บิต โดยมีรูปแบบข้อมูลดังนี้

ตัวแปร 1								ตัวแปร 2							
D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	0	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0

ข้อมูลที่สนใจและต้องการใช้งานมีเพียง 12 บิตนี้

(5) ข้อมูลที่ต้องการ 12 บิต ประกอบด้วย บิต D0 ถึง บิต D3 ของไบต์สูงและบิต D1 ถึง D7 ของไบต์ต่ำ เพื่อให้นำไปประมวลผลต่อได้ จึงต้องนำข้อมูลทั้งหมดมาเรียงต่อกันเป็นข้อมูลขนาด 16 บิต

จะได้ข้อมูลที่ถูกต้องนำมาแปลงค่าเป็นแรงดันในหน่วยโวลต์ (V) จากความสัมพันธ์ทางคณิตศาสตร์ต่อไปนี้

```
Vout = Data x (VREF/4096)
   โดยที่ VREF = 5V

Vout = Data x (5/4096)
Vout = Data x 0.001220703125
```

9.2 การติดตั้ง spidev เพื่อใช้งาน SPI บน Raspberry Pi

ในการติดต่ออุปกรณ์บัส SPI ผ่าน GPIO ของบอร์ด Raspberry Pi 2 จำเป็นต้องติดตั้งไดรเวอร์เพื่อให้เชื่อมต่อกับบัส SPI ได้ ไดรเวอร์ที่ต้องการนั้นคือ **spidev**

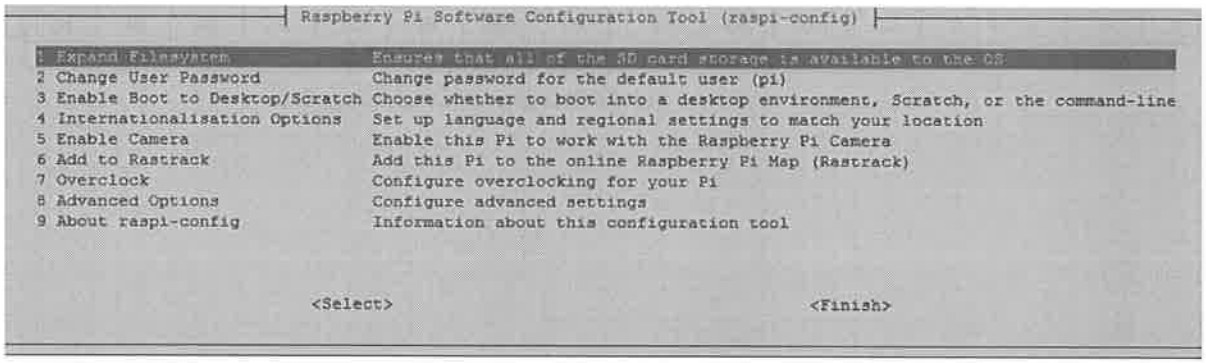
spidev เป็น Linux Kernel Driver สำหรับบอร์ด Raspberry Pi เพื่อให้ติดต่อสื่อสารแบบบัส SPI ด้วยภาษา Python ได้เดิมทีการเขียนโปรแกรมเพื่อเชื่อมต่อบัส SPI โดยตรง จะต้องเขียนโปรแกรมที่ค่อนข้างยุ่งยาก ดังนั้นการใช้ spidev เข้ามาช่วย จะทำให้สะดวกยิ่งขึ้น

สำหรับการติดตั้ง spidev บน Raspberry Pi มีขั้นตอนดังนี้

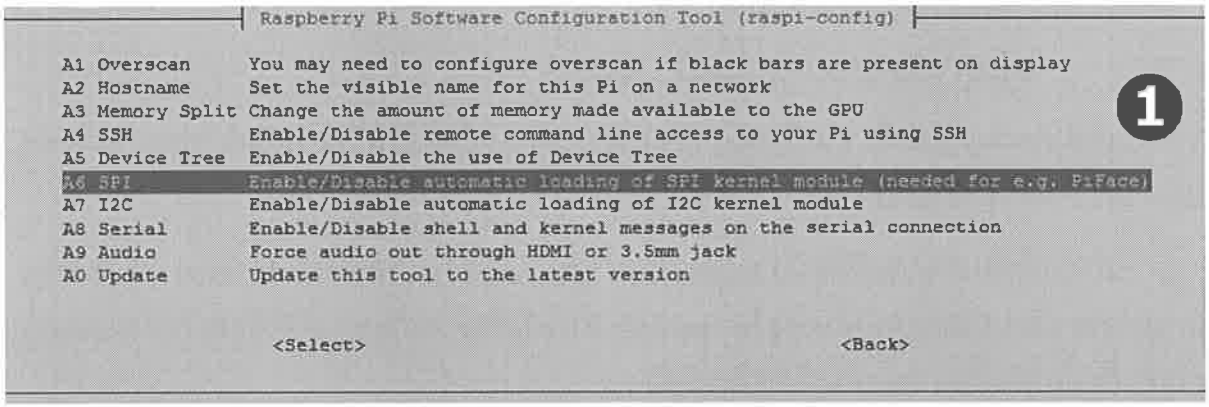
(1) เปิดความสามารถในการติดต่อบัส SPI โดยเปิดหน้าต่างตั้งค่าการทำงานของ Raspberry Pi ด้วยการพิมพ์คำสั่ง

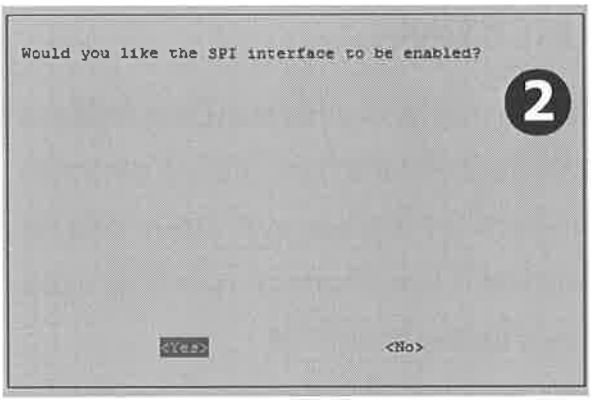
```
sudo raspi-config
```

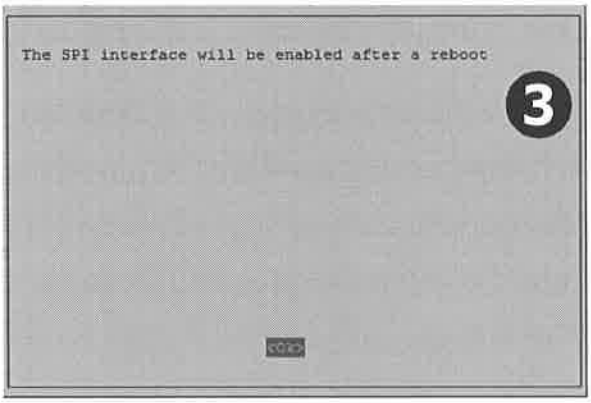
จะปรากฏหน้าต่าง **Raspberry Pi Software Configuration Tools** เลือกข้อ 8 **Advance Option**

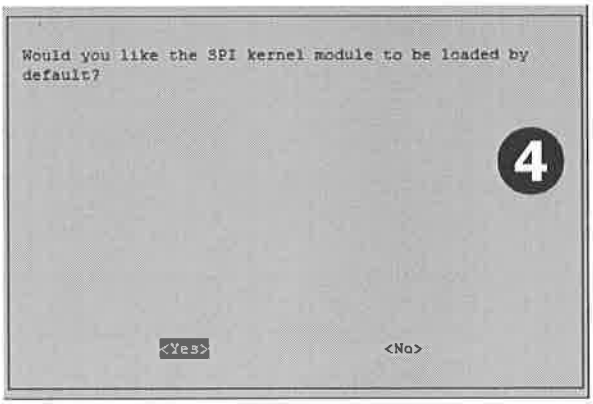


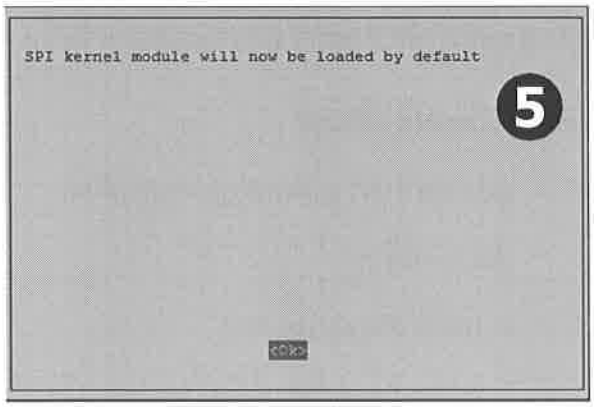
จะเข้าสู่หน้าต่างตั้งค่าการทำงานขั้นสูง เลือกหัวข้อ A6 เพื่อเปิดความสามารถการเชื่อมต่อกับอุปกรณ์บัส SPI โดยกดปุ่ม **Select** และ **Yes** เพื่อตอบรับ แล้วทำการรีบูต











(2) ติดตั้ง **python3-pip** และ **python3-dev** ด้วยคำสั่ง

```
sudo apt-get install python3-pip python3-dev
```

(3) ทำการติดตั้ง **spidev** ด้วยคำสั่ง

```
sudo pip-3.2 install spidev
```

(4) ทำการรีบูตอีกครั้ง จากนี้บอร์ด Rasapberry Pi 2 พร้อมสำหรับการเขียนโปรแกรมติดต่อกับบัส SPI ด้วยภาษา Python

อย่างไรก็ตาม สำหรับผู้ที่จัดซื้อ Raspberry Pi 2 Education kit โปรแกรมและไลบรารีที่เกี่ยวข้องกับการทดลองในหนังสือเล่มนี้ที่บรรจุใน microSD การ์ดได้รับการปรับปรุงให้ใช้งานได้ทั้งหมดแล้ว จึงไม่จำเป็นต้องดำเนินการตามที่อธิบายในหัวข้อ 9.2

9.3 การอ่านสัญญาณอะนาลอกด้วยไอซี MCP3208

ถึงแม้ว่าบอร์ด Raspberry Pi จะมีขา GPIO ให้ใช้งานมากถึง 26 ขา แต่ทั้งหมดเป็นพอร์ตดิจิตอล การรับสัญญาณอะนาลอกจึงต้องใช้อุปกรณ์ช่วย ในที่นี้คือ ไอซี MCP3208 โดยไอซี MCP3208 จะรับสัญญาณอะนาลอกแล้วแปลงเป็นข้อมูลดิจิตอลส่งกลับมายังบอร์ด Raspberry Pi 2 ผ่านทางบัส SPI ซึ่งขาพอร์ต GPIO ของ Raspberry Pi 2 มีความสามารถนี้รองรับแล้ว การต่อวงจรระหว่างไอซี MCP3208 กับบอร์ด Raspberry Pi 2 แสดงไว้ในรูปที่ 9-3 ซึ่งใช้ในการทดสอบในบทนี้ด้วย

ในการเขียนโปรแกรมเพื่อติดต่อกับ MCP3208 ผ่าน SPI จะใช้ spidev ที่เป็น Linux Kernel Driver เข้ามาช่วยเพื่อให้การเขียนโปรแกรมสะดวกยิ่งขึ้น

9.3.1 เตรียมอุปกรณ์

อุปกรณ์สำหรับทดลองประกอบด้วย

ประกอบด้วย

1. บอร์ด Raspberry Pi 2

2. บอร์ด Rpi-I/O40 ที่ติดตั้งเบรดบอร์ดหรือแผงต่อวงจร พร้อมกับสายแพ 40 เส้นสำหรับต่อกับบอร์ด Raspberry Pi 2

3. ไอซี MCP3208

4. ตัวเก็บประจุ 0.1μF 50V หรือ 63V โพลีเอสเตอร์หรือเซรามิก

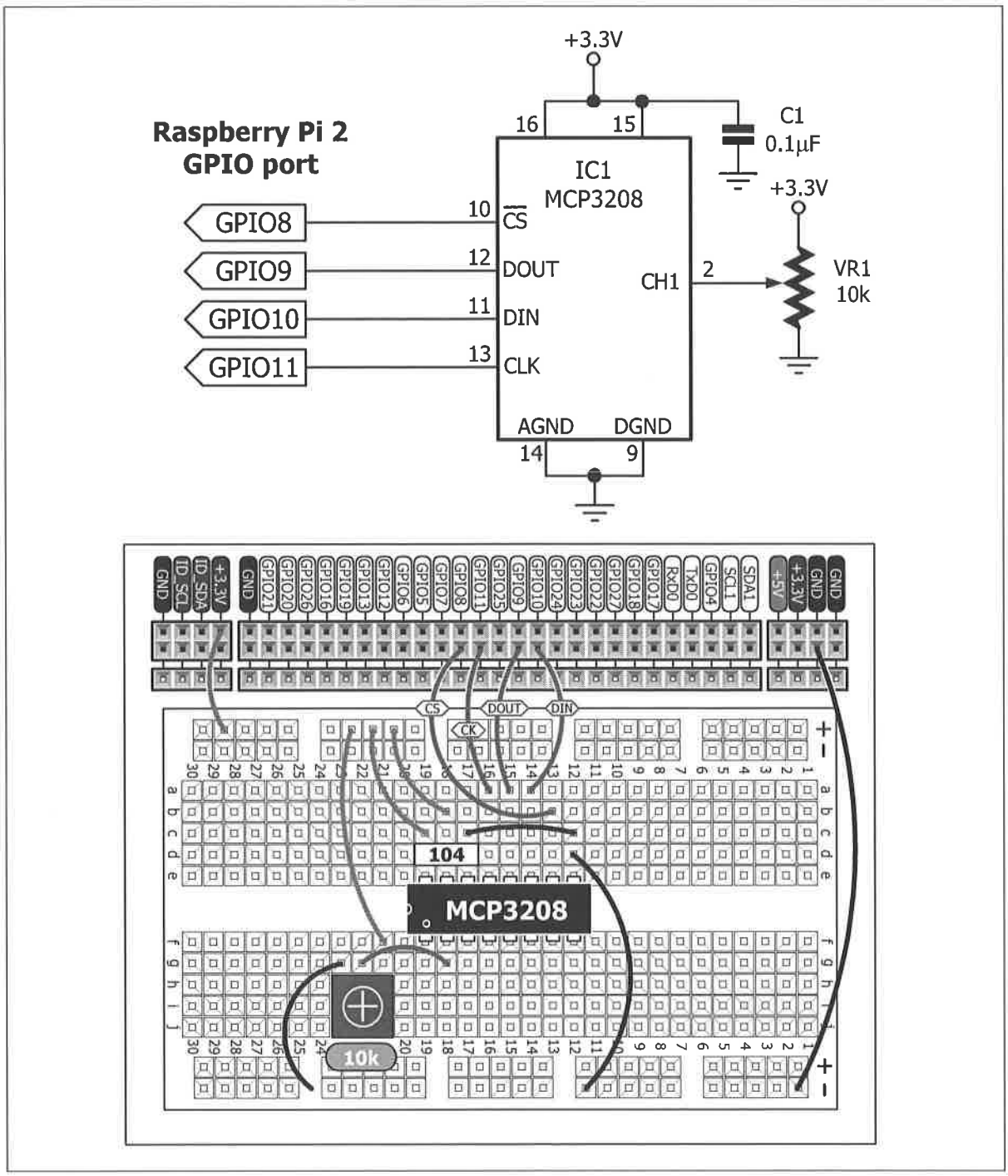
5. ตัวต้านทานปรับค่าได้ 10kΩ แบบมีปุ่มปรับ

6. สายต่อวงจร

9.3.2 ขั้นตอนการทดลองติดต่อ MCP3208 ของบอร์ด Raspberry Pi 2

มีขั้นตอนดังนี้

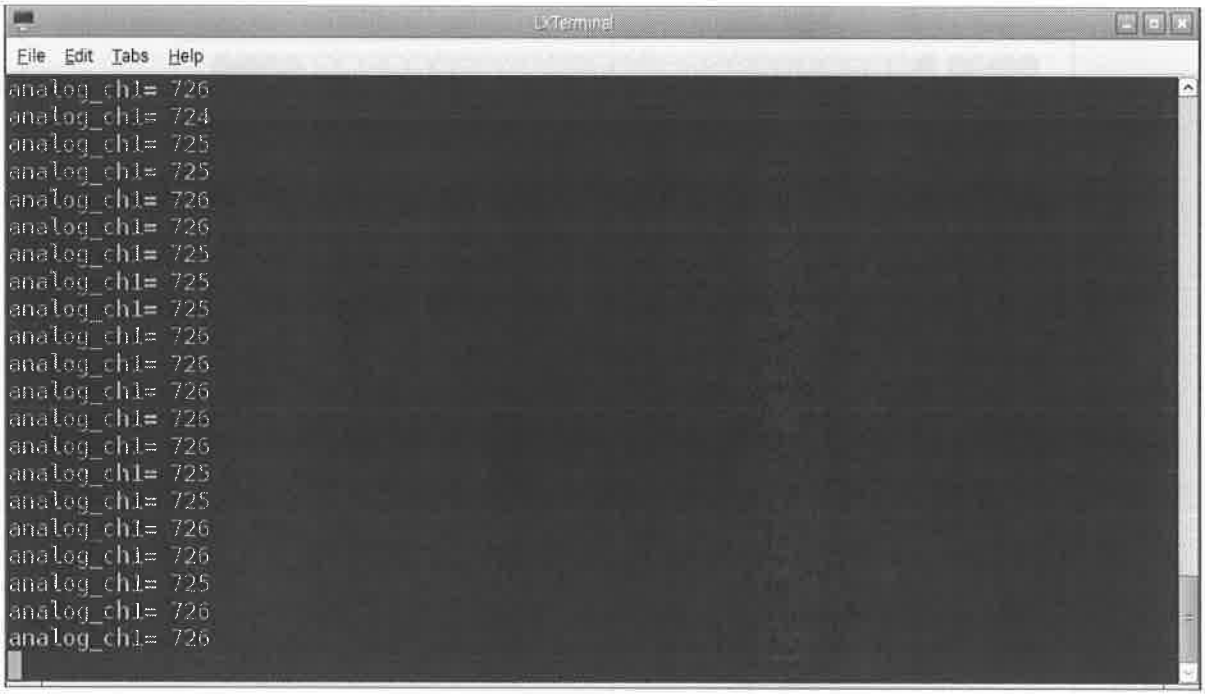
(1) ใช้วงจรจากรูปที่ 9-3 ในการทดลอง โดยต่อวงจรลงบนรเบรดบอร์ดหรือแผงต่อวงจร



รูปที่ 9-3 วงจรและตัวอย่างการเชื่อมต่อบอร์ด Raspberry Pi 2 กับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลที่ใช้ไอซีเบอร์ MCP3208 ผ่านทางขาพอร์ต GPIO

```
import spidev
import time
analog_ch = 1
spi = spidev.SpiDev()
spi.open(0,0)
def readADC(adcnum):
  if adcnum > 7 or adcnum < 0:
     return -1;
  r = spi.xfer2([4 | 2 | (adcnum >> 2), (adcnum & 3) << 6, 0])
  adcout = ((r[1] & 15) << 8) + r[2]
  return adcout
while True:
  value = readADC(analog_ch)
  print("analog_ch1=",value)
  time.sleep(0.3)
```

โปรแกรมที่ 9-1 โค้ด Python ไฟล์ MCP3208ADC1.py สำหรับบอร์ด Raspberry Pi 2 เพื่อติดต่อกับไอซีเบอร์ MCP3208 อ่านค่าแรงดันไฟตรงซึ่งเป็นสัญญาณอะนาลอกมาแสดงเป็นข้อมูลดิจิตอลบนหน้าต่างเทอร์มินอล



รูปที่9-4 ผลการทำงานของโปรแกรมที่ 9-1 บอร์ด Raspberry Pi 2 ติดต่อกับไอซีเบอร์ MCP3208 เพื่อนำค่าแรงดันไฟตรงที่ได้จากการแปลงสัญญาณอะนาลอกเป็นข้อมูลดิจิตอลมาแสดงบนหน้าต่างเทอร์มินอล

(2) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) เขียนคำสั่งตามโปรแกรมที่ 9-1 บันทึกไฟล์เป็น MCP3208ADC1.py

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **`sudo python3 MCP3208ADC1.py`**

(5) ทดสอบการทำงานด้วยการหมุนแกนของตัวต้านทานปรับค่าได้แล้วดูผลการทำงานที่เกิดขึ้นผ่านทางหน้าต่างเทอร์มินอล

โปรแกรมนี้เป็นการอ่านค่าแรงดันไฟตรงอันเป็นสัญญาณอะนาลอกจากขาอินพุต CH1 ของ MCP3208 เพื่อนำค่ามาแสดงที่หน้าจอเทอร์มินอลของ Raspberry Pi 2 เมื่อหมุนตัวต้านทานปรับค่าได้ที่ต่อเข้าที่ขา CH1 ค่าที่ได้จะเปลี่ยนแปลงระหว่าง 0 ถึง 4095 ดังแสดงในรูปที่ 9-4

จากโปรแกรมที่ 9-1 อาจต่อยอดโปรแกรมให้คำนวณค่าข้อมูลดิจิตอลที่แปลงได้เป็นค่าแรงดันไฟฟ้าในหน่วยโวลต์ (V : Volt) และเพิ่มเติมการรับอินพุตให้แก่ไอซี MCP3208 เพื่อให้รับและแปลงสัญญาณได้มากกว่าหนึ่งช่อง

ด้านสัญญาณอินพุตที่ป้อนให้แก่ไอซี MCP3208 อาจเปลี่ยนจากตัวต้านทานปรับค่าได้เป็นตัวตรวจจับปริมาณทางกายภาพอื่นๆ เช่น ตัววัดความชื้น, วัดแสง, วัดสนามแม่เหล็ก ที่ให้ผลการทำงานเป็นแรงดันไฟตรง ผนวกกับความสามารถของ Raspberry Pi 2 จะช่วยนำไปสู่การสร้างเครื่องบันทึกข้อมูลหรือดาต้าล็อกเกอร์ ที่อาจต่อยอดไปยังการส่งผ่านข้อมูลผ่านเครือข่ายอินเทอร์เน็ตเพื่อสร้างระบบบันทึกข้อมูลที่ใหญ่ขึ้นได้อีกด้วย

9.4 ตัวอย่างการติดต่อกับตัวตรวจจับที่ให้ผลการทำงานเป็นแรงดันของ Raspberry Pi 2 ผ่านทางไอซี MCP3208

ในหัวข้อนี้จะเป็นการนำเสนอตัวอย่างการใช้งานจริงของบอร์ด Raspberry Pi 2 ในการติดต่อกับตัวตรวจจับอุณหภูมิเบอร์ MCP9701 ซึ่งให้ผลการทำงานเป็นแรงดันไฟตรงที่แปรผันตรงกับค่าอุณหภูมิ โดยการติดต่อจะใช้ไอซี MCP3208 เป็นตัวช่วย โดยไอซี MCP3208 จะทำการรับสัญญาณอะนาลอกหรือแรงดันเอาต์พุตจากไอซี MCP9701 แล้วแปลงเป็นข้อมูลดิจิตอลส่งกลับมายังบอร์ด Raspberry Pi 2 ผ่านทางบัส SPI เพื่อนำมาแสดงผลที่หน้าต่างเทอร์มินอล

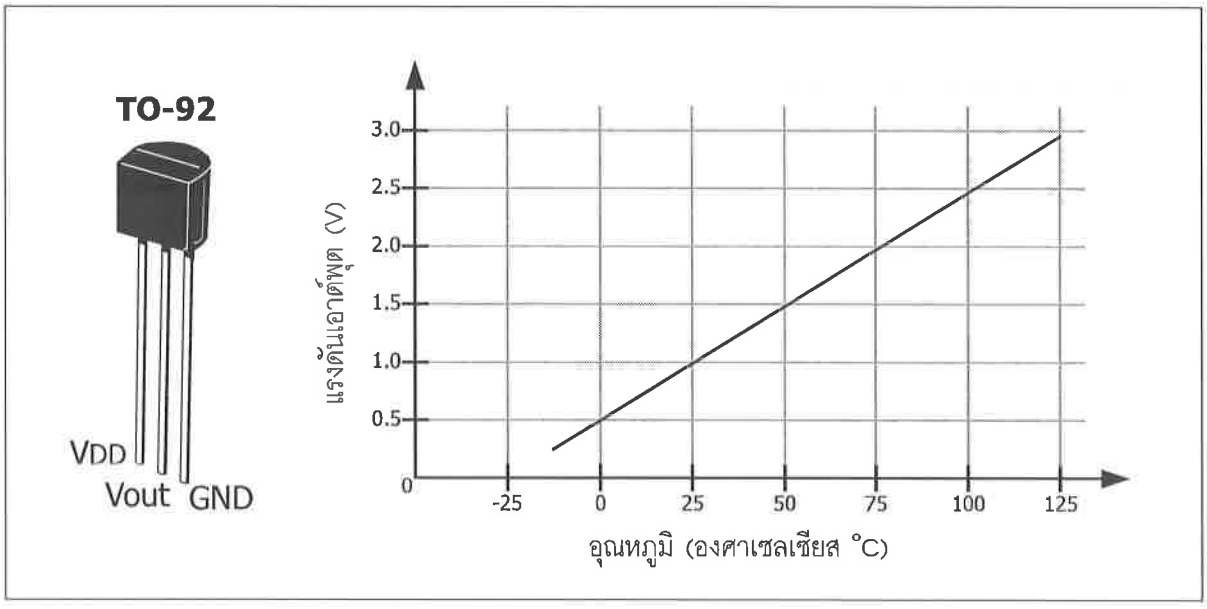
โดยตัวอย่างในหัวข้อนี้มี 2 ตัวอย่างคือ การติดต่อกับตัวตรวจจับอุณหภูมิ 1 ช่องและ 4 ช่อง

9.4.1 แนะนำ MCP9701 ไอซีวัดอุณหภูมิ

เป็นอุปกรณ์ตรวจจับและวัดอุณหภูมิที่ให้ผลการทำงานเป็นแรงดันไฟฟ้าแบบเชิงเส้น รับรู้การเปลี่ยนแปลงของอุณหภูมิภายในเวลาไม่ถึง 2 วินาที เชื่อมต่อกับอินพุตอะนาลอก A0 ถึง A6 ของแผงวงจรหลัก IPST-SE ได้

คุณสมบัติทางเทคนิคของ MCP9701 ที่ควรทราบ

- เป็นไอซีวัดอุณหภูมิในกลุ่มเทอร์มิสเตอร์แบบแอกตีฟที่ให้ผลการทำงานแบบเชิงเส้น
- ย่านวัด -40 ถึง +125 องศาเซลเซียส
- ผลการวัดอ้างอิงกับหน่วยขององศาเซลเซียสโดยตรง
- ความผิดพลาดเฉลี่ย ±2 องศาเซลเซียส
- ย่านไฟเลี้ยง +3.1 ถึง +5.5V กินกระแสไฟฟ้าเพียง 6uA ใช้แบตเตอรี่เป็นแหล่งจ่ายไฟได้
- ค่าแรงดันเอาต์พุต 500mV (ที่ 0°C) ถึง 2.9375V (ที่ 125°)
- ค่าแรงดันเอาต์พุตต่อการเปลี่ยนแปลงอุณหภูมิ 19.5mV/°C ใช้งานกับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลความละเอียดตั้งแต่ 8 บิตได้ โดยมีความคลาดเคลื่อนต่ำ



รูปที่ 9-5 การจัดขาของ MCP9701 และกราฟคุณสมบัติการทำงาน

9.4.2 TempMonitor ระบบตรวจจับอุณหภูมิ 1 ช่อง

ในตัวอย่างนี้เป็นการสร้างระบบอ่านและแสดงผลค่าอุณหภูมิ 1 ช่องโดยใช้ Raspberry Pi 2, ไอซี MCP3208 และ MCP9701 ซึ่งเป็นไอซีวัดอุณหภูมิที่ให้ผลการทำงานเป็นแรงดัน ดังมีไดอะแกรมการทำงานในภาพรวมแสดงในรูปที่ 9-6

สำหรับขั้นตอนการทดลองและทดสอบมีดังนี้

(1) ต่อวงจรตามรูปที่ 9-7

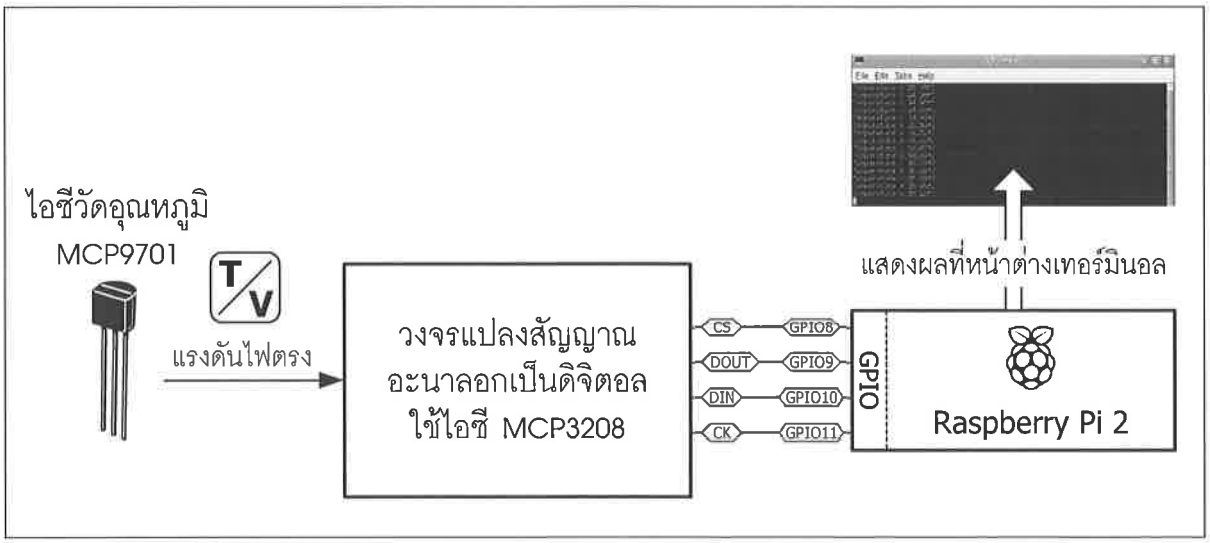
(2) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) เขียนคำสั่งตามโปรแกรมที่ 9-2 บันทึกไฟล์เป็น TempMonitor.py

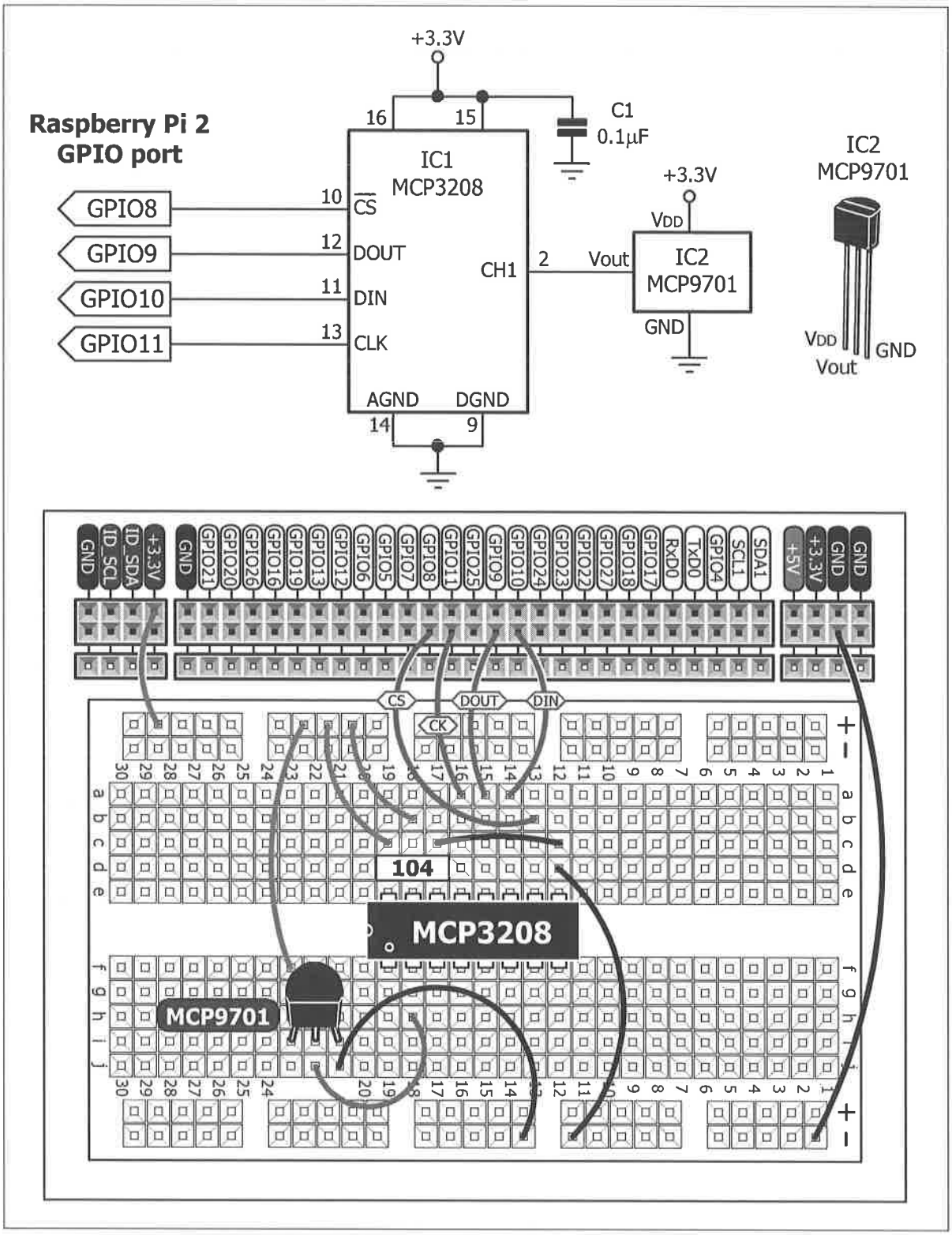
(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **`sudo python3 TempMonitor.py`**

(5) ทดสอบการทำงานด้วยการให้ความร้อนแก่ไอซี MCP9701 โดยใช้นิ้วจับตัวถังของไอซีเพื่อถ่ายเทความร้อน แล้วใช้ลมเป่าเพื่อลดความร้อน ดูผลการทำงานที่เกิดขึ้นผ่านทางหน้าต่างเทอร์มินอล

ตัวอย่างนี้ Raspberry Pi 2 ทำการอ่านค่าข้อมูลที่ได้จากการแปลงสัญญาณอะนาลอกเป็นดิจิตอลของไอซี MCP3208 ซึ่งแรงดันอะนาลอกอินพุตได้มาจากการทำงานของไอซีวัดอุณหภูมิ MCP9701 เมื่อได้ค่ามาแล้ว Raspberry Pi 2 จะทำการคำนวณเพื่อเปลี่ยนข้อมูลดิบมาเป็นข้อมูลอุณหภูมิที่เข้าใจได้ง่ายในหน่วยองศาเซลเซียส แล้วนำมาแสดงที่หน้าต่างเทอร์มินอล ดังรูปที่ 9-8



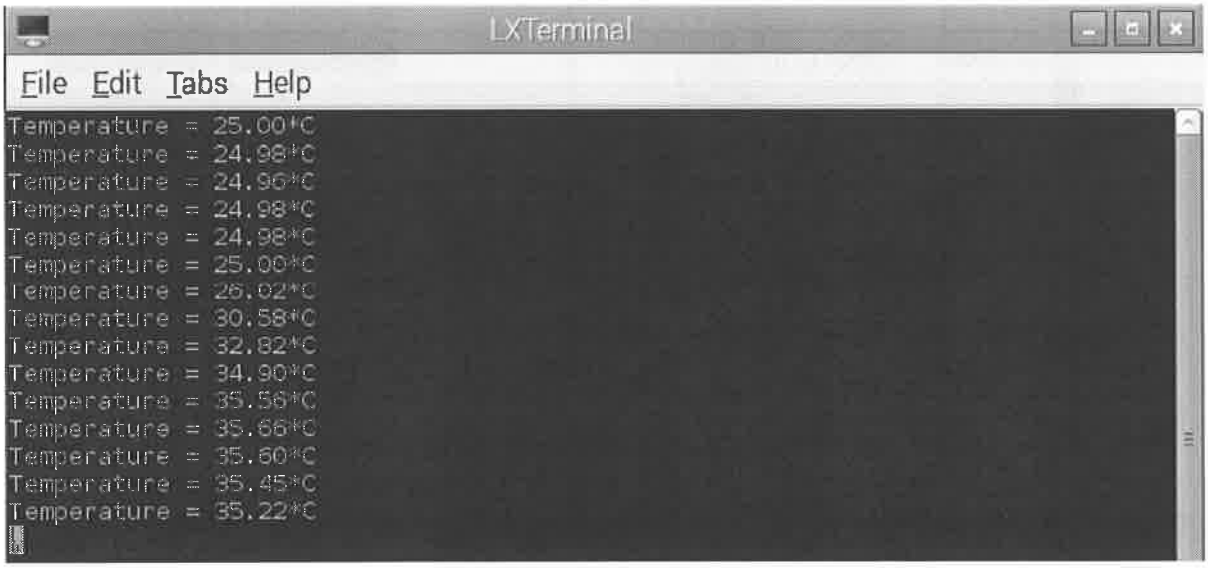
รูปที่ 9-6 ไดอะแกรมการทำงานของ TempMonitor ระบบอ่านและแสดงค่าอุณหภูมิที่ใช้ Raspberry Pi 2 เป็นอุปกรณ์ควบคุมหลัก



รูปที่ 9-7 วงจรและตัวอย่างการเชื่อมต่อบอร์ด Raspberry Pi 2 กับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลที่ใช้ไอซีเบอร์ MCP3208 ผ่านทางขาพอร์ต GPIO และไอซี MCP9701 เพื่อสร้างระบบอ่านและแสดงผลค่าอุณหภูมิ

```
import spidev
import time
analog_ch = 1

spi = spidev.SpiDev()
spi.open(0,0)

def readADC(adcnum):
  if adcnum > 7 or adcnum < 0:
     retrun -1;
  r = spi.xfer2([4|2|(adcnum >> 2), (adcnum & 3) << 6, 0])
  adcout = ((r[1] & 15) << 8) + r[2]
  return adcout

while True:
  value = readADC(analog_ch)
  voltage = value*3.3/4096
  temp = (voltage-0.5)/0.0195
  print("Temperature = %2.2f*C" % temp)
  time.sleep(0.3)
```

โปรแกรมที่ 9-2 โค้ด Python ไฟล์ TempMonitor.py สำหรับระบบอ่านและแสดงผลค่าอุณหภูมิที่ใช้ไอซี MCP9701 ในการวัดอุณหภูมิ แล้วส่งค่าผ่านไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลเบอร์ MCP3208 ที่ได้รับการควบคุมบอร์ด Raspberry PI 2 เพื่อนำค่ามาแสดงผลที่หน้าต่างเทอร์มินอล



รูปที่ 9-8 หน้าต่าง LX Terminal แสดงผลการทำงานของโปรแกรมที่ 9-2 บอร์ด Raspberry Pi 2 แสดงค่าอุณหภูมิที่วัดโดยไอซีวัดอุณหภูมิเบอร์ MCP9701

9.4.3 TempMonitor4 ระบบตรวจจับอุณหภูมิ 4 ช่อง

ในตัวอย่างนี้เป็นการต่อยอดจากระบบ TempMonitor ที่เดิมอ่านและแสดงผลค่าอุณหภูมิได้ 1 ช่อง โดยในตัวอย่างนี้จะเพิ่มเป็น 4 ช่อง นั่นคือมีการเพิ่มไอซีวัดอุณหภูมิ MCP9701 เป็น 4 ตัว ต่อเข้ากับไอซี MCP3208 โดยการควบคุมและแสดงผลการทำงานยังกระทำโดยบอร์ด Raspberry Pi 2 ดังมีไดอะแกรมการทำงานในภาพรวมแสดงในรูปที่ 9-9

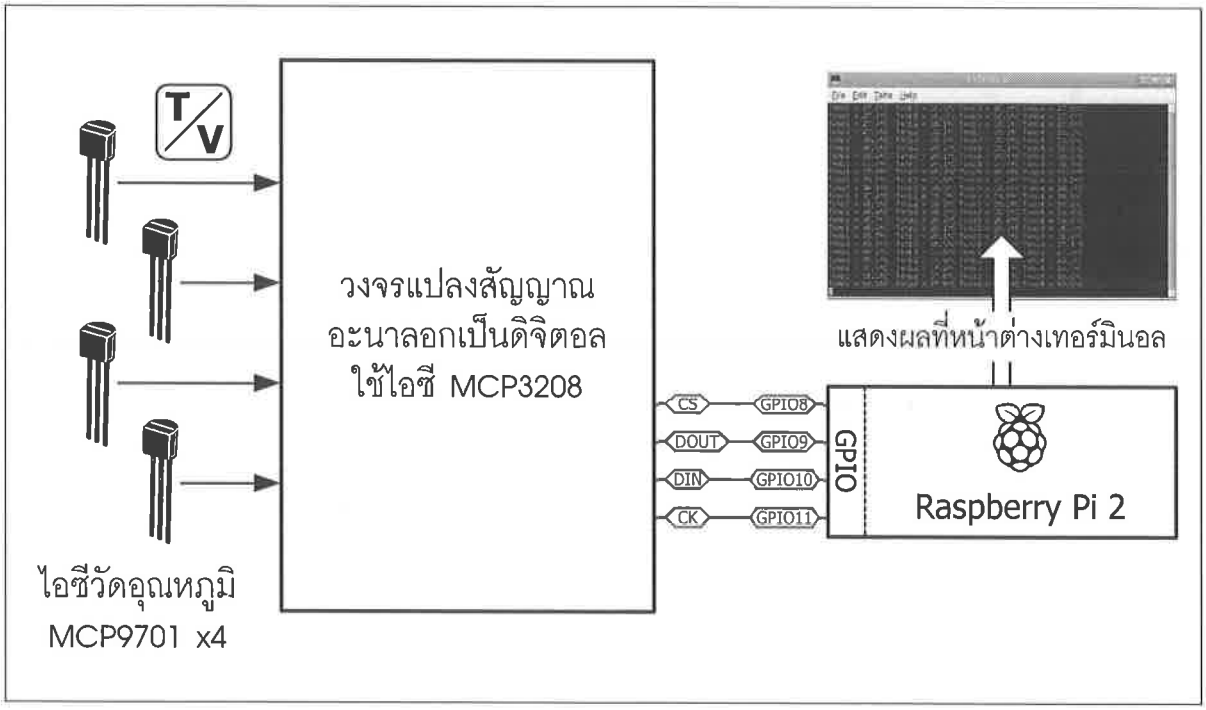
สำหรับขั้นตอนการทดลองและทดสอบมีดังนี้

(1) ต่อวงจรตามรูปที่ 9-10

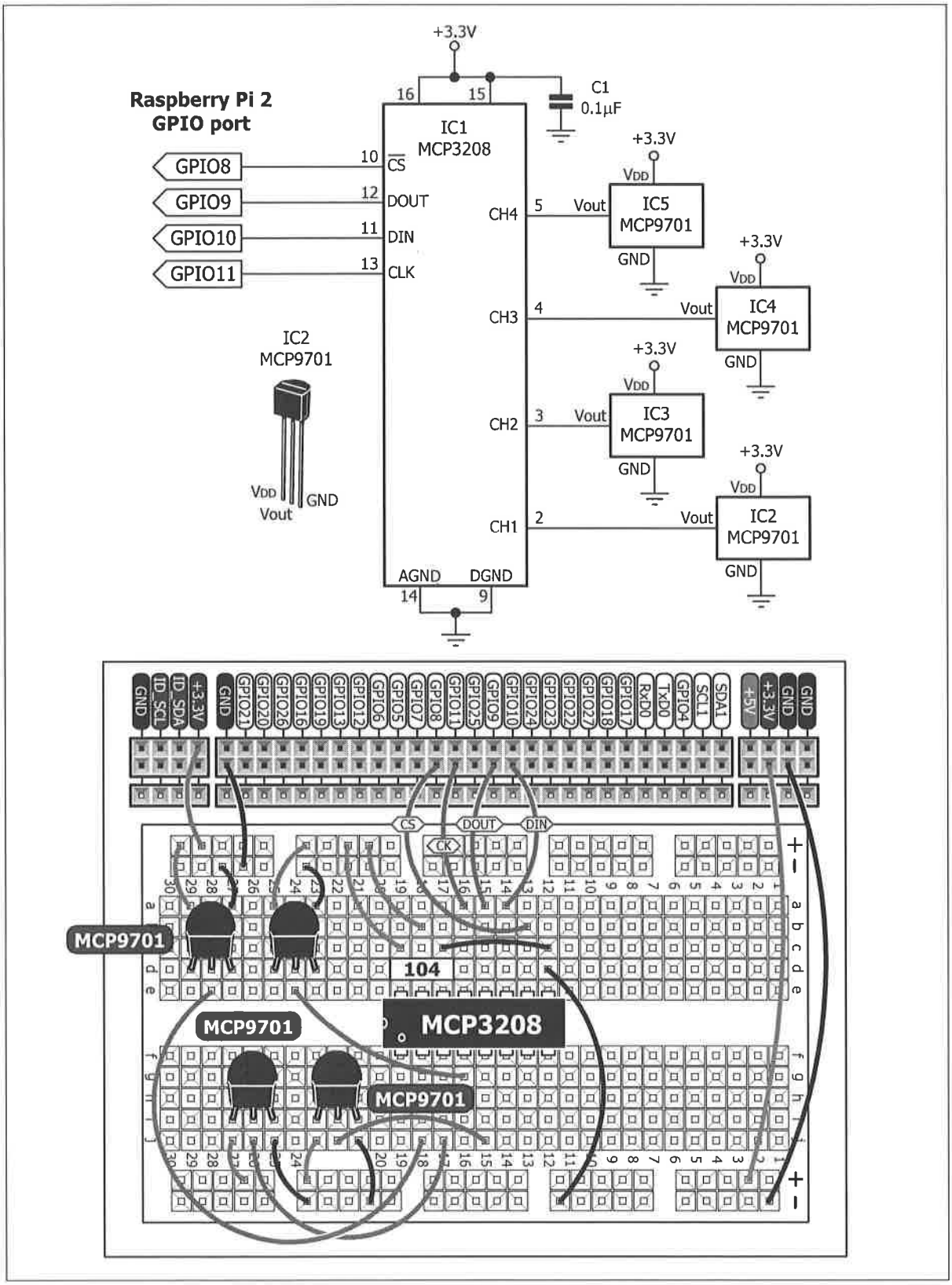
(2) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) เขียนคำสั่งตามโปรแกรมที่ 9-3 บันทึกไฟล์เป็น TempMonitor4.py

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **`sudo python3 TempMonitor4.py`**



รูปที่ 9-9 ไดอะแกรมการทำงานของ TempMonitor4 ระบบอ่านและแสดงค่าอุณหภูมิ 4 ช่องที่ใช้ Raspberry Pi 2 เป็นอุปกรณ์ควบคุมหลัก



รูปที่ 9-10 วงจรและตัวอย่างการเชื่อมต่อบอร์ด Raspberry Pi 2 กับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลที่ใช้ไอซีเบอร์ MCP3208 ผ่านทางขาพอร์ต GPIO และไอซี MCP9701 เพื่อสร้างระบบอ่านและแสดงผลค่าอุณหภูมิ 4 ช่อง

```
import spidev
import time
value = [0,0,0,0,0]
volt = [0,0,0,0,0]
temp = [0,0,0,0,0]

spi = spidev.SpiDev()
spi.open(0,0)

def readADC(adcnum):
  if adcnum > 7 or adcnum < 0:
     retrun -1;
  r = spi.xfer2([4|2|(adcnum >> 2), (adcnum & 3) << 6, 0])
  adcout = ((r[1] & 15) << 8) + r[2]
  return adcout

while True:
  for ch in range (0,4):
     value[ch] = readADC(ch)
     volt[ch] = value[ch]*3.3/4096
     temp[ch] = (volt[ch-0.5)/0.0195

  print('Temp1 = %0.1f*C  Temp2 = %0.1f*C Temp3 = %0.1f*C Temp4 = %0.1f*C'
  % ((temp[0],temp[1], temp[2], temp[3])))
  time.sleep(0.3)
```

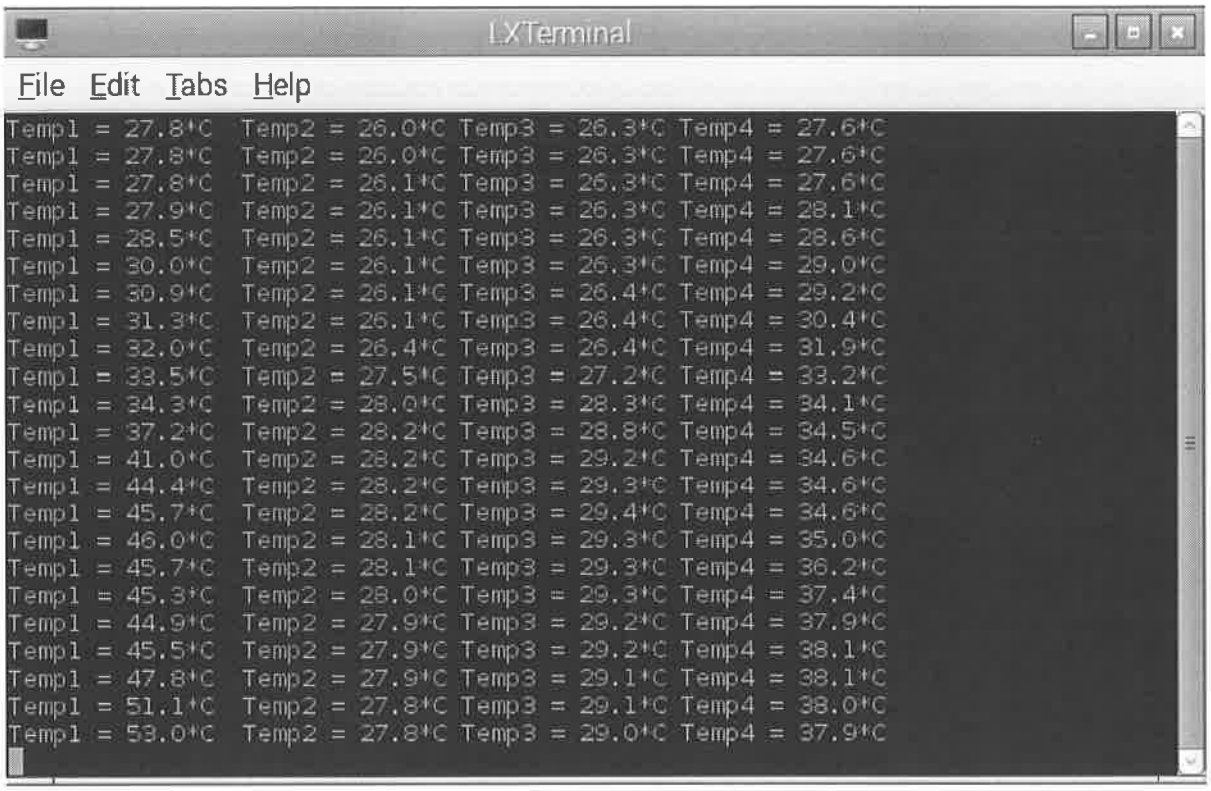
คำอธิบายโปรแกรมเพิ่มเติม

ในโปรแกรมนี้ แบ่งการทำงานเป็น 3 ส่วนหลักคือ ส่วนของการติดต่อกับ MCP3208 ด้วยไลบรารี spidev, ส่วนของการวนอ่านค่าอะนาลอก 4 ช่อง และคำสั่งแสดงค่าการทำงาน ในการอ่านค่าจะมีการคำนวณข้อมูลดิบเป็นค่าอุณหภูมิในหน่วยองศาเซลเซียสด้วยคำสั่ง `volt[ch] = value[ch]*3.3/4096` เพื่อแปลงเป็นแรงดันก่อน จากนั้นใช้คำสั่ง `temp[ch] = (volt[ch-0.5)/0.0195` เพื่อแปลงเป็นค่าอุณหภูมิ แล้วนำไปแสดงที่หน้าต่างเทอร์มินอล ด้วยคำสั่ง `print('Temp1 = %0.1f*C Temp2 = %0.1f*C Temp3 = %0.1f*C Temp4 = %0.1f*C'`

โปรแกรมที่ 9-3 โค้ด Python ไฟล์ TempMonitor4.py สำหรับระบบอ่านและแสดงผลค่าอุณหภูมิ 4 ช่อง ที่ใช้ไอซี MCP9701 ในการวัดอุณหภูมิ แล้วส่งค่าผ่านไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลเบอร์ MCP3208 ที่ได้รับการควบคุมบอร์ด Raspberry PI 2 เพื่อนำค่ามาแสดงผลที่หน้าต่างเทอร์มินอล

(5) ทดสอบการทำงานด้วยการให้ความร้อนแก่ไอซี MCP9701 แต่ละตัว สลับไปมา ดูผลการทำงานที่เกิดขึ้นผ่านทางหน้าต่างเทอร์มินอล

ตัวอย่างนี้ Raspberry Pi 2 ทำการอ่านค่าข้อมูลที่ได้จากการแปลงสัญญาณอะนาลอกเป็นดิจิตอลของไอซี MCP3208 จำนวน 4 ช่อง ซึ่งแรงดันอะนาลอกอินพุตได้มาจากการทำงานของไอซีวัดอุณหภูมิ MCP9701 รวม 4 ตัว เมื่อได้ค่ามาแล้ว Raspberry Pi 2 จะทำการคำนวณเพื่อเปลี่ยนข้อมูลดิบมาเป็นข้อมูลอุณหภูมิที่เข้าใจได้ง่ายในหน่วยองศาเซลเซียส แล้วนำมาแสดงที่หน้าต่างเทอร์มินอล ดังรูปที่ 9-11



รูปที่ 9-11 หน้าต่าง LX Terminal แสดงผลการทำงานของโปรแกรมที่ 9-3 บอร์ด Raspberry Pi 2 แสดงค่าอุณหภูมิที่วัดโดยไอซีวัดอุณหภูมิเบอร์ MCP9701 จำนวน 4 ช่อง

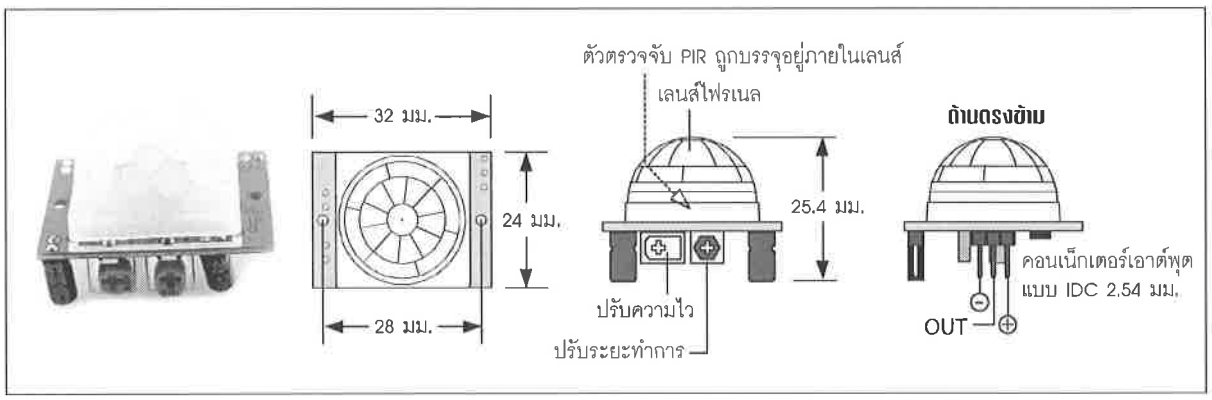
บทที่ 10
Raspberry Pi 2 กับการติดต่อตัวตรวจจับสภาพแวดล้อม

การเรียนรู้และใช้งานระบบสมองกลฝังตัวสมัยใหม่ จำเป็นอย่างยิ่งที่ต้องรู้จักและใช้งานอุปกรณ์ต่อพ่วงที่มีความสามารถแตกต่างออกไป โดยเฉพาะอย่างยิ่งตัวตรวจจับหรือเซนเซอร์ (sensor) ในแบบต่างๆ และยิ่งเมื่อตัวควบคุมหลักมีความสามารถสูงขึ้นมากอย่างบอร์ดคอมพิวเตอร์ Raspberry Pi 2 ทำให้การใช้ประโยชน์จากตัวตรวจจับทำได้อย่างมีประสิทธิภาพมากขึ้น กอรปกับการพัฒนาอย่างต่อเนื่องของผู้ผลิต ทำให้ผู้เริ่มต้นเรียนรู้ได้ใช้งานตัวตรวจจับปริมาณทางกายภาพแบบใหม่ๆ ในราคาที่เข้าถึงได้

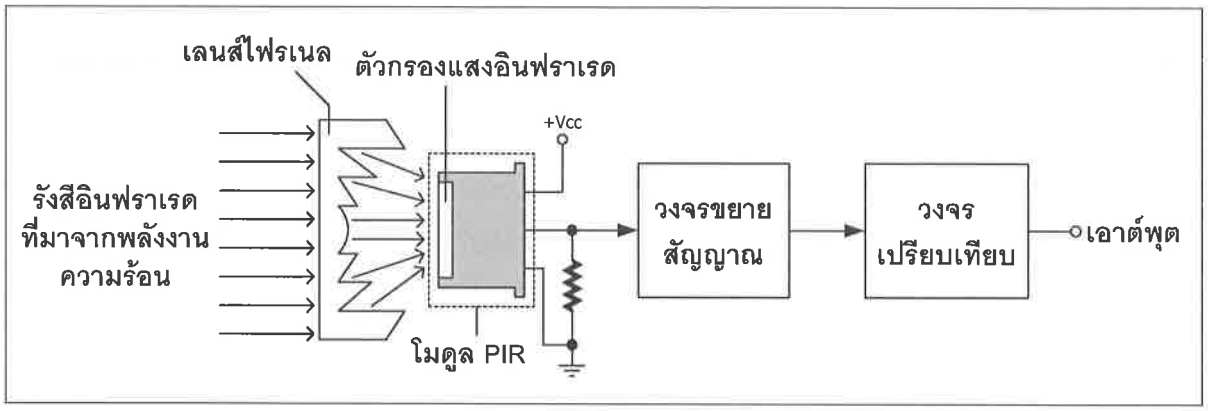
ในบทนี้แนะนำถึงการใช้งานบอร์ด Raspberry Pi 2 กับตัวตรวจจับสภาพแวดล้อมที่น่าสนใจ 3 ตัวคือ **PIR** หรือโมดูลตรวจจับความเคลื่อนไหว, **DHT11** โมดูลตรวจจับความชื้นและอุณหภูมิ และ **BH1750** โมดูลวัดความเข้มแสงที่ให้ผลการทำงานเป็นหน่วยลักซ์

10.1 ใช้งานโมดูลตรวจจับความเคลื่อนไหว

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



รูปที่ 10-1 แสดงลักษณะทางกายภาพของ ZX-PIR2.0 โมดูลตรวจจับความเคลื่อนไหว



รูปที่ 10-2 ไดอะแกรมการทำงานของตัวตรวจจับการแผ่รังสีอินฟราเรดซึ่งใช้ตรวจจับความเคลื่อนไหว

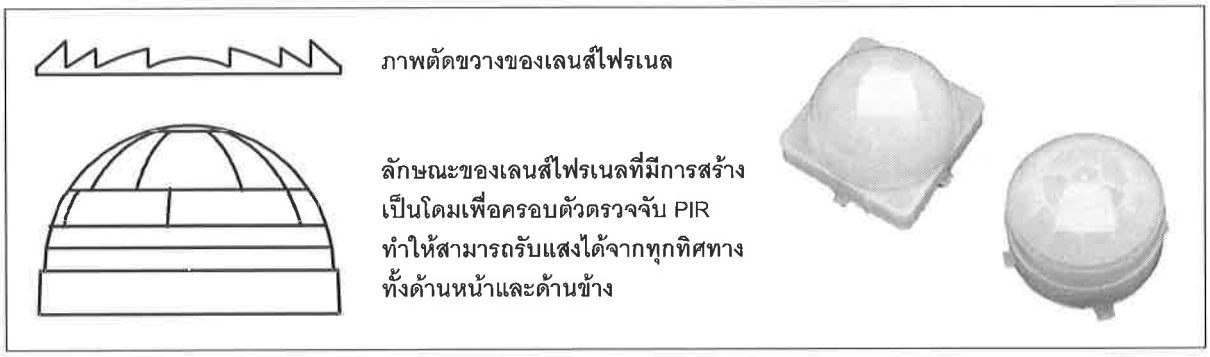
10.1.1 หลักการตรวจจับความเคลื่อนไหว

ตัวตรวจจับความเคลื่อนไหวของสิ่งมีชีวิต (motion sensor) ที่ได้รับความนิยมและใช้งานง่ายคือ **ตัวตรวจจับแบบอินฟราเรด** ใช้หลักการตรวจจับที่เรียกว่า **ไพโรอิเล็กตริก (pyro-electric)** *อันเป็นการตรวจจับการแผ่รังสีอินฟราเรด หากระดับของการแผ่รังสีไม่เปลี่ยนแปลง แสดงว่า สิ่งมีชีวิตที่ต้องการตรวจจับนั้นไม่มีการเคลื่อนไหว แต่ถ้าหากมีการเคลื่อนไหวเกิดขึ้น ระดับของการแผ่รังสีอินฟราเรดจะเปลี่ยนแปลง*

สิ่งมีชีวิตไม่ว่าจะเป็นมนุษย์หรือสัตว์เลือดอุ่นในภาวะที่ยังมีชีวิตอยู่จะกระจายพลังงานความร้อนออกมาจากตัวเองในรูปของการแผ่รังสีอินฟราเรดตลอดเวลา โดยจะมีปริมาณมากหรือน้อยขึ้นอยู่กับสภาพของร่างกายในขณะนั้น เมื่อมีการเคลื่อนไหว ปริมาณของการแผ่รังสีก็จะเปลี่ยนแปลง รังสีอินฟราเรดจากมนุษย์หรือสัตว์เลือดอุ่นที่มีระดับความเข้มสูงสุดจะมีความยาวคลื่นประมาณ 9.4 ไมโครเมตร

ในรูปที่ 10-2 เป็นไดอะแกรมแสดงหลักการทำงานพื้นฐานของตัวตรวจจับ PIR พลังงานความร้อนจากมนุษย์หรือสัตว์เลือดอุ่นทำให้เกิดการแผ่รังสีอินฟราเรดขึ้น รังสีจะถูกรวมหรือโฟกัสไปยังตัวตรวจจับหลักโดยใช้เลนส์แบบพิเศษที่เรียกว่า *เลนส์ไฟรเนล (Fresnel lens)* จากนั้นตัวตรวจจับหลักจะทำการขยายสัญญาณแล้วส่งไปยังวงจรเปรียบเทียบเพื่อสร้างสัญญาณเอาต์พุตต่อไป

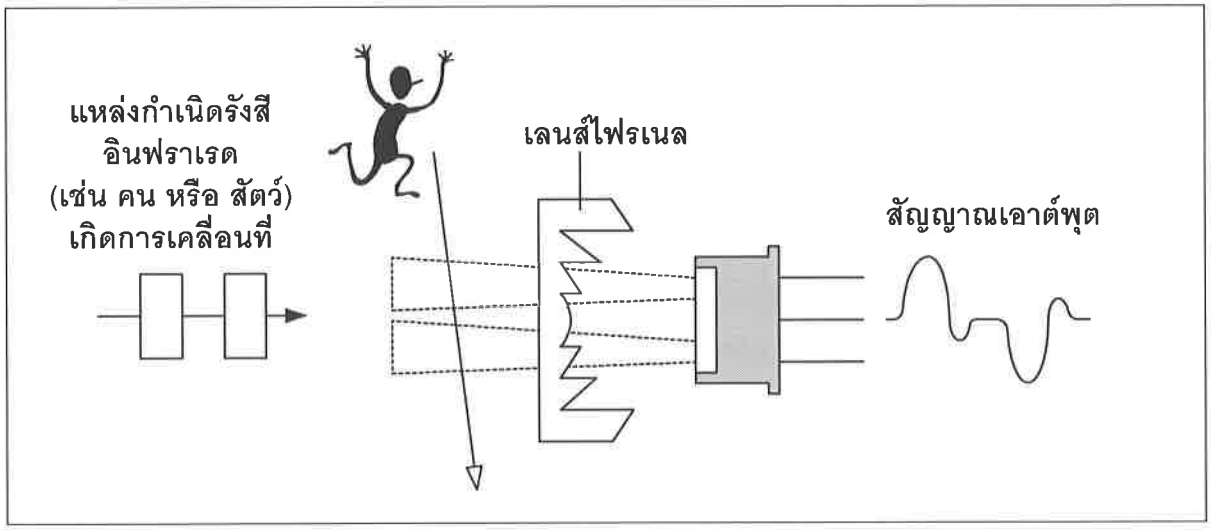
เลนส์ไฟรเนลเป็นเลนส์แบบพิเศษที่ได้รับการค้นคิดจากนักฟิสิกส์ชาวฝรั่งเศสชื่อ *ออกัสติน ชอง ไฟรเนล (Augustin-Jean Fresnel)* โดยแนวคิดของเลนส์แบบนี้คือ เป็นเลนส์แบบขั้นบันไดที่ยอมให้แสงผ่านได้มากและจากทุกทิศทาง ดังมีโครงสร้างตามรูปที่ 10-3 ทั้งนี้เนื่องจากตัวเลนส์ได้ถูกสร้างขึ้นโดยลดเนื้อวัสดุในส่วนที่ไม่มีผลกับการหักเหของแสงลงไป ทำให้สามารถทำเลนส์ขนาดใหญ่ที่มีน้ำหนักเบาได้ เดิมทีเลนส์ไฟรเนลนี้ได้รับการออกแบบเพื่อให้นำมาใช้ในการกระจายในประภาคาร เพื่อให้สามารถมองเห็นประภาคารได้จากระยะไกล ต่อมาได้มีการพัฒนาให้มีขนาดเล็กลง แล้วนำมาครอบหลอดไฟเพื่อทำเป็นตะเกียง ทำให้ตะเกียงสามารถส่องแสงได้สว่างและมองเห็นได้จากระยะไกล



รูปที่ 10-3 แสดงโครงสร้างและหน้าตาของเลนส์ไฟรเนลซึ่งนำมาใช้ในโมดูล PIR

แต่ในโมดูล ZX-PIR นำเลนส์ไฟรเนลมาใช้งานในลักษณะกลับกันคือ ใช้เลนส์ไฟรเนลในการรวมแสงเข้ามาจากทุกทิศทางเพื่อโฟกัสลงไปยังส่วนตรวจจับแสงอินฟราเรดของตัวตรวจจับ PIR เพื่อให้การตรวจจับการเปลี่ยนแปลงของรังสีอินฟราเรดมีความไวสูง

ในรูปที่ 10-4 แสดงสถานการณ์ที่แหล่งกำเนิดรังสีอินฟราเรด (อาจเป็นมนุษย์หรือสัตว์เลือดอุ่น) เกิดการเคลื่อนไหวภายในระยะทำการของตัวตรวจจับ จะทำให้ตัวตรวจจับ PIR ตรวจจับพบการแผ่รังสีอินฟราเรดที่แตกต่างกัน จึงให้สัญญาณเอาต์พุตเป็นลอจิกสูง (high) อยู่ชั่วขณะเมื่อตรวจจับพบการเคลื่อนไหว จากนั้นกลับมาเป็นลอจิกต่ำ (low) จนกว่าจะตรวจจับพบการเปลี่ยนแปลงของระดับรังสีอินฟราเรดอีกครั้ง



รูปที่ 10-4 แสดงการทำงานของโมดูล PIR เมื่อนำมาใช้ในการตรวจจับความเคลื่อนไหว

10.1.2 คุณสมบัติทางเทคนิคของโมดูล ZX-PIR2.0

ZX-PIR2.0 เป็นโมดูลตรวจจับความเคลื่อนไหวที่ติดตั้งเลนส์ไฟรเนลครอบตัวโมดูล PIR มีวงจรขยายสัญญาณและวงจรเปรียบเทียบรวมกันไว้บนแผงวงจรเดียวกัน คุณสมบัติทางเทคนิคมีดังนี้

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

10.1.3 การใช้งานโมดูล ZX-PIR

จากรูปที่ 10-1 ซึ่งแสดงลักษณะทางกายภาพของโมดูล ZX-PIR2.0 และขาต่อใช้งาน โดยมีข้อมูลเพิ่มเติมดังนี้

ตำแหน่งขา	ชื่อขา	คำอธิบาย
−	GND	ต่อกับกราวด์
+	Vcc	ต่อกับไฟเลี้ยง +3V ถึง +5V
OUT	Output	เป็นขาเอาต์พุตของโมดูล ใช้ต่อกับไมโครคอนโทรลเลอร์หรืออินพุตของวงจรขับโหลดใด ๆ

10.1.4. การเชื่อมต่อ ZX-PIR2.0 กับ Raspberry Pi 2

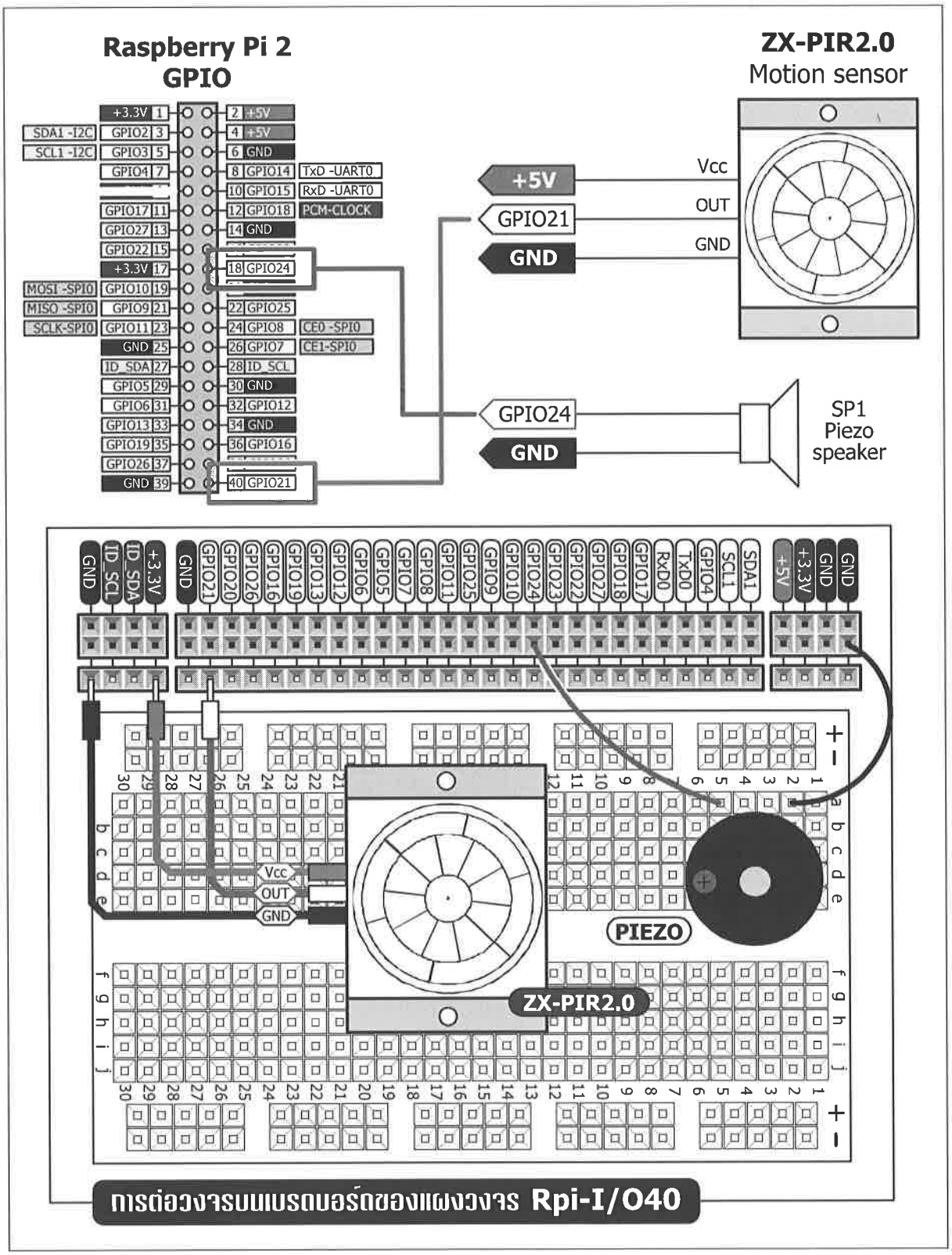
เนื่องจากเอาต์พุตของ ZX-PIR2.0 เป็นสัญญาณดิจิตอลที่มีสองสถานะคือ ลอจิกสูง หรือ “1” และลอจิกต่ำหรือ “0” จึงสามารถเชื่อมต่อกับขาพอร์ตดิจิตอลของไมโครคอนโทรลเลอร์และพอร์ต GPIO ของบอร์ด Raspberry Pi 2 ได้ทุกขา โดยต้องกำหนดให้ขาพอร์ตที่เชื่อมต่อนั้นเป็นอินพุตดิจิตอลก่อน และไม่ต้องต่อตัวต้านทานพูลอัปที่ขาพอร์ต แต่อย่างใด

10.1.5 ทดลองอ่านค่าการตรวจจับความเคลื่อนไหวของโมดูล ZX-PIR2.0

ขั้นตอนการทดลองและทดสอบมีดังนี้

(1) ต่อวงจรตามรูปที่ 10-5

(2) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์



รูปที่ 10-5 วงจรและตัวอย่างการเชื่อมต่อบอร์ด Raspberry Pi 2 กับโมดูล ZX-PIR2.0 เพื่อสร้างระบบตรวจจับความเคลื่อนไหวของมนุษย์โดยมีการแจ้งเตือนด้วยเสียงสัญญาณผ่านลำโพงเปียโซ

```
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(18,500)
blink.start(0)
GPIO.setup(21,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(21)

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

โปรแกรมที่ 10-1 โค้ด Python ไฟล์ MovementAlarm.py สำหรับบอร์ด Raspberry PI 2 ติดต่อกับโมดูล ZX-PIR2.0 เพื่อตรวจจับความเคลื่อนไหว หากตรวจพบ จะขับสัญญาณเสียงเตือนออกมาทางลำโพงเปียโซ

(3) เขียนคำสั่งตามโปรแกรมที่ 10-1 บันทึกไฟล์เป็น MovementAlarm.py

(4) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **`sudo python3 MovementAlarm.py`** เพื่อรันโปรแกรม

เมื่อตัวตรวจจับตรวจพบความเคลื่อนไหว บอร์ด Raspberry Pi 2 จะสร้างสัญญาณ PWM ส่งออกมาทางขา GPIO24 เพื่อขับให้ได้ยินผ่านลำโพงเปียโซ

10.2 ใช้งาน ZX-DHT11 บอร์ดวัดความชื้นสัมพัทธ์และอุณหภูมิ

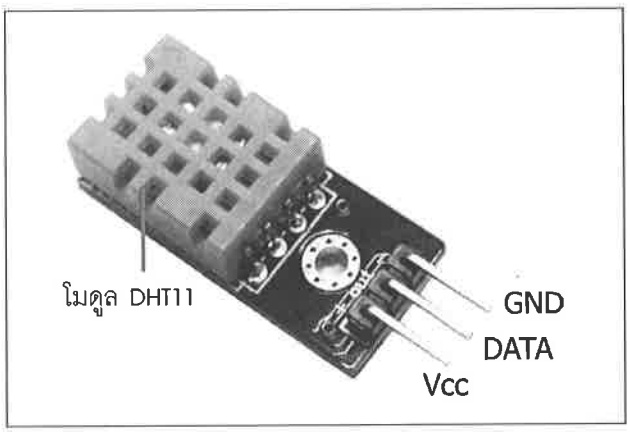
ZX-DHT11 เป็นแผงวงจรขนาดเล็กที่บรรจุโมดูลตรวจจับและวัดความชื้นสัมพัทธ์เบอร์ DHT11 ซึ่งนอกจากจะวัดความชื้นสัมพัทธ์ได้แล้ว ยังให้ค่าของอุณหภูมิของพื้นที่ที่ตรวจวัดความชื้นด้วย การติดต่อเป็นแบบหนึ่งสาย นั่นคือใช้ขาพอร์ตของไมโครคอนโทรลเลอร์หรือ GPIO ของ Raspberry Pi 2 เพียง 1 หนึ่งขาในการทำงาน

10.2.1 คุณสมบัติทางเทคนิคของ ZX-DHT11

ZX-DHT มีคุณสมบัติทางเทคนิคที่ควรทราบเพื่อเป็นข้อมูลประกอบในการใช้งานดังนี้

- ใช้โมดูล DHT11 ติดตั้งบนแผ่นวงจรพิมพ์
- มีตัวต้านทานต่อพูลอัปที่ขา DATA ทำให้เชื่อมต่อกับขาพอร์ตของไมโครคอนโทรลเลอร์หรือ GPIO ของ Raspberry Pi 2 ได้โดยไม่ต้องต่อตัวต้านทานเพิ่ม
- ใช้ไฟเลี้ยง +3 ถึง +5.5V ต้องการกระแสไฟฟ้า 2.5mA ในขณะทำการวัดค่า และ 0.5mA ในโหมดสลีป
- วัดความชื้นสัมพัทธ์ได้ 20 ถึง 80%RH มีความผิดพลาด ±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 10-6 แสดงหน้าตาของ ZX-DHT11 และการจัดขา



รูปที่ 10-6 หน้าตาและการจัดขาของ ZX-DHT11 แผงวงจรวัดความชื้นสัมพัทธ์และอุณหภูมิ

หมายเหตุ : มีผู้ผลิตแผงวงจรตรวจจับที่ใช้โมดูล DHT11 หลายราย อาจมีการจัดขาที่ต่างไปจากนี้ ดังนั้น จึงควรตรวจสอบตำแหน่งขาให้ถูกต้องก่อนเชื่อมต่อเพื่อใช้งาน

10.2.2 การติดต่อกับ DHT11

โมดูล DHT11 ติดต่อกับไมโครคอนโทรลเลอร์ด้วยสายสัญญาณเพียง 1 เส้น บางครั้งเรียกการติดต่อแบบนี้ว่า การสื่อสารแบบบัสเดี่ยว (single bus communication) ดังนั้นจึงต้องมีการกำหนดจังหวะระหว่างไมโครคอนโทรลเลอร์และ DHT11 ให้เหมาะสม (synchronization) จึงจะอ่านและเขียนค่ากับ DHT11 ได้อย่างถูกต้อง

10.2.2.1 รูปแบบข้อมูลของ DHT11

ข้อมูลใน 1 รอบของการติดต่อมีทั้งสิ้น 40 บิต และ DHT11 จะส่งข้อมูลบิตนัยสำคัญสูงสุดออกมาก่อนเป็นบิตแรก รูปแบบของข้อมูลเป็นดังนี้

ไบต์ที่	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

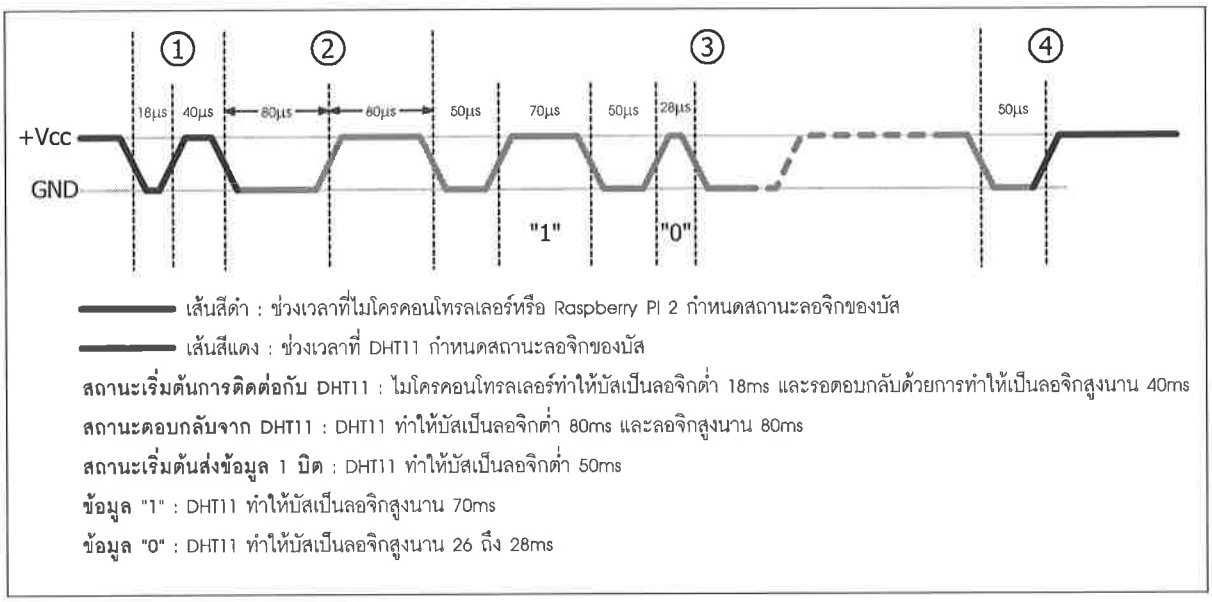
10.2.2.2 กระบวนการสื่อสารกับ DHT11

ในรูปที่ 10-7 เป็นไดอะแกรมเวลาแสดงกระบวนการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์กับ DHT11 โดยเริ่มจาก

(1) ไมโครคอนโทรลเลอร์ส่งสัญญาณเริ่มต้น ด้วยการทำให้บัสเป็นลอจิกต่ำนานอย่างน้อย 18 ไมโครวินาที จากนั้นทำให้บัสกลับมาเป็นลอจิกสูง เพื่อรอการตอบสนองจาก 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 ข้อมูลแต่ละบิตจะถูกคั่นด้วยสถานะลอจิกต่ำนาน 50 ไมโครวินาที



รูปที่ 10-7 ไดอะแกรมเวลาแสดงกระบวนการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์กับ DHT11

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

10.2.3 การพัฒนาโปรแกรมเพื่อใช้งาน DHT11 กับบอร์ด Raspberry Pi 2 ด้วย Python3

DHT11 เป็นโมดูลที่ใช้ในการวัดอุณหภูมิกับความชื้นในอากาศที่มีการรับส่งข้อมูลด้วยสายเส้นเดียว โดยเดิมทีการพัฒนาโปรแกรมสำหรับ Raspberry Pi 2 ด้วยภาษา Python เพื่อติดต่อและอ่านค่าจาก DHT11 มีไลบรารีของ Adafruit ให้เรียกใช้งานอยู่แล้ว โดยดาวน์โหลดได้จาก ***https://github.com/adafruit/Adafruit_Python_DHT*** ทว่าไลบรารีของ Adafruit นั้นรองรับถึง Python2 เท่านั้น

ในการพัฒนาโปรแกรมด้วย Python3 จึงต้องเปลี่ยนไลบรารีใหม่ เนื่องจากไลบรารีดังกล่าวได้เปิดให้นักพัฒนาคนอื่นสามารถเข้ามาแก้ไขได้ จึงได้มีนักพัฒนาที่ชื่อว่า **JoBergs** ได้ทำการแก้ไขให้ไลบรารีดังกล่าวใช้งานกับ Python3 ได้ โดยดูข้อมูลและซอร์สโค้ดได้ที่ ***https://github.com/JoBergs/Adafruit_Python_DHT***

ขั้นตอนการติดตั้งไลบรารีและเรียกใช้งานมีดังนี้

(1) เปิดหน้าต่างเทอร์มินอลของ Raspberry Pi 2 ทำการดาวน์โหลดไลบรารีของ DHT11 ด้วยคำสั่ง

```
git clone https://github.com/JoBergs/Adafruit_Python_DHT
```

```
pi@raspberrypi ~ $ git clone https://github.com/JoBergs/Adafruit_Python_DHT
```

(2) ติดตั้ง Python 3 และชุด Build โปรแกรมด้วยคำสั่ง

```
sudo apt-get install build-essential python3-dev -y
```

```
pi@raspberrypi ~ $ sudo apt-get install build-essential python3-dev -y
```

(3) เปิดเข้าไปในไดเร็กตอรีที่พึ่งดาวน์โหลดมา แล้วทำการติดตั้งด้วยคำสั่ง

```
cd Adafruit_Python_DHT
sudo python3 setup.py install
```

```
pi@raspberrypi ~ $ cd Adafruit_Python_DHT/
pi@raspberrypi ~/Adafruit_Python_DHT $ sudo python3 setup.py install
```

(4) เมื่อติดตั้งเสร็จแล้วให้ทดสอบด้วยการใช้อินเตอร์แอกตีฟเชลของ Python 3 ด้วยคำสั่ง

```
sudo python3
```

(5) จากนั้นให้ทดสอบการทำงานของ DHT11 ด้วยคำสั่ง

```
import Adafruit_DHT
Adafruit_DHT.read_retry(11, 4)
```

```
pi@raspberrypi ~ $ sudo python3
Python 3.2.3 (default, Mar  1 2013, 11:53:50)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import Adafruit_DHT
>>> Adafruit_DHT.read_retry(11, 4)
(55.0, 30.0)
>>>
```

สำหรับ 11 คือใช้งานกับ DHT11 (ไลบรารีชุดนี้รองรับ DHT11, DHT22 และ AM2302) ส่วน 4 ก็คือขาพอร์ต GPIO ของบอร์ด Raspberry Pi 2 ที่เชื่อมต่อกับ DHT11 โดยอิงตำแหน่งขาแบบ BCM

หลังจากรันคำสั่ง จะเห็นว่า มีข้อมูลจาก DHT11 แสดงอยู่ในวงเล็บต่อท้ายคำสั่ง โดยข้อมูลตัวแรกเป็นค่าความชื้นสัมพันธ์ในหน่วย %RH ส่วนข้อมูลตัวที่สองเป็นอุณหภูมิในหน่วยองศาเซลเซียส จากตัวอย่างคือ ความชื้นสัมพัทธ์ 55%RH และอุณหภูมิ 30 องศาเซลเซียส

สำหรับคำสั่ง `read_retry` เป็นคำสั่งอ่านค่าจาก DHT11 โดยคำสั่งนี้จะวนอ่านค่าจาก DHT11 ใหม่ทุกๆ 2 วินาทีจนกว่าจะอ่านค่าได้ และจะวนอ่านค่าใหม่ 15 ครั้งเท่านั้น

เมื่อทดสอบการทำงานเรียบร้อยแล้ว ต่อไปจะเป็นการเขียนโปรแกรมภาษา Python เพื่อใช้งาน DHT11 ด้วยไลบรารีนี้

10.2.4 ตัวอย่างการทดสอบอ่านค่าจาก DHT11 ด้วยบอร์ด Raspberry Pi 2

ตัวอย่างที่นำเสนอต่อไปนี้ เป็นการทดลองติดต่อกับ ZX-DHT11 โดยใช้บอร์ด Raspberry Pi 2 เพื่ออ่านค่าความชื้นสัมพัทธ์และอุณหภูมิจากโมดูล DHT11 โดยพัฒนาโปรแกรมติดต่อด้วย Python3

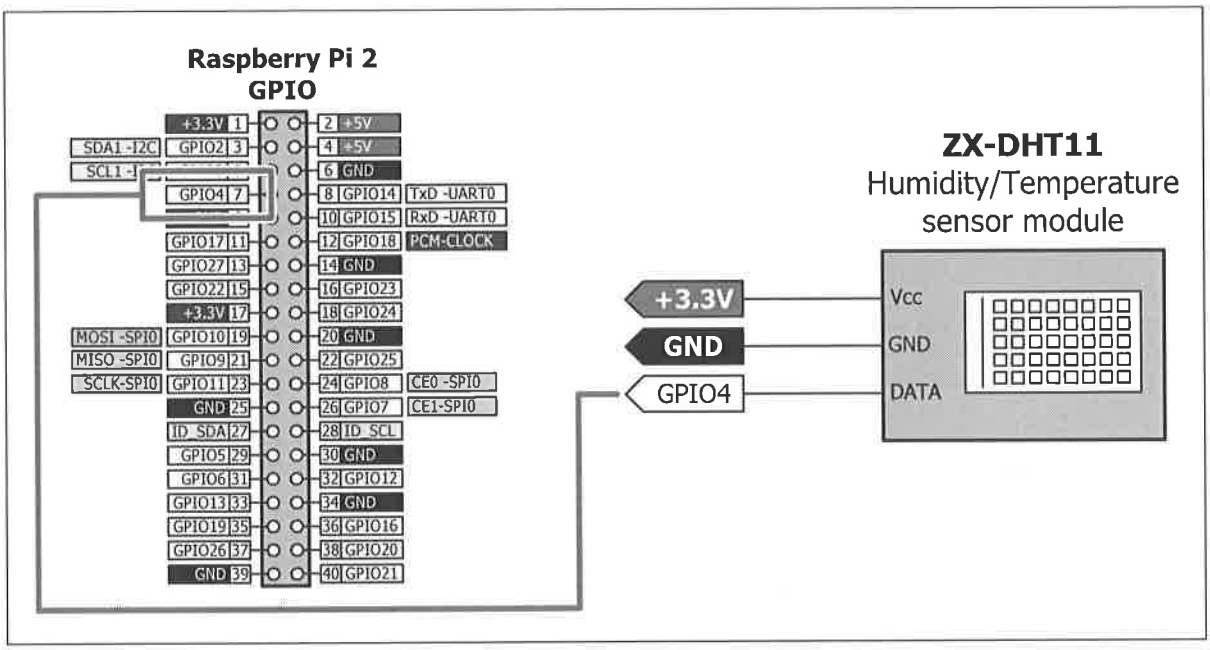
(1) ต่อวงจรตามรูปที่ 10-8 เพื่อทดสอบการทำงาน

(2) สร้างไฟล์สคริปต์ของ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) เขียนคำสั่งตามโปรแกรมที่ 10-2 บันทึกไฟล์เป็น WeatherMonitor.py

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

เมื่อ Raspberry Pi 2 ติดต่อกับโมดูล DHT11 ได้ก็จะอ่านความชื้นสัมพัทธ์และอุณหภูมิที่ตัวตรวจจับวัดได้ นำมาแสดงผลที่หน้าต่างเทอร์มินอล ดังแสดงในรูปที่ 10-9



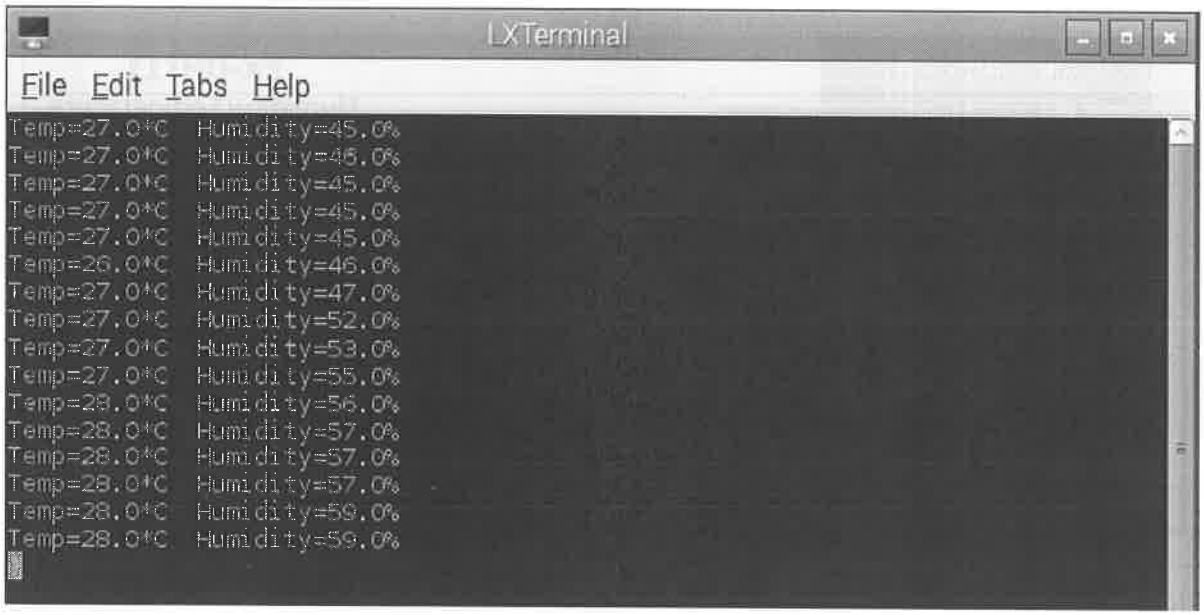
รูปที่ 10-8 วงจรการเชื่อมต่อบอร์ด Raspberry Pi 2 กับบอร์ด ZX-DHT11 เพื่ออ่านค่าความชื้นสัมพัทธ์และอุณหภูมิจากโมดูล DHT11 มาแสดงที่หน้าต่างเทอร์มินอล

```
import time
import Adafruit_DHT

Sensor = Adafruit_DHT.DHT11
GPIO = 4

while True:
  humidity, temperature = Adafruit_DHT.read_retry(Sensor,GPIO)

  if humidity is not None and temperature is not None:
    print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature,
humidity))
  else:
    print('Failed to get reading. Try again!')
```

คำอธิบายโปรแกรมเพิ่มเติม

ในโปรแกรมนี้ ที่บรรทัด `print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))` เป็นการกำหนดให้ Rapberry Pi 2 ทำการแสดงค่าของอุณหภูมิและความชื้นสัมพัทธ์ที่อ่านได้จาก DHT11 บนบรรทัดเดียวที่หน้าต่างเทอร์มินอล โดยการพิมพ์คำสั่งต้องพิมพ์ต่อเนื่อง ไม่ต้องขึ้นบรรทัดใหม่ (ในหน้านี้ต้องขึ้นบรรทัดใหม่เนื่องจากข้อจำกัดด้านพื้นที่ของหน้ากระดาษ)

โปรแกรมที่ 10-2 โค้ดภาษา Python ไฟล์ WeatherMonitor.py สำหรับบอร์ด Raspberry Pi 2 อ่านค่าความชื้นสัมพันธ์และอุณหภูมิจากโมดูลตรวจจับ DHT11 โดยใช้ขาพอร์ต GPIO4 นำมาค่ามาแสดงผลที่หน้าต่างเทอร์มินอล



รูปที่ 10-9 ผลการทำงานของโปรแกรมที่ 10-2 หน้าต่างเทอร์มินอลของ Raspberry Pi 2 แสดงค่าความชื้นสัมพันธ์และอุณหภูมิที่ได้จากโมดูลตรวจจับ DHT11

10.3 BH1750 มินิบอร์ดวัดความเข้มแสงติดต่อผ่านบัส I^2C

เป็นแผงวงจรขนาดเล็กที่ติดตั้งตัวตรวจจับแสงเบอร์ BH1750 โดย BH1750 เป็นผลงานของ ROHM Semiconductor (www.rohm.com) ผู้ผลิตอุปกรณ์สารกึ่งตัวนำชั้นนำของโลก BH1750 นับเป็นตัวตรวจจับแสงที่มีประสิทธิภาพสูง ใช้งานง่าย ด้วยการติดต่อผ่านบัส 2 สายหรือ I^2C ให้ผลการวัดความเข้มแสงเป็นหน่วยลักซ์ (Lux) ทำให้นำข้อมูลที่ได้ไปใช้ประโยชน์ต่อได้ทันที โดยไม่ต้องพึ่งกระบวนการทางคณิตศาสตร์เพื่อแปลงหน่วย ภายในตัวตรวจจับมีวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลความละเอียด 16 บิตทำให้ได้ข้อมูลดิจิตอลของความเข้มแสงที่มีความละเอียดและแม่นยำมากพอสำหรับการนำไปสร้างเครื่องวัดความเข้มแสงหรือลักซ์มอเตอร์ (Luxmeter)

10.3.1 คุณสมบัติทางเทคนิคของบอร์ดวัดความเข้มแสง BH1750

คุณสมบัติทางเทคนิคที่ควรทราบของบอร์ดวัดความเข้มแสง BH1750 มีดังนี้

● ติดตั้งตัวตรวจจับแสงเบอร์ BH1750 บนบอร์ด ภายในมีตัวรับแสงเป็นโฟโต้ไดโอดต่อร่วมกับวงจรขยายสัญญาณ, วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล และวงจรเชื่อมต่อระบบบัส I2C

● มีตัวต้านทานต่อพูลอัปที่ขาเชื่อมต่อบัส I^2C ไว้พร้อม ทำให้เมื่อนำไปเชื่อมต่อกับไมโครคอนโทรลเลอร์หรือ Raspberry Pi ทำได้ทันที โดยไม่ต้องต่อตัวต้านทานเพิ่ม

● ใช้ไฟเลี้ยง +3 ถึง +5Vdc กินกระแสไฟฟ้าต่ำมาก ประมาณ 200μA เท่านั้น

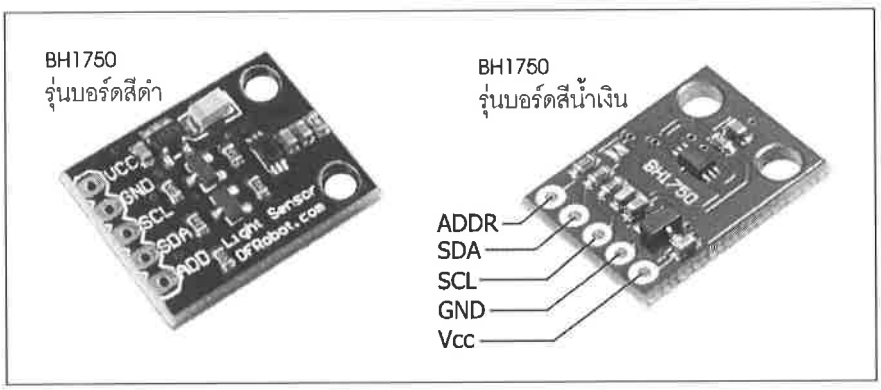
● ย่านวัดความเข้มแสง 1 ถึง 65,535 ลักซ์ มีค่าความผิดพลาด 20%

● กำหนดแอดเดรสให้กับ BH1750 ได้ 2 รูปแบบผ่านทางขา ADDR

● ทนต่อการรบกวนจากแสงอินฟราเรด

● ขนาด 21 x 16 มม.

รูปที่ 10-10 แสดงหน้าตาและการจัดขาของบอร์ดวัดความเข้มแสงที่ใช้ตัวตรวจจับเบอร์ BH1750 จะเห็นได้ว่า มีการผลิตออกมา 2 แบบ มีการจัดขาสัญญาณสลับกันเล็กน้อย ดังนั้น เมื่อนำมาต่อใช้งานควรตรวจสอบตำแหน่งขาให้ถูกต้องก่อน



รูปที่ 10-10 หน้าตาและการจัดขาของบอร์ดวัดความเข้มแสง BH1750 มีการผลิตออกมาจำหน่าย 2 รูปแบบหลัก

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

10.3.2 การเขียนโปรแกรมภาษา Python เพื่อติดต่อกับอุปกรณ์บัส I^2C

เนื่องจาก BH1750 ใช้การติดต่อผ่านบัส I^2C เมื่อนำมาเชื่อมต่อกับบอร์ด Raspberry Pi 2 และต้องการพัฒนาโปรแกรมควบคุมด้วยภาษา Python โดยเฉพาะอย่างยิ่งกับ Python3 จึงต้องเตรียมการเกี่ยวกับไลบรารีให้พร้อมเสียก่อน ดังนี้

10.3.2.1 ติดตั้งไลบรารี python-smbus

โดยเชื่อมต่อบอร์ด Raspberry Pi 2 เข้ากับเครือข่ายอินเทอร์เน็ต จากนั้นเข้าสู่หน้าต่างเทอร์มินอล ทำการดาวน์โหลดไฟล์ไลบรารีด้วยการพิมพ์คำสั่ง

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

เมื่อติดตั้งเสร็จแล้ว ให้ทำการรีบูตด้วยคำสั่ง

```
sudo reboot
```

10.3.2.2 แก้ไขโมดูล smbus ให้ใช้กับ Python3 ได้

เนื่องจากโมดูล smbus ในขณะที่ทำหนังสือเล่มนี้ยังไม่สามารถใช้ร่วมกับ Python3 ได้ทันที จะต้องมีการแก้ไข ดังมีขั้นตอนต่อไปนี้

```
(1) cd ~
(2) sudo apt-get -y install python3-dev
(3) wget http://ftp.de.debian.org/debian/pool/main/i/
    i2c-tools/i2c-tools_3.1.0.orig.tar.bz2
```

```
(4) tar xf i2c-tools_3.1.0.orig.tar.bz2
(5) cd i2c-tools-3.1.0/py-smbus
(6) mv smbusmodule.c smbusmodule.c.orig
(7) wget -O smbusmodule.c http://piborg.org/downloads/
    picoborgrev/smbusmodule.c.txt
(8) wget http://lm-sensors.org/svn/lm-sensors/tags/V2-10-
    8/kernel/include/i2c-dev.h
(9) python3 setup.py build
(10) sudo python3 setup.py install
```

เมื่อเสร็จสิ้นทุกขั้นตอนแล้ว ให้ทำการรีบูตระบบ

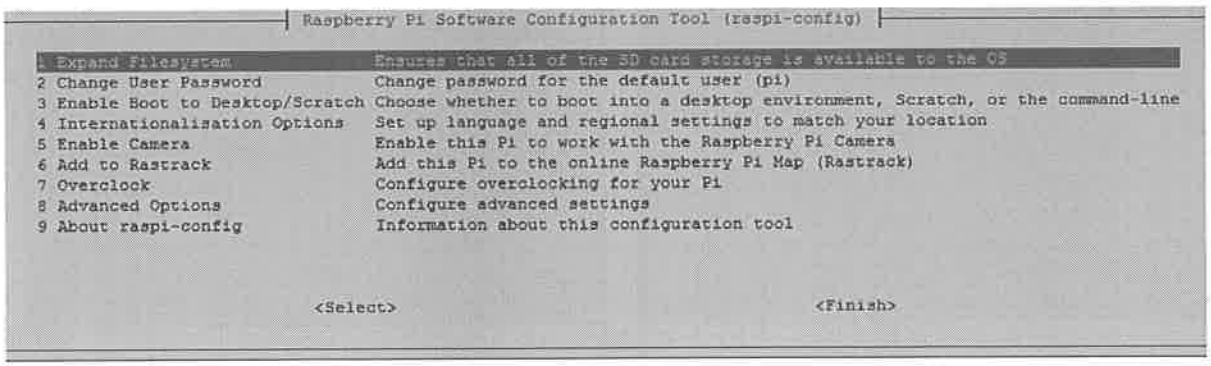
อย่างไรก็ตาม สำหรับผู้ที่จัดซื้อเรียนรู้ Raspberry Pi 2 จาก INEX โปรแกรมและไลบรารีที่เกี่ยวข้องกับการทดลองในหนังสือเล่มนี้ที่บรรจุใน microSD การ์ดได้รับการปรับปรุงให้ใช้งานได้ทั้งหมดแล้ว จึงไม่จำเป็นต้องดำเนินการตามที่อธิบายในหัวข้อ 10.3.2.1 และ 10.3.2.2

10.3.2.3 เปิดความสามารถของการติดต่อกับบัส I²C แก่บอร์ด Raspberery Pi 2

(1) ทำการเปิดความสามารถในการติดต่อบัส I²C ของ Raspberry Pi 2 ด้วยการเข้าสู่หน้าต่างเทอร์มินอล เปิดหน้าต่างตั้งค่าการทำงานของ Raspberry Pi ด้วยโดยพิมพ์คำสั่ง

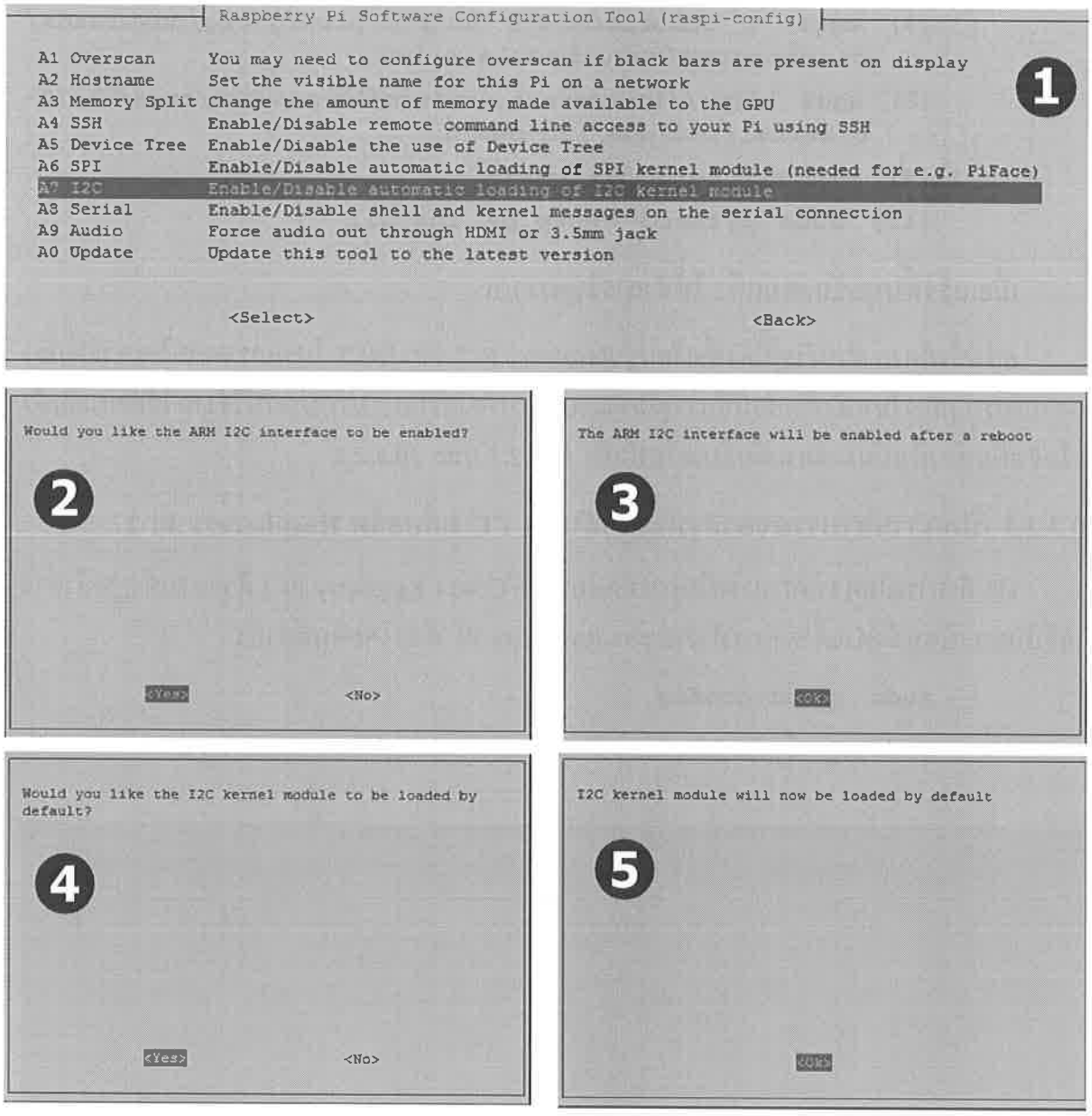
```
sudo raspi-config
```

จะปรากฏหน้าต่าง Raspberry Pi Software Configuration Tools ดังรูป



เลือกข้อ 8 **Advance Option**

จะเข้าสู่หน้าต่างตั้งค่าการทำงานขั้นสูง เลือกหัวข้อ A7 เพื่อเปิดความสามารถการเชื่อมต่อกับอุปกรณ์บัส I²C โดยกดปุ่ม **Select** และ **Yes** เพื่อตอบรับ



10.3.3 ตัวอย่างการใช้งานบอร์ด Raspberry Pi 2 กับบอร์ดวัดความเข้มแสง BH1750

10.3.3.1 อ่านค่าความเข้มแสง

(1) ต่อวงจรตามรูปที่ 10-11

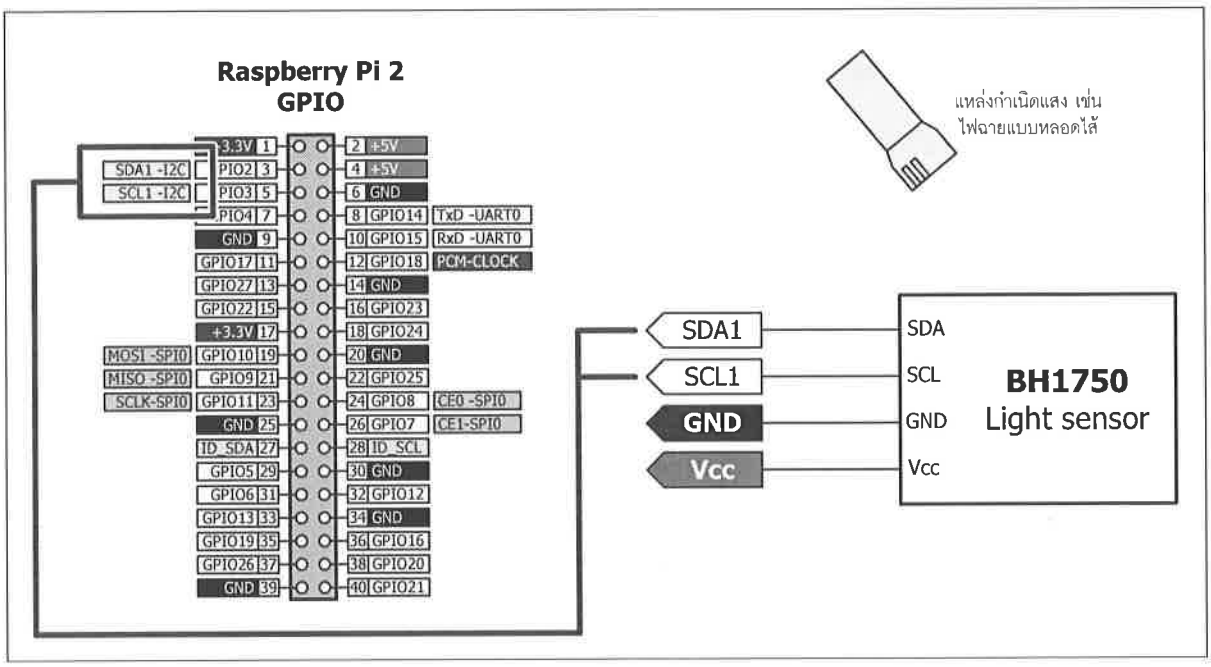
(2) ไปที่หน้าต่างเทอร์มินอล พิมพ์คำสั่งต่อไปนี้ เพื่อตรวจสอบการเชื่อมต่อ

```
sudo i2cdetect -y 1
```

หน้าจอแสดงผลลัพธ์ i2c adress หรือค่าแอดเดรสของตัวอุปกรณ์ที่ต่อกับ Raspberry Pi

```
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: -- -- -- -- -- -- -- --
pi@raspberrypi ~ $
```

(3) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์



รูปที่ 10-11 วงจรการเชื่อมต่อบอร์ด Raspberry Pi 2 กับบอร์ดวัดความเข้มแสง BH1750 เพื่ออ่านค่านำมาแสดงที่หน้าต่างเทอร์มินอล

```
import time
import smbus
bus = smbus.SMBus(1)

addr = 0x23 # i2c adress

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

  print ("Luminosity " ,lum,"lx")
  time.sleep(0.5)
```

โปรแกรมที่ 10-3 โค้ดภาษา Python ไฟล์ SimpleLightMeter.py สำหรับบอร์ด Raspberry Pi 2 อ่านค่าความเข้มแสงจากบอร์ดวัดความเข้มแสง BH1750 มาแสดงผลที่หน้าต่างเทอร์มินอล

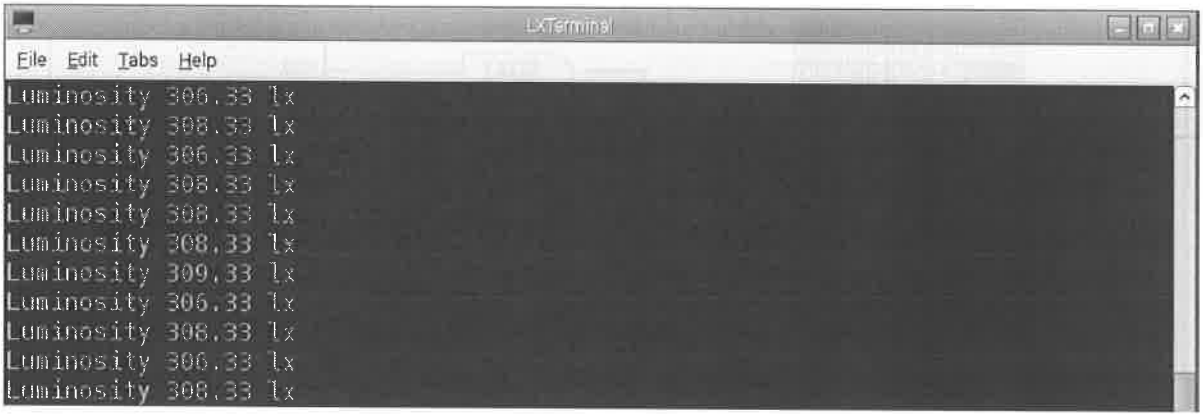
(4) พิมพ์คำสั่งตามโปรแกรมที่ 10-3 โดยระบุแอดเดรสของอุปกรณ์จากที่ตรวจสอบได้ในขั้นตอนที่ (2) ลงในบรรทัดคำสั่ง `addr = xxxx # i2c adress` ในที่นี้ได้ค่าแอดเดรสคือ **0x23**

(5) บันทึกไฟล์ชื่อ SimpleLightMeter.py

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

(7) นำแหล่งกำเนิดแสงมาส่องไปยังตัวตรวจจับ แล้วค่อยๆ ปรับลดและเพิ่มความเข้มแสงหรือใช้มือบังแผงวงจรตรวจจับ สังเกตการทำงานของวงจรผ่านทางหน้าต่างเทอร์มินอล

เมื่อรันโปรแกรม หน้าต่างเทอร์มินอลปรากฏขึ้นมาแสดงค่าความเข้มแสงที่ตรวจวัดได้ (Luminosity) ดังรูปที่ 10-12



รูปที่ 10-12 ผลการทำงานของโปรแกรมที่ 10-3 หน้าต่างเทอร์มินอลของ Raspberry Pi 2 แสดงค่าความเข้มแสงที่วัดได้จากบอร์ดวัดความเข้มแสง BH1750

10.3.3.2 ระบบแจ้งเตือนเมื่อความเข้มแสงต่ำ

ตัวอย่างนี้เป็นการประยุกต์ใช้งานตัวตรวจจับแสงกับบอร์ด Raspberry Pi 2 เพื่อสร้างระบบแจ้งเตือนอัตโนมัติ เมื่อระดับความเข้มของแสงที่ทำการตรวจสอบลดต่ำลงกว่าค่าที่กำหนด

(1) ต่อวงจรตามรูปที่ 10-13

(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

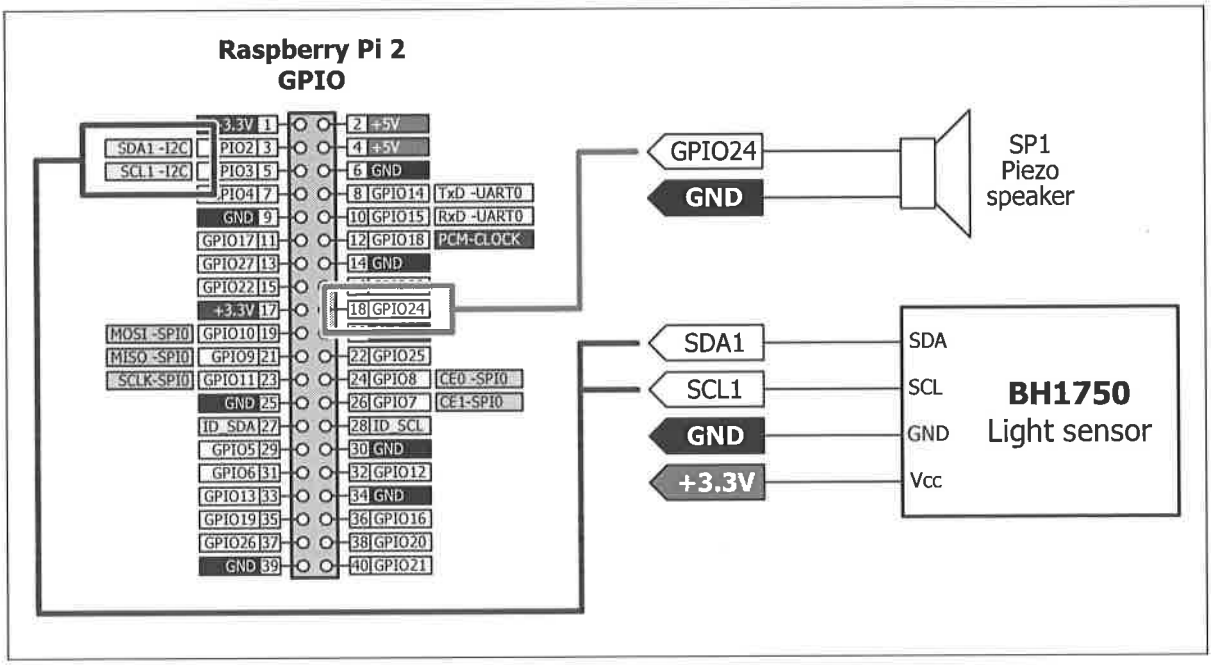
(3) พิมพ์คำสั่งตามโปรแกรมที่ 10-4

(4) บันทึกไฟล์ชื่อ LightAlarm.py

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

(6) หาแหล่งกำเนิดแสงมาส่องไปยังตัวตรวจจับ แล้วค่อยๆ ลดความเข้มลง หรือใช้มือบังแผงวงจรตรวจจับ สังเกตการทำงานของวงจร

เมื่อตัวตรวจจับ BH1750 ตรวจจับความเข้มแสงได้ค่าน้อย (แสงหรี่ลงหรือถูกบังแสง) จะได้ยินเสียงสัญญาณเตือนดังขึ้นจากลำโพงเปียโซ ค่าอ้างอิงที่ใช้ในการตัดสินใจคือ 90 ลักซ์



รูปที่ 10-13 วงจรการเชื่อมต่อบอร์ด Raspberry Pi 2 กับบอร์ดวัดความเข้มแสง BH1750 เพื่ออ่านค่ามาเปรียบเทียบกับค่าที่กำหนด เพื่อสร้างระบบแจ้งเตือนเมื่อความเข้มแสงลดลง

```
import RPi.GPIO as GPIO
import time
import smbus
import datetime
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
blink = GPIO.PWM(18,500)
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 10-4 โค้ดภาษา Python ไฟล์ LightAlarm.py สำหรับบอร์ด Raspberry Pi 2 อ่านค่าความเข้มแสงจากบอร์ดวัดความเข้มแสง BH1750 ผ่านบัส I^2C เพื่อเปรียบเทียบกับค่าที่กำหนดแล้วขับสัญญาณเสียงออกทางลำโพงเมื่อค่าความเข้มแสงต่ำกว่าค่าอ้างอิง

10.3.3.3 LightLogger บันทึกข้อมูลความเข้มแสงลงไฟล์

ตัวอย่างนี้เป็นการนำตัวอย่างในหัวข้อ 10.3.3.1 มาต่อยอด จากการแสดงค่าความเข้มแสงบ่านทางหน้าต่างเทอร์มินอลเท่านั้น มาเพิ่มเติมให้ระบบทำการบันทึกข้อมูลของความเข้มแสงที่ระบบตรวจวัดได้ เก็บในรูปของแฟ้มข้อมูลแบบตัวอักษรหรือเท็กซ์ไฟล์ (text file) ที่มีนามสกุล .txt

(1) ใช้วงจรในรูปที่ 10-11 ในการทดสอบ

```
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 Fileและชื่อ 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 10-5 โค้ดภาษา Python ไฟล์ LightLogger.py สำหรับบอร์ด Raspberry Pi 2 อ่านค่าความเข้มแสงจากบอร์ดวัดความเข้มแสง BH1750 ผ่านบัส I2C แล้วบันทึกเป็นแฟ้มข้อมูลลงในไฟล์ BH1750.txt

(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) พิมพ์คำสั่งตามโปรแกรมที่ 10-5

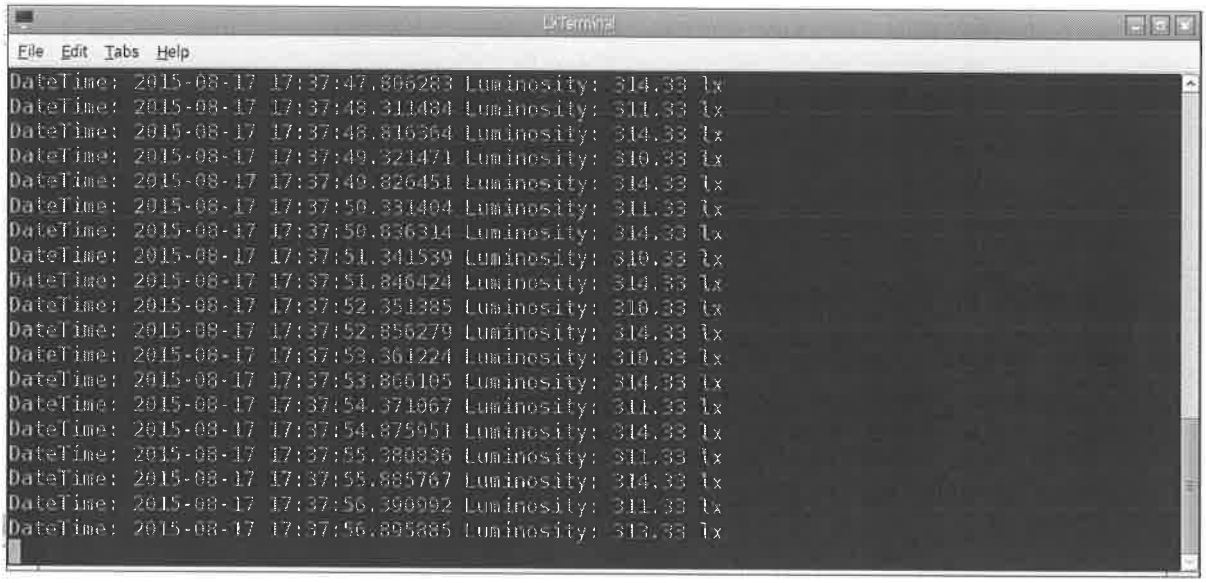
(4) บันทึกไฟล์ชื่อ LightLogger.py

(5) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build > Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **`sudo python3 LightLogger.py`**

(6) หาแหล่งกำเนิดแสงมาส่องไปยังตัวตรวจจับ ปรับความสว่างในระดับต่างๆ กัน ดูผลการทำงาสนผ่านหน้าต่างเทอร์มินอล

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

รูปแบบการบันทึกข้อมูลแสดงในรูปที่ 10-14



รูปที่ 10-14 ผลการทำงานของโปรแกรมที่ 10-5 ข้อมูลที่ได้จากการตรวจวัดความเข้มแสงของ BH1750 จะรวมกับข้อมูลของค่าวันเวลา จะถูกบันทึกลงไปในไฟล์ BH1750.txt ซึ่งเป็นแฟ้มข้อมูลตัวอักษรหรือ text file

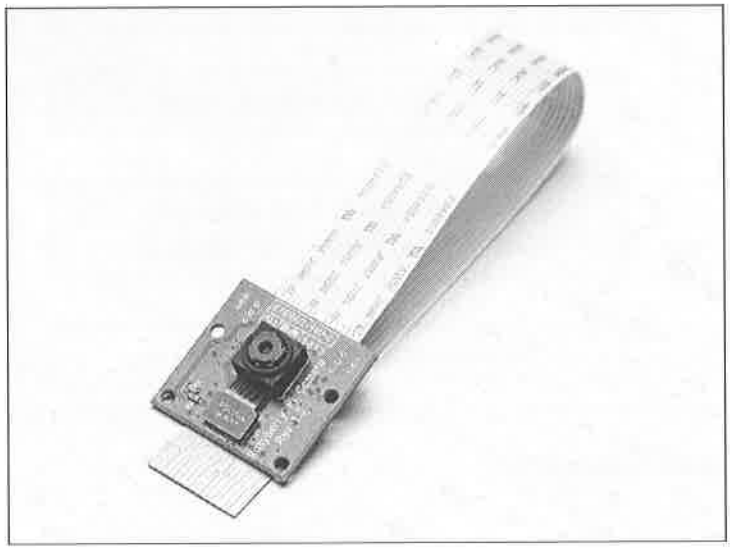
บทที่ 11
Raspberry Pi กับโมดูลกล้อง

อีกหนึ่งอุปกรณ์ต่อพ่วงที่ส่งเสริมให้บอร์ดคอมพิวเตอร์ Raspberry Pi มีศักยภาพเพิ่มขึ้น นั่นคือ **โมดูลกล้อง Raspberry Pi** ที่ได้รับการออกแบบมาให้กับบอร์ดคอมพิวเตอร์ Raspberry Pi โดยเฉพาะ ทั้งนี้เนื่องจากโดยปกติแล้วการหากล้องหรือเว็บแคมสำหรับคอมพิวเตอร์ที่ติดตั้งระบบปฏิบัติการ Linux ค่อนข้างยาก โดยเฉพาะอย่างยิ่งเมื่อต้องใช้งานร่วมกับบอร์ด Raspberry Pi ต้องคำนึงถึงพลังงานไฟฟ้าที่จ่ายให้กับตัวกล้อง รวมถึงการปรับแต่งและขนาดของกล้องก็ต้องเหมาะสมกับบอร์ด Raspberry Pi

สำหรับโมดูลกล้อง Raspberry Pi ที่เป็นรุ่นอย่างเป็นทางการของมูลนิธิ Raspberry Pi ได้รับการผลิตและจำหน่ายโดย Element 14 และ RS โดยออกสู่ตลาดตั้งแต่เดือนพฤษภาคม ค.ศ. 2013 มีหน้าตาแสดงในรูปที่ 10-1 ในบทนี้นำเสนอถึงข้อมูลทางเทคนิค การตั้งค่า และใช้งานโมดูลกล้องกับบอร์ดคอมพิวเตอร์ Raspberry Pi 2

11.1 คุณสมบัติทางเทคนิคที่สำคัญ

โมดูลกล้อง Raspberry Pi ไม่ได้มีชื่อรุ่นที่เป็นเฉพาะเจาะจง ทางมูลนิธิ Raspberry Pi ตั้งชื่อไว้ง่ายๆ เพียงว่า **บอร์ดกล้อง Raspberry Pi (Raspberry Pi Camera board)** ในหนังสือเล่มนี้จึงขอเรียกบอร์ดกล้องนี้ว่า **โมดูลกล้อง** หรือ **โมดูลกล้อง Raspberry Pi** มีคุณสมบัติทางเทคนิคที่น่าสนใจดังนี้



รูปที่ 11-1 หน้าตาของโมดูลกล้อง Raspberry Pi ที่พัฒนาโดยมูลนิธิ Raspberrypi (www.raspberrypi.org)

● มีความละเอียดสูงถึง 5 ล้านพิกเซล

● รองรับการถ่ายภาพขนาดสูงสุด 2592 x 1944 พิกเซล

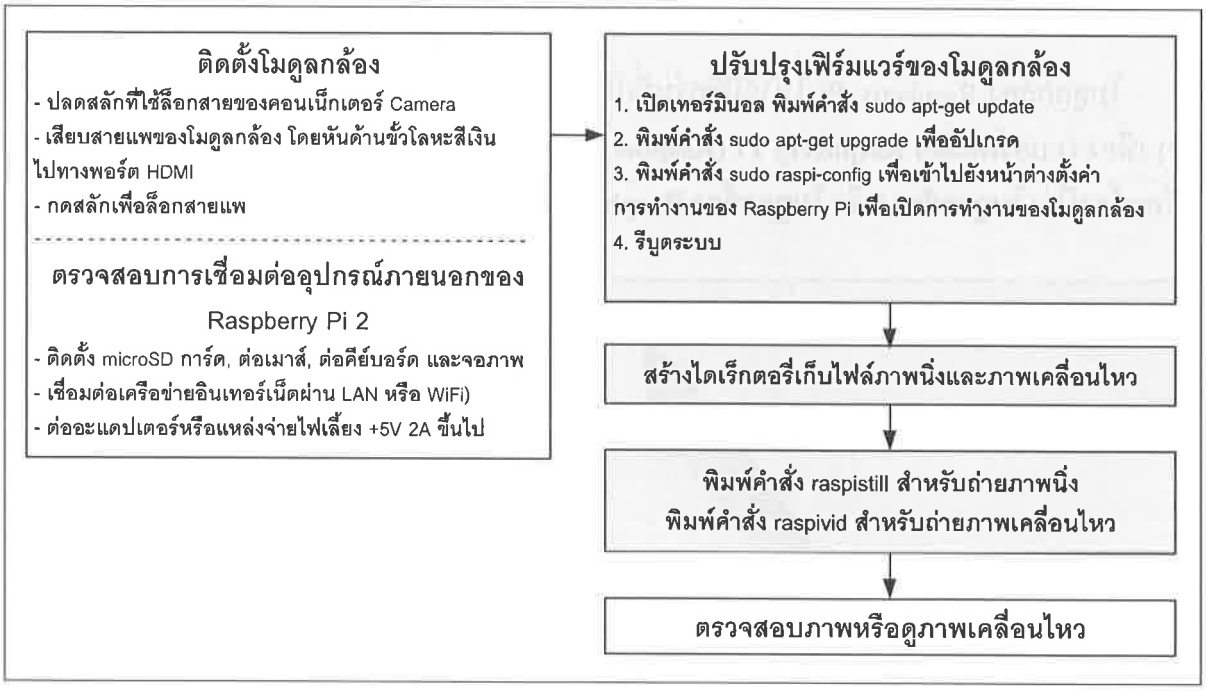
● ถ่ายวิดีโอคุณภาพระดับ HD ความคมชัด 1080p, 720p และ 640x480 ด้วยอัตราแสดงผล 30 (1080p), 60 (720p และ 640x480) และ 90 (640x480) เฟรมต่อวินาที

● ตัวแผงวงจรมีขนาด 25 x 20 x 9 มม. น้ำหนักเพียง 3 กรัม

● เชื่อมต่อกับบอร์ด Raspberry Pi ได้ทุกรุ่นผ่านบัส CSI (Common System Interface) ซึ่งเป็นการเชื่อมต่อแบบ point-to-point บัสนี้พัฒนาโดย Inteloออกแบบมาเพื่อการรับส่งข้อมูลความเร็วสูง 12 ถึง 16 GB/s ด้วยการใช้เทคนิค low-voltage differential signaling ใช้กระแสไฟฟ้าต่ำเหมาะกับพวกอุปกรณ์กล้องที่ต้องถ่ายข้อมูลจำนวนมากอย่างรวดเร็ว

● ใช้กับบอร์ด Raspberry Pi ที่ติดตั้งอิมเมจเวอร์ชั่นล่าสุด

การติดตั้งและเตรียมการเพื่อใช้งานโมดูลกล้องกับบอร์ด Raspberry Pi 2 ได้ทำการสรุปเป็นไดอะแกรมไว้ตามรูปที่ 11-2



รูปที่ 11-2 แผนผังการตั้งค่าและใช้งานโมดูลกล้อง Raspberry Pi

11.2 เตรียมการใช้งานโมดูลกล้อง

11.2.1 ติดตั้งตัวยึด

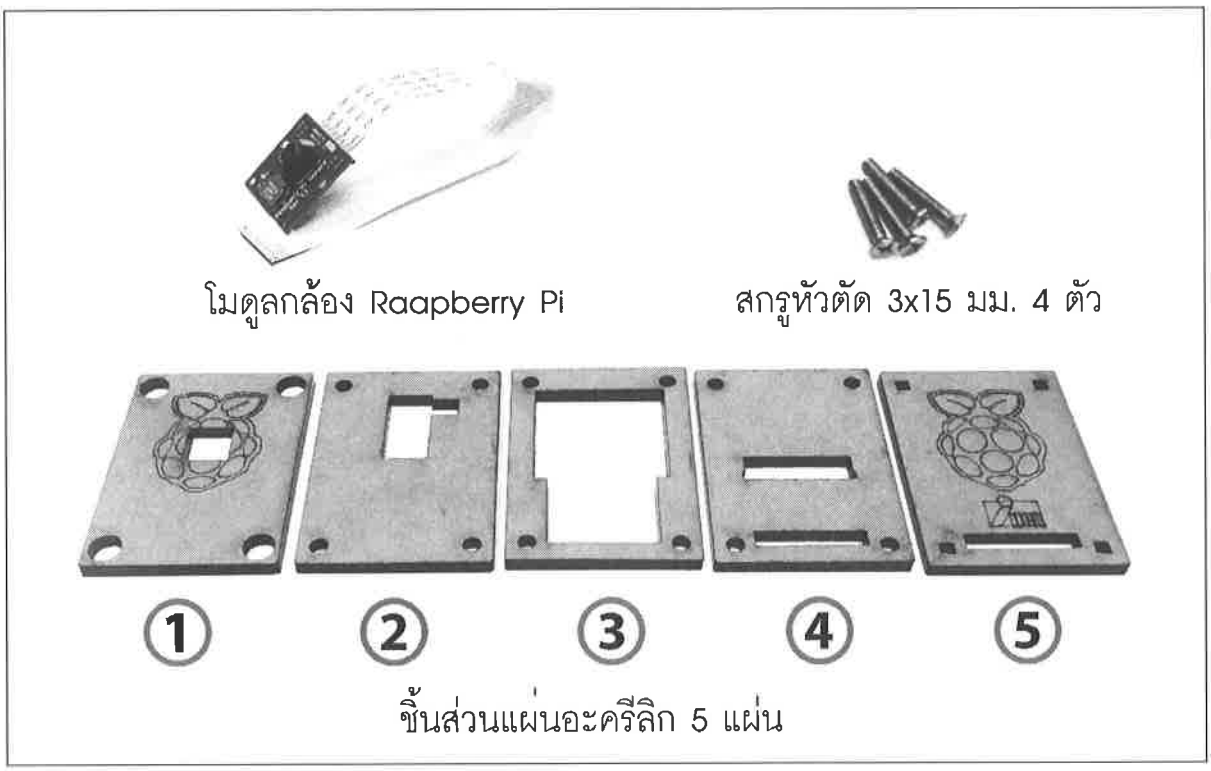
เนื่องจากโมดูลกล้องเป็นแผงวงจรที่ไม่มีอะไรปกป้อง หากผู้ใช้งานมีงบประมาณเพียงพอ ขอแนะนำให้ใช้ประกอบตัวยึดโมดูลกล้องหรือ **Pi Camera Case** เพื่อใช้เป็นทั้งกล่องบรรจุและตัวยึดโมดูลกล้อง ทำให้การใช้งานสะดวกขึ้น และลดโอกาสที่โมดูลกล้องจะเสียหายจากการปะทะหรือการกระแทก

Pi Camra Case เป็นชุดชิ้นส่วนที่ผลิตโดย บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด (INEX) สอบถามได้ที่ www.inex.co.th หากจัดซื้อชุดเรียนรู้ Raspberry Pi 2 จาก INEX จะมีชุดตัวยึดโมดูลกล้องนี้มาให้ด้วย

11.2.1.1 ส่วนประกอบของชุด Pi Camera Case

ในชุดประกอบด้วยชิ้นส่วนดังนี้

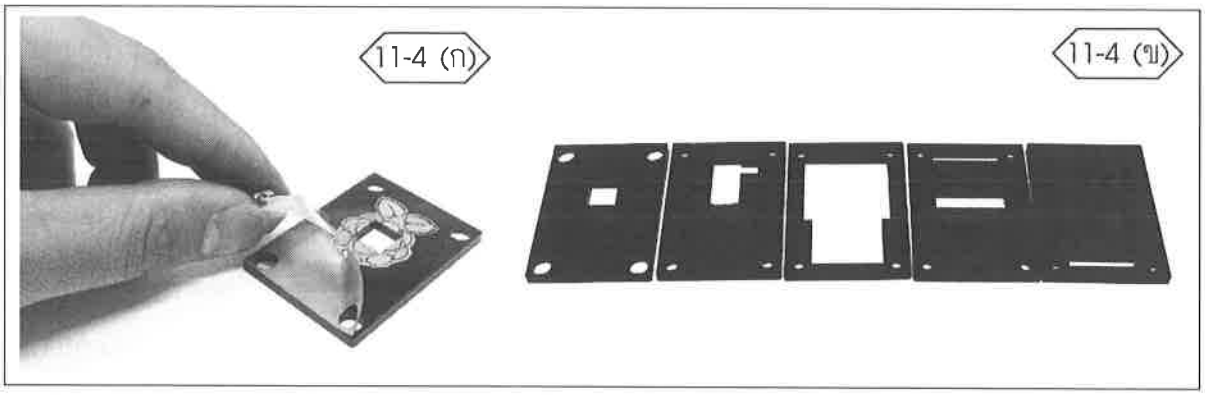
1. กรอบแผ่นอะครีลิก จำนวน 5 แผ่น
2. สกรูหัวตัดขนาด 3x15 มม. จำนวน 4 ตัว



รูปที่ 11-3 แสดงชิ้นส่วนในชุด Pi Camera Case

11.2.1.2 ขั้นตอนการประกอบ

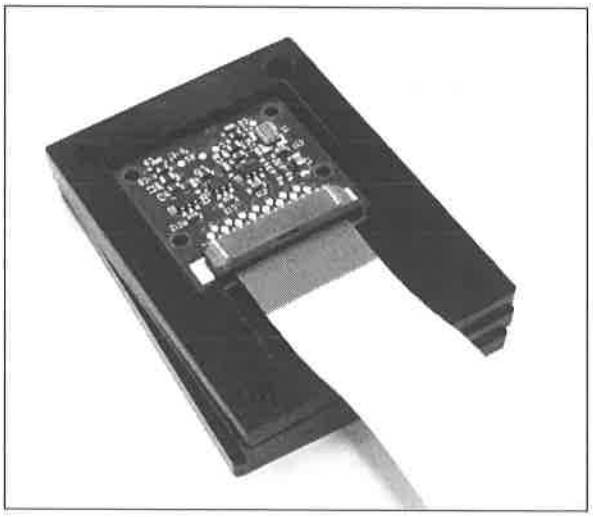
(1) ลอกกระดาษบนแผ่นอะครีลิกออกทั้งสองด้านทุกแผ่น ดังรูปที่ 11-4



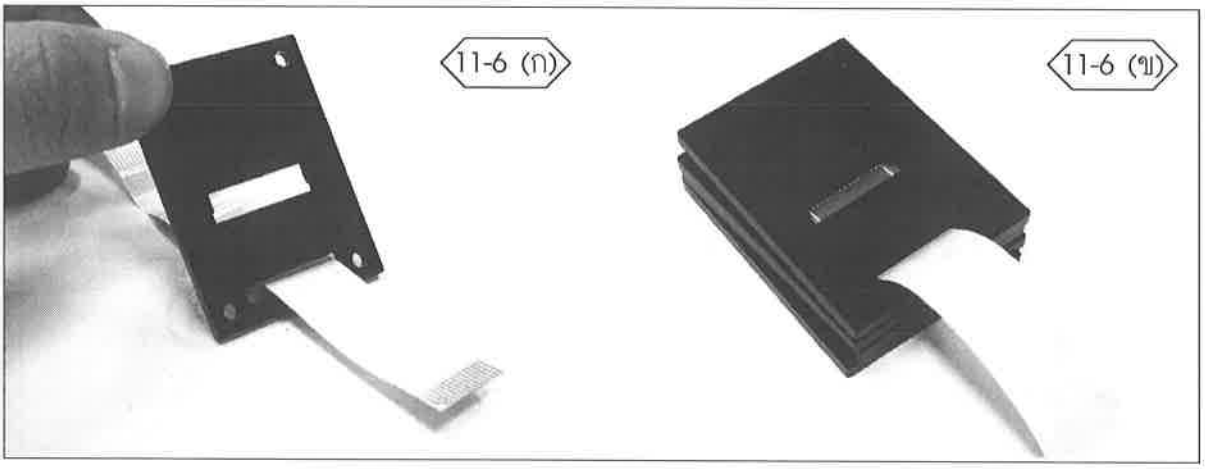
รูปที่ 11-4 ลอกกระดาษปิดแผ่นอะครีลิกออกทั้งหมด

(2) นำแผ่นอะครีลิกชิ้นที่ 1 คว่ำหน้า (โลโก้ Raspberry Pi อยู่ด้านล่าง) ตามด้วยคว่ำแผ่นที่ 2 และ 3 จากนั้นนำโมดูลกล้องวางคว่ำหน้าตามลงไปดังรูปที่ 11-5 โดยให้สายแพอยู่ด้านบนของแผ่นอะครีลิกแผ่นที่ 3

(3) สอดสายแพผ่านช่องด้านล่างของแผ่นอะครีลิกชิ้นที่ 4 แล้ววางไว้ด้านบนของแผ่นที่ 3 ดังรูปที่ 11-6

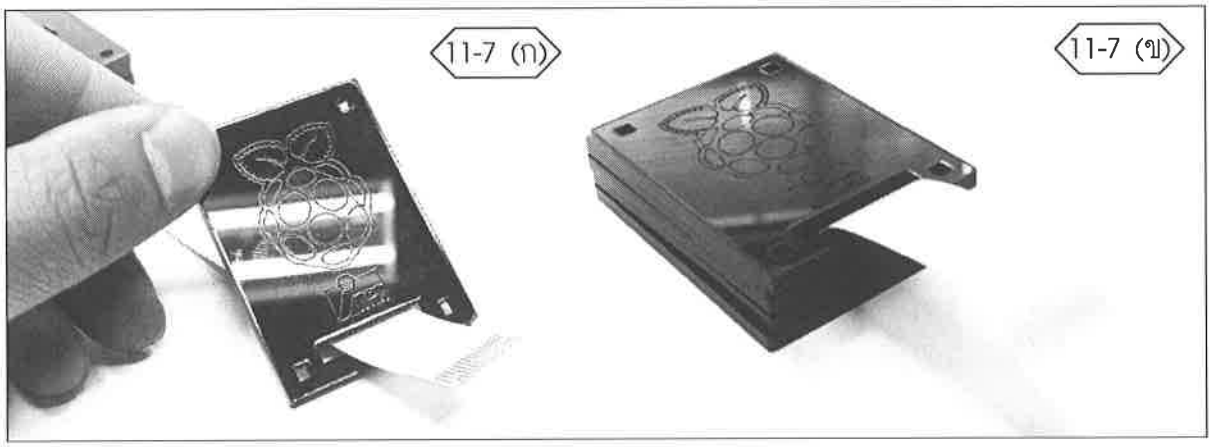


รูปที่ 11-5 ประกอบกล่อง 3 ชั้นแรก



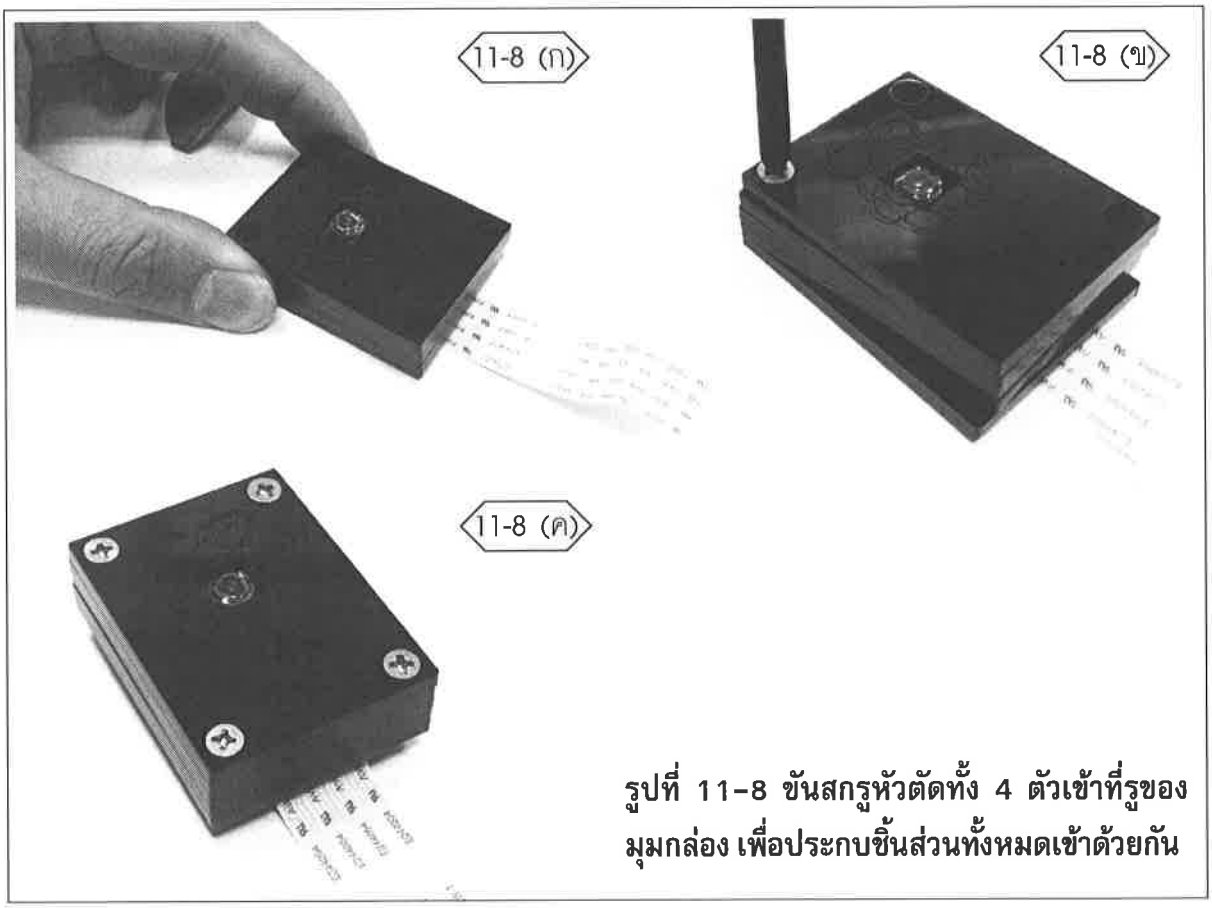
รูปที่ 11-6 ประกอบกล่องชั้นที่ 4

(4) สอดสายแพผ่านช่องด้านล่างของแผ่นอะครีลิกชิ้นที่ 5 โดยหันโลโก้ออกด้านหลังดังรูปที่ 11-7 (ก) จากนั้นจัดให้ชิ้นส่วนทุกแผ่นอยู่ในระนาบเดียวกันดังรูปที่ 11-7 (ข)



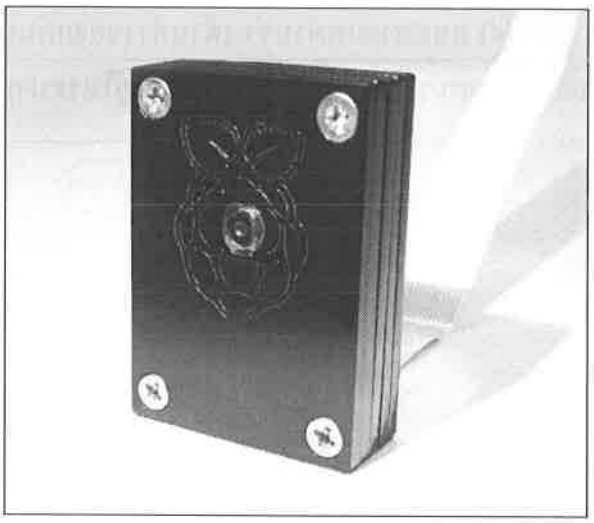
รูปที่ 11-7 ประกอบกล่องชั้นที่ 5

(5) หงายชุดแผ่นอะครีลิกที่ประกอบกับตัวโมดูลกล้องขึ้น แล้วขันสกรูหัวตัด 3x15 มม. ทั้ง 4 มุม ดังรูปที่ 11-8 โดยไม่ต้องใช้นอต ทำการขันสกรูอัดให้ทุกชิ้นส่วนประกบเข้าด้วยกัน



รูปที่ 11-8 ขันสกรูหัวตัดทั้ง 4 ตัวเข้าที่รูของมุมกล่อง เพื่อประกบชิ้นส่วนทั้งหมดเข้าด้วยกัน

(6) อาจคลายสกรูเล็กน้อยเพื่อจัดให้ชิ้นส่วนพลาสติกที่ประกอบกันเป็นกล่องห่อหุ้มโมดูลกล้อง Raspberry Pi อยู่ในระนาบเดียวกัน ตรงกัน และดูเรียบร้อย แล้วขันสกรูให้แน่นอีกครั้ง เป็นอันเสร็จสิ้นการประกอบชุด Pi Camera Case กับโมดูลกล้อง



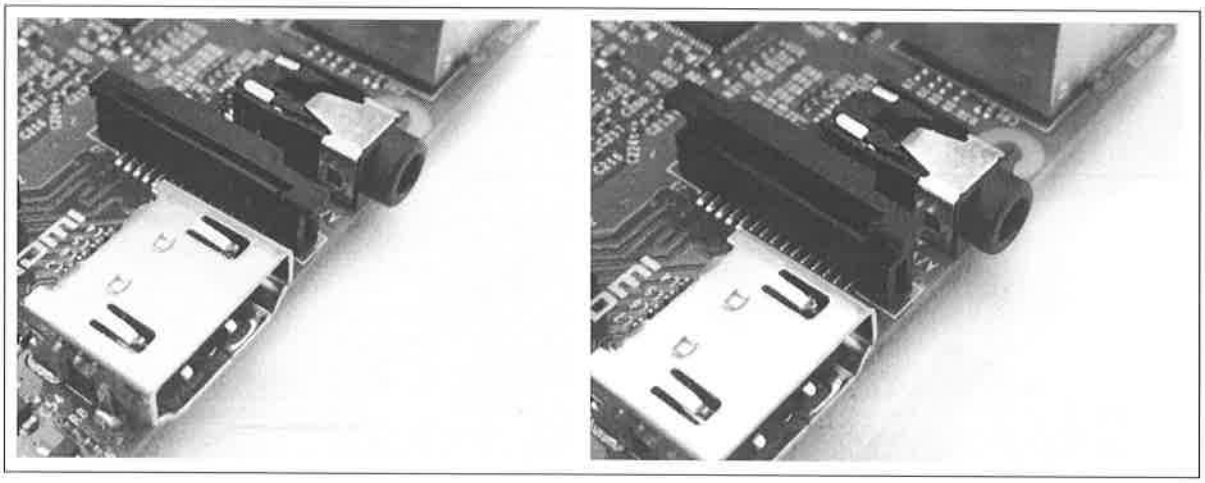
รูปที่ 11-9 กล่องยึดโมดูลกล้องที่ประกอบเสร็จแล้ว

หมายเหตุ *รูปโลโก้ Raspberry Pi ที่สลักบนกล่องยึดโมดูลกล้องเป็นตัวบ่งบอกถึงทิศทางการถ่ายภาพ กล่าวคือ ถ้าโลโก้กลับหัวก็จะได้ภาพที่กลับหัวเช่นกัน*

11.2.2 เชื่อมต่อโมดูลกล้องกับบอร์ด Raspberry Pi 2

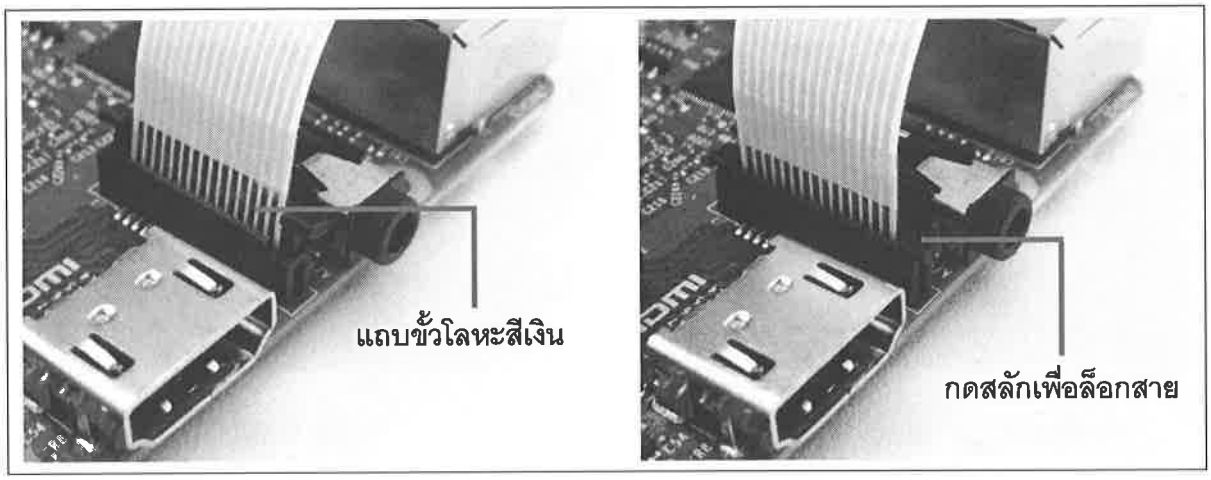
การเชื่อมต่อโมดูลกล้องเข้ากับบอร์ด Raspberry Pi 2 จะต้องดำเนินการอย่างระมัดระวัง เนื่องจากสายสัญญาณของโมดูลกล้องค่อนข้างแบบบาง ขั้นตอนการเชื่อมต่อมีดังนี้

(1) ที่บอร์ด Raspberry Pi 2 ระหว่างพอร์ต HDMI กับจุดต่อ LAN จะมีคอนเน็กเตอร์สีดำอยู่ ดังรูปที่ 11-10 (ซ้าย) ให้ดึงสลักขึ้นตามรูปที่ 11-10 (ขวา) ต้องระมัดระวังในการดึงสลักขึ้น



รูปที่ 11-10 การปลดสลักที่ใช้ในการล็อกสายของคอนเน็กเตอร์สำหรับเชื่อมต่อโมดูลกล้อง

(2) ที่โมดูลกล้องมีสายแพต่ออยู่ ด้านหน้าจะมีแถบสีน้ำเงิน ด้านหลังจะมีแถบขั้วโลหะสีเงิน ให้หันด้านขั้วโลหะสีเงินเข้าหาพอร์ตHDMI ดังรูปที่ 11-11 (ซ้าย) แล้วสอดสายลงไปในช่อง กดสลักลงไปเพื่อล็อกสาย ดังรูปที่ 11-11 (ขวา) ขณะนี้โมดูลกล้องเชื่อมต่อกับบอร์ดคอมพิวเตอร์ Raspberry Pi 2 เรียบร้อย พร้อมใช้งาน

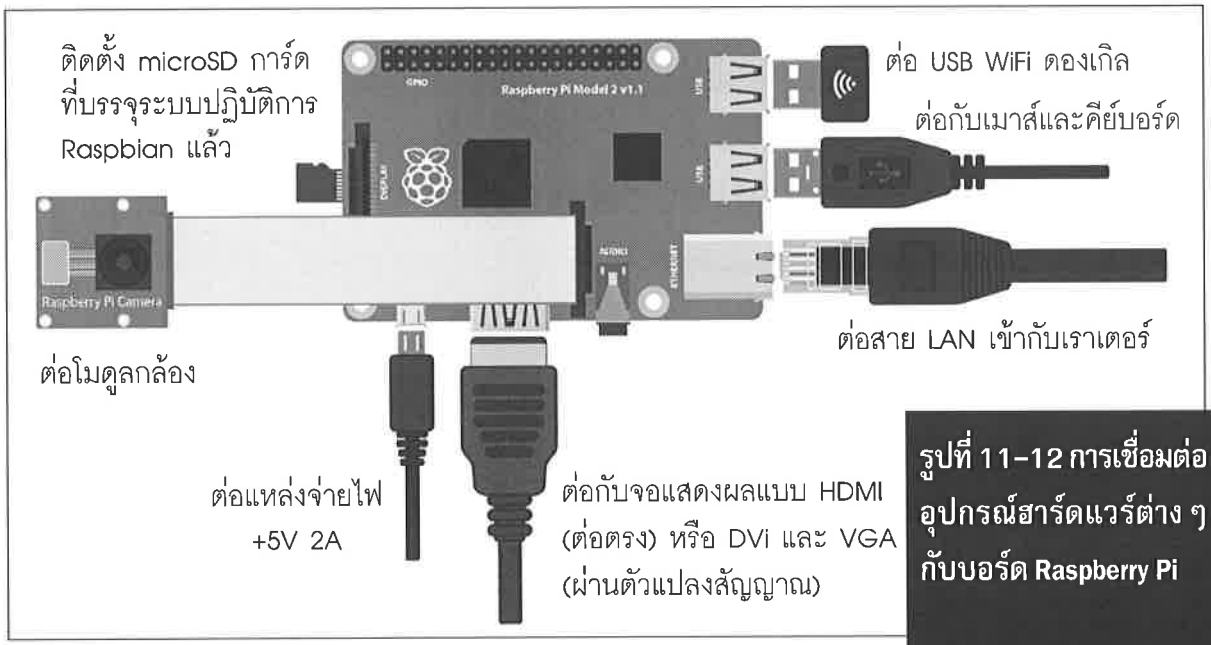


รูปที่ 11-11 การเสียบสายแพของโมดูลกล้องกับคอนเน็กเตอร์สำหรับเชื่อมต่อโมดูลกล้อง

11.2.3 การตั้งค่าเฟิร์มแวร์ของโมดูลกล้อง

มีขั้นตอนดังนี้

(1) ตรวจสอบการเชื่อมต่อบอร์ด Raspberry Pi กับอุปกรณ์ต่างๆ ทั้งสายไฟเลี้ยง, สาย HDMI , คีย์บอร์ด, เมาส์, สาย LAN หรือ Wi-Fi ดองเกิล, โมดูลกล้อง และ microSD การ์ดที่ติดตั้งระบบปฏิบัติการ Raspbiang รุ่นล่าสุด ดังรูปที่ 11-12



รูปที่ 11-12 การเชื่อมต่ออุปกรณ์ฮาร์ดแวร์ต่าง ๆ กับบอร์ด Raspberry Pi

(2) จ่ายไฟเลี้ยง ล็อกอินเข้าระบบ และเชื่อมต่อบอร์ด Raspberry Pi เข้ากับเครือข่ายอินเทอร์เน็ตโดยผ่านจุดต่อ LAN หรือใช้ Wi-Fi ดองเกิล

(3) จากนั้นเปิดเทอร์มินอล พิมพ์คำสั่ง **sudo apt-get update** แล้วรอการโหลดข้อมูลจนเสร็จ

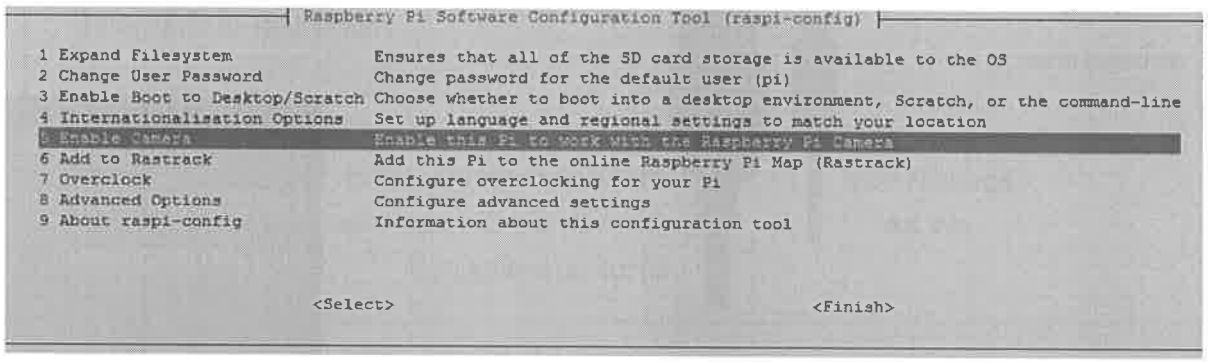
```
pi@raspberrypi ~ $ sudo apt-get update
Get:1 http://raspberrypi.collabora.com wheezy Release.gpg [836 B]
Get:2 http://mirrordirector.raspbian.org wheezy Release.gpg [490 B]
Get:3 http://raspberrypi.collabora.com wheezy Release [7,514 B]
Get:4 http://archive.raspberrypi.org wheezy Release.gpg [490 B]
Get:5 http://mirrordirector.raspbian.org wheezy Release [14.4 kB]
Get:6 http://raspberrypi.collabora.com wheezy/rpi armhf Packages [2,214 B]
Get:7 http://archive.raspberrypi.org wheezy Release [10.2 kB]
Get:8 http://mirrordirector.raspbian.org wheezy/main armhf Packages [6,904 kB]
16% [8 Packages 1,076 kB/6,904 kB 16%] [Waiting for headers] [Waiting for headers]
```

(4) พิมพ์คำสั่ง **sudo apt-get upgrade** แล้วกด **Y** เพื่อยืนยัน ตามด้วยกด **Enter** เพื่อเริ่มการอัปเกรด

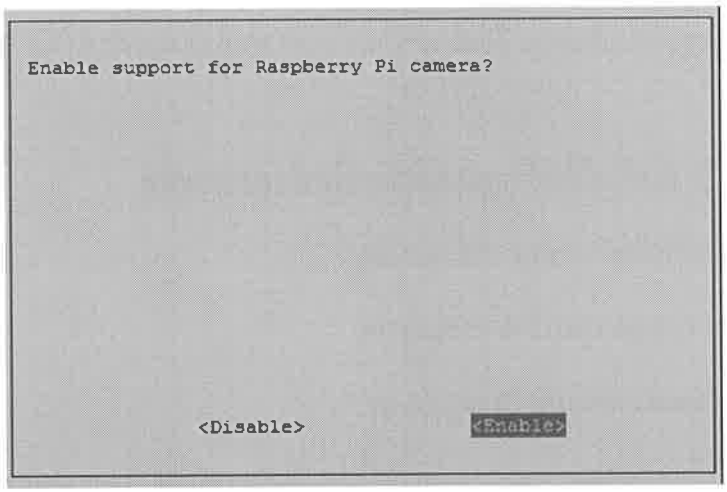
```
pi@raspberrypi ~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  cups-bsd cups-client cups-common file gnupg gpgv libarchive12 libavcodec53 libavutil51
  libc-bin libc-dev-bin libc6 libc6-dev libcups2 libcupsimage2 libgcrypt11 libgd2-xpm
  libgnutls26 libicu48 libldap-2.4-2 libmagic1 libnss3 libraspberrypi-bin
  libraspberrypi-dev libraspberrypi-doc libraspberrypi0 libruby1.9.1 libsmbclient
  libssh2-1 libssl1.0.0 libtag1-vanilla libtagc0a libtasn1-3 libwbclient0 libx11-6
  libx11-data libx11-xcb1 libxfont1 libxml2 locales multiarch-support ntp openssl
  python-picamera python3-picamera raspberrypi-bootloader ruby1.9.1 samba-common
  smbclient
49 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 104 MB of archives.
After this operation, 274 kB disk space will be freed.
Do you want to continue [Y/n]? y
```

(5) เมื่ออัปเดตและอัปเกรดเฟิร์มแวร์เป็นเวอร์ชั่นปัจจุบันเรียบร้อยแล้ว ให้พิมพ์คำสั่ง **sudo raspi-config** เพื่อเข้าไปเปิดการทำงานของโมดูลกล้อง

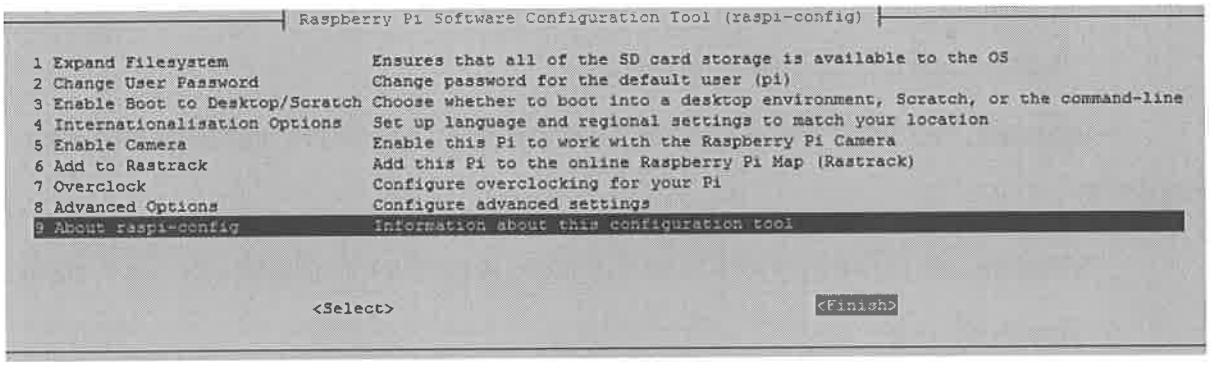
(6) เลือกไปที่หัวข้อ **Enable Camera**



(7) จะเข้าสู่หน้าต่างควบคุมการทำงานของโมดูลกล้อง เลือก **enable**



(8) กลับมาที่หน้าต่าง **raspi-config** เลื่อนไปที่ปุ่ม **Finish** ทำการรีบูตบอร์ด Raspberry Pi 2 หลังจากนี้ก็พร้อมที่จะใช้งานโมดูลกล้องแล้ว



11.3 ชุดคำสั่งควบคุมการทำงานของโมดูลกล้อง

การควบคุมและสั่งงานให้กล้องบันทึกภาพ รวมถึงรายละเอียดการตั้งค่า จะใช้ชุดคำสั่งเพื่อสั่งงาน สรุปได้ดังนี้

11.3.1 raspistill คำสั่งเปิดโมดูลกล้องเพื่อถ่ายภาพนิ่ง

มีพารามิเตอร์ของคำสั่งเพื่อกำหนดการทำงานดังนี้

--width,-w ปรับขนาดความกว้างของรูปภาพ

--height,-h ปรับขนาดความสูงของรูปภาพ

--quality,-q ปรับแต่งคุณภาพของรูปภาพที่ถ่ายเป็นนามสกุล .jpeg โดยปรับแต่งได้แต่ 0 ถึง 100

--output,-o เก็บบันทึกภาพที่กำหนดชื่อของไฟล์ได้

ตัวอย่างที่ 11-1

```
raspistill -o image.jpg
```

บันทึกภาพแล้วเก็บในไฟล์ชื่อ image.jpg

--verbose,-v แสดงรายละเอียดต่างๆ ของการตั้งค่าในโหมดต่างๆ ของโมดูลกล้องขณะเก็บรูปภาพ

--timeout,-t ใช้กำหนดช่วงเวลาเปิดทำงานของโมดูลกล้องก่อนปิด *ปกติตั้งไว้ที่ 5 วินาที*

ตัวอย่างที่ 11-2

```
raspistill -t 10000
```

เป็นการสั่งให้เปิดการทำงานของโมดูลกล้องนาน 10 วินาที

--timelapse,-tl ใช้กำหนดให้โมดูลกล้องถ่ายภาพแบบโหมด Timelapse ช่วงเวลาเป็นมิลลิวินาที

ตัวอย่างที่ 11-3

```
raspistill -t 20000 -tl 2000 -o image%04d.jpg
```

คำสั่งนี้สั่งให้เปิดการทำงานของโมดูลกล้องนาน 20 วินาที โดยมีการถ่ายภาพทุก 2 วินาที แล้วบันทึกภาพเป็นชื่อ image0001.jpg ถึง image0011.jpg

--demo,-d เปิดให้โมดูลกล้องทำงานโดยไม่บันทึกภาพ ช่วงเวลาเป็นมิลลิวินาที

ตัวอย่างที่ 11-4

```
raspistill -d 30000
```

เปิดการทำงานของโมดูลกล้องเป็นเวลา 30 วินาที

--encoding,-e บีบอัดไฟล์รูปภาพเป็นนามสกุลอื่นนอกจาก .jpg โดยรองรับนามสกุล .bmp , .png และ .gif

ตัวอย่างที่ 11-5

```
raspistill -o  image.png -e png
```

สั่งให้บันทึกภาพเป็นไฟล์ .png

11.3.2 raspivid คำสั่งเปิดโมดูลกล้องเพื่อถ่ายภาพเคลื่อนไหว

มีพารามิเตอร์ของคำสั่งเพื่อกำหนดการทำงานดังนี้

--width,-w ปรับความกว้างของภาพเคลื่อนไหว ควรตั้งค่าอยู่ในช่วง 64 ถึง 1920

--height,-h ปรับความสูงของภาพเคลื่อนไหว ควรตั้งค่าอยู่ในช่วง 64 ถึง 1080

--bitrate,-b ปรับค่าบิตเรตต่อวินาที สำหรับไฟล์ h264 จะตั้งค่าอยู่ที่ 10Mbit/s (-b 10000000) และ 1080p จะตั้งค่าอยู่ที่ 15Mbit/s (-b 15000000)

--output,-o บันทึกภาพเคลื่อนไหวเป็นชื่อไฟล์ได้

ตัวอย่างที่ 11-6

```
raspivid -o movie1.h264
```

สั่งให้บันทึกไฟล์วิดีโอเป็นไฟล์ .h264

--verbose,-v แสดงรายละเอียดการตั้งค่าโหมดต่างๆ ของโมดูลกล้องขณะเก็บภาพเคลื่อนไหว

--timeout,-t กำหนดช่วงเวลาเปิดทำงานของโมดูลกล้อง เมื่อครบเวลาจะปิดการทำงานลง ค่าปกติคือ 5 วินาที หากต้องการปิดการทำงานก่อน กดคีย์ Ctrl ตามด้วย C

--demo,-d เปิดให้โมดูลกล้องทำงานโดยไม่ได้บันทึกภาพเคลื่อนไหว หน่วยเป็นมิลลิวินาที

ตัวอย่างที่ 11-7

```
raspivid -d 20000
```

เปิดการทำงานของโมดูลกล้องเป็นเวลา 20 วินาที

--framerate,-fps กำหนดจำนวนภาพที่เก็บต่อวินาที เพื่อนำมาสร้างเป็นภาพเคลื่อนไหว ตั้งค่าได้ตั้งแต่ 2 ถึง 30 เฟรมต่อวินาที

11.3.3 รหัสคำสั่งสำหรับการจัดการแสดงภาพ (Preview Windows)

รหัสคำสั่งที่นำไปต่อท้ายคำสั่ง `raspistill` เพื่อจัดการเกี่ยวกับการแสดงรูปภาพที่ได้จากโมดูลกล้อง ประกอบด้วย

`--preview,-p` ใช้กำหนดขนาดของหน้าต่างแสดงผลรูปภาพจากกล้อง ต้องใส่ค่าพิกัด x และ y บนจอภาพ, ขนาดความกว้างของภาพ และความสูงของภาพ

ตัวอย่างที่ 11-8

```
raspistill -p 100,150,1000,800
```

แสดงภาพที่พิกัด (100, 150) ด้วยขนาด 1000 x 800

`--fullscreem,-f` กำหนดให้แสดงรูปภาพเต็มจอภาพ เช่น `raspistill -f`

`--nopreview,-n` กำหนดให้ปิดการแสดงรูปภาพบนจอภาพ เช่น `raspistill -n`

`--opacity,-op` กำหนดค่าความเข้มของรูปภาพที่แสดงบนจอภาพโดยมีค่าในช่วง 0 ถึง 255

ตัวอย่างที่ 11-9

```
raspistill -op 0
```

จะไม่เห็นการแสดงภาพ

```
raspistill -op 150
```

จะเห็นภาพลางๆ

```
raspistill -op 255
```

จะเห็นภาพที่แสดงผลชัดเจนที่สุด

11.3.4 รหัสคำสั่งควบคุมกล้อง

รหัสคำสั่งที่นำไปต่อท้ายคำสั่ง `raspistill` และ `raspivid` เพื่อควบคุมการทำงานภายในของโมดูลกล้อง ประกอบด้วย

`--sharpness,-sh` กำหนดความคมชัดของภาพหรือ Sharpness โดยค่าปกติเป็น 0 ปรับแต่งได้ตั้งแต่ -100 ถึง 100 ตัวอย่างคำสั่งคือ `raspistill -sh 10`

`--contrast,-co` ปรับแต่งความแตกต่างของส่วนที่สว่างที่สุดกับส่วนที่มืดที่สุดของภาพหรือ contrast ค่าปกติเป็น 0 ปรับแต่งได้ตั้งแต่ -100 ถึง 100 ตัวอย่างคำสั่งคือ `raspistill -co 20`

`--brightness,-br` ใช้กำหนดความสว่างของภาพหรือ Brightness ค่าปกติเป็น 50 ปรับแต่งได้ตั้งแต่ 0 ถึง 100 โดยถ้าใส่ค่าเป็น 0 ภาพจะมืดสนิท และ ถ้าใส่ค่าเป็น 100 ภาพจะสว่างเป็นสีขาว ตัวอย่างคำสั่งคือ `raspistill -br 60`

--saturation,-sa กำหนดความอิ่มตัวของสีหรือ Saturation ค่าปกติเป็น 0 ปรับแต่งได้ตั้งแต่ -100 ถึง 100 ตัวอย่างคำสั่งคือ `raspistill -sa 50`

--ISO,-ISO กำหนดค่าความไวแสง ใช้ปรับเวลาถ่ายภาพในที่แสงน้อย โดยปรับแต่งได้ตั้งแต่ 100 ถึง 800 ตัวอย่างคำสั่งคือ `raspistill -ISO 200`

--vstab,-vs เปิดระบบกันสั่นเมื่อใช้โมดูลกล้องถ่ายภาพเคลื่อนไหว ตัวอย่างคำสั่งคือ `raspivid -vs`

--ev,-ev กำหนดค่าความสว่างของแสงที่เปิดเข้ามายังตัวรับภาพ (EV Compensation) ค่าปกติเป็น 0 ปรับแต่งได้ตั้งแต่ -10 ถึง +10 ตัวอย่างคำสั่งคือ `raspistill -ev +3`

--exposure,-ex เปิดใช้งานระบบถ่ายภาพหรือ Exposure mode แบ่งเป็นโหมดย่อยดังนี้

auto : เปิดโหมด Automatic exposure

night : เปิดโหมดถ่ายภาพกลางคืนหรือ Night shooting

backlight : เปิดโหมด Back-light

sport : เปิดโหมด Sport (ชัตเตอร์เปิดปิดเร็ว)

snow : เปิดโหมดสำหรับถ่ายภูมิประเทศที่เป็นหิมะ

beach : เปิดโหมดสำหรับถ่ายภูมิประเทศที่เป็นชายหาด

verylong : เปิดโหมด Long exposures

fixedfps : เปิดโหมดจำกัดจำนวนภาพที่ถ่ายต่อ 1 วินาที

antishake : เปิดโหมดป้องกันภาพเบลอจากการสั่นสะเทือน

fireworks : เปิดโหมดสำหรับถ่ายภาพดอกไม้ไฟ

--awb,-awb เปิดใช้งานระบบปรับเปลี่ยนสีภายใต้สภาวะแสงต่างๆ หรือ Auto white balance (AWB) แบ่งเป็นโหมดย่อยดังนี้

auto : เปิดโหมด Automatic

sun : เปิดโหมด Sunny

cloudshade : เปิดโหมด Cloudy

tungsten : เปิดโหมด Tungsten lighting

fluorescent : เปิดโหมด Fluorescent lighting

incandescent : เปิดโหมด Incandescent lighting

flash : เปิดโหมด Flash

horizon : เปิดโหมด Horizon

--imxfx,-ifx เปิดใช้งานถ่ายภาพแบบเอฟเฟ็กต์ (Image effect) แบ่งเป็นโหมดย่อยดังนี้

negative : เปิดเอฟเฟ็กต์แบบ Negative image

solarise : เปิดเอฟเฟ็กต์แบบ Solarise

sketch : เปิดเอฟเฟ็กต์แบบ Sketch-style

denoise : เปิดเอฟเฟ็กต์แบบ Denoise image

emboss : เปิดเอฟเฟ็กต์แบบ Embossed

oilpaint : เปิดเอฟเฟ็กต์แบบ Oil paint-style

hatch : เปิดเอฟเฟ็กต์แบบ Cross-hatch sketch

gpen : เปิดเอฟเฟ็กต์แบบ Graphite sketch

pastel : เปิดเอฟเฟ็กต์แบบ Pastel

watercolour : เปิดเอฟเฟ็กต์แบบ Watercolour

film : เปิดเอฟเฟ็กแบบ Grainy film

blur : เปิดเอฟเฟ็คแบบ Blur image

colourswap : เปิดเอฟเฟ็กต์แบบ Colourswap

washedout : เปิดเอฟเฟ็กต์แบบ Washedout

posterise : เปิดเอฟเฟ็กต์แบบ Posterise

cartoon : เปิดเอฟเฟ็กต์แบบ Cartoon

--rotation , -rot เปิดใช้งานโมดูลกล้อง แล้วหมุนมุมมองภาพที่เก็บบันทึก รองรับมุมหมุนที่ 0, 90, 180 และ 270 องศา ตัวอย่างคำสั่งคือ `raspistill -rot 180`

--hflip , -hf เปิดใช้งานโมดูลกล้อง แล้วบันทึกภาพแบบกลับกระจกในแนวนอน

--vflip , -vf เปิดใช้งานโมดูลกล้อง แล้วบันทึกภาพแบบกลับกระจกในแนวตั้ง

11.4 ทดสอบถ่ายภาพนิ่ง

หลังจากเชื่อมต่อโมดูลกล้องกับบอร์ด Raspberry Pi 2 และทำการอัปเดตโปรแกรมเรียบร้อยแล้ว เปิดเทอร์มินอล พิมพ์คำสั่ง `raspistill -f` ถ้าไม่มีอะไรผิดพลาด โมดูลกล้องจะทำงานและแสดงภาพที่ได้จากกล้องที่หน้าจอภาพ แต่ถ้าหากเชื่อมต่อสายผิด จะปรากฏข้อความแจ้งดังรูปที่ 11-11 ให้ตรวจสอบการเชื่อมต่อของโมดูลกล้องกับบอร์ด Raspberry Pi อีกครั้ง

```
pi@raspberrypi ~ $ raspistill -f
mmal: mmal_vc_component_enable: failed to enable component: ENOSPC
mmal: camera component couldn't be enabled
mmal: main: Failed to create camera component
mmal: Failed to run camera app. Please check for firmware updates

pi@raspberrypi ~ $
```

รูปที่ 11-11 หน้าจอแจ้งการเชื่อมต่อสัมเหลวระหว่างโมดูลกล้องกับบอร์ด Raspberry Pi 2

(1) เปิดเทอร์มินอล พิมพ์คำสั่ง **`mkdir MyPicture`** (การตั้งชื่อห้ามเว้นวรรค) เพื่อสร้างไดเร็กตอรี่สำหรับเก็บภาพ ซึ่งเก็บไว้ใน พาธ **home/pi/MyPicture** จากนั้นพิมพ์คำสั่ง **`cd MyPicture`** เพื่อเข้าไปยังไดเร็กตอรี่ที่สร้างไว้

```
pi@raspberrypi ~ $ mkdir MyPicture
pi@raspberrypi ~ $ cd MyPicture
pi@raspberrypi ~/MyPicture $
```

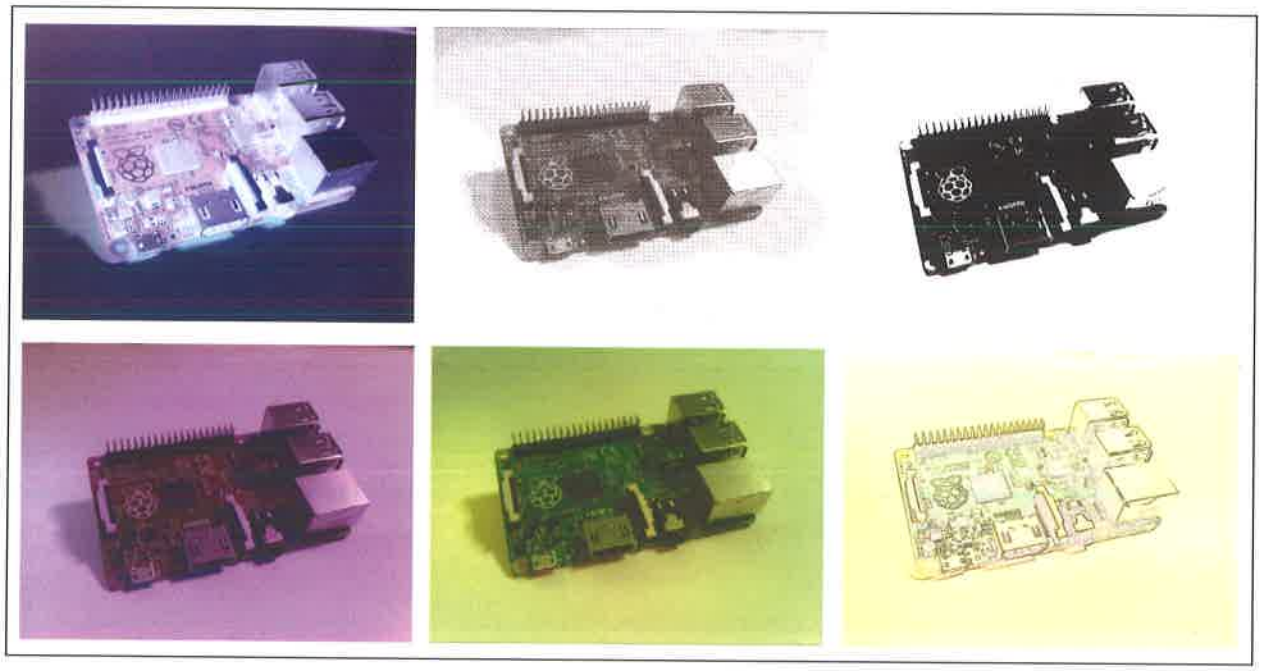
(2) จากนั้นหันกล้องไปยังวัตถุที่ต้องการถ่ายภาพ พิมพ์ชุดคำสั่งสำหรับถ่ายภาพ เช่น

`raspistill -t 2000 -o TestImage1.jpg`

สั่งให้เปิดโมดูลกล้อง 2 วินาที เพื่อถ่ายภาพ แล้วบันทึกไฟล์ในชื่อ Test Image1.jpg

จะได้ภาพตัวอย่างตามรูปที่ 11-12

รูปที่ 11-12 ตัวอย่างภาพที่ถ่ายด้วยโมดูลกล้อง Raspberry Pi โดยใช้คำสั่ง `raspistill -t 2000 -o TestImage1.jpg`



รูปที่ 11-13 ภาพถ่ายที่ได้จากการเปิดเอฟเฟ็กต์เพื่อแต่งภาพในแบบต่าง ๆ (ไล่จากซ้ายไปขวา บนลงล่าง)

(a) Negative **(b) Hatch** **(c) gpen**

(d) Colourswap **(e) Oilpaint** **(f) Sketch**

(3) พิมพ์คำสั่งสำหรับถ่ายภาพในแบบอื่นๆ ดังนี้

```
raspistill -t 2000 -o TestImage2.jpg -w 640 -h 480
```

เปิดโมดูลกล้อง 2 วินาที ถ่ายภาพนิ่งขนาด 640 x 480 พิกเซล บันทึกไว้ในไฟล์ชื่อTestImage2.jpg

```
raspistill -t 4000 -o TestImage3.jpg -q 80
```

เปิดโมดูลกล้อง 4 วินาที ถ่ายภาพนิ่ง บันทึกไว้ในไฟล์ชื่อ TestImage3.jpg ด้วยความละเอียด 80

```
raspistill -t 3000 -o TestImage4.jpg -br 40 -ISO 20
```

เปิดโมดูลกล้อง 3 วินาที ปรับความสว่าง 40 ตั้ง ISO 20 ถ่ายภาพนิ่ง บันทึกไว้ในไฟล์ชื่อ TestImage4.jpg

```
raspistill -t 3000 -o TestImage5.jpg -rot 90
```

เปิดโมดูลกล้อง 3 วินาที ถ่ายภาพ แล้วหมุนภาพ 90 องศา บันทึกในไฟล์ชื่อ Test Image5.jpg

```
raspistill -t 2000 -o TestImage6.jpg -ex sport
```

เปิดโมดูลกล้อง 2 วินาที เปิดระบบถ่ายภาพ Exposure mode แบบ Sport ถ่ายภาพนิ่ง บันทึกไว้ในไฟล์ชื่อ TestImage6.jpg

raspistill -t 6000 -o TestImage7.jpg -awb fluorescent

เปิดโมดูลกล้อง 6 วินาที เปิดโหมดปรับแสงแบบ fluorescent ถ่ายภาพนิ่ง บันทึกไว้ในไฟล์ชื่อ TestImage7.jpg

raspistill -t 3000 -o TestImage8.jpg -ifx negative

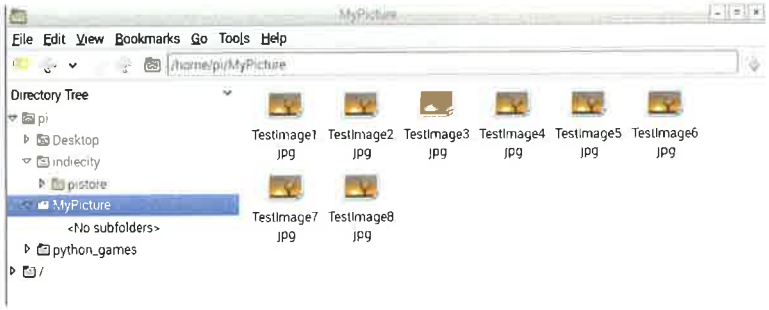
เปิดโมดูลกล้อง 3 วินาที เปิดเอฟเฟ็กต์แบบ Negative image ถ่ายภาพนิ่ง บันทึกไว้ในไฟล์ชื่อ TestImage8.jpg

raspistill -t 120000 -tl 2000 -o ImageT%03d.jpg

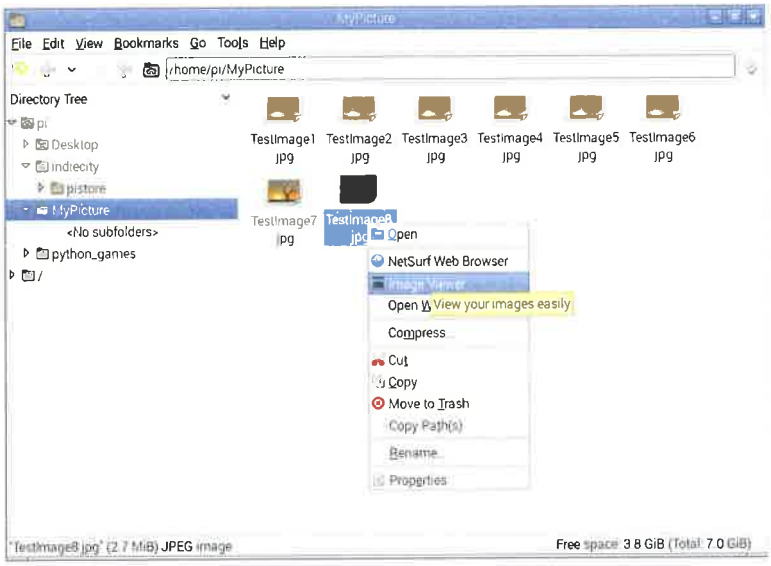
เปิดโมดูลกล้อง 120 วินาที แล้วถ่ายภาพนิ่งทุกๆ 2 วินาที บันทึกไว้ในไฟล์ชื่อ ImageT001.jpg ถึง ImageT061.jpg รวม 60 ภาพ

(4) ทดลองถ่ายภาพแล้วเรียกคำสั่งเพื่อตกแต่งภาพด้วยเอฟเฟ็กต์ต่างๆ ในรูปที่ 11-13 รวบรวมภาพถ่ายที่ผ่านการใช้เอฟเฟ็กต์ต่างๆ ที่มีในโมดูลกล้อง Raspberry Pi

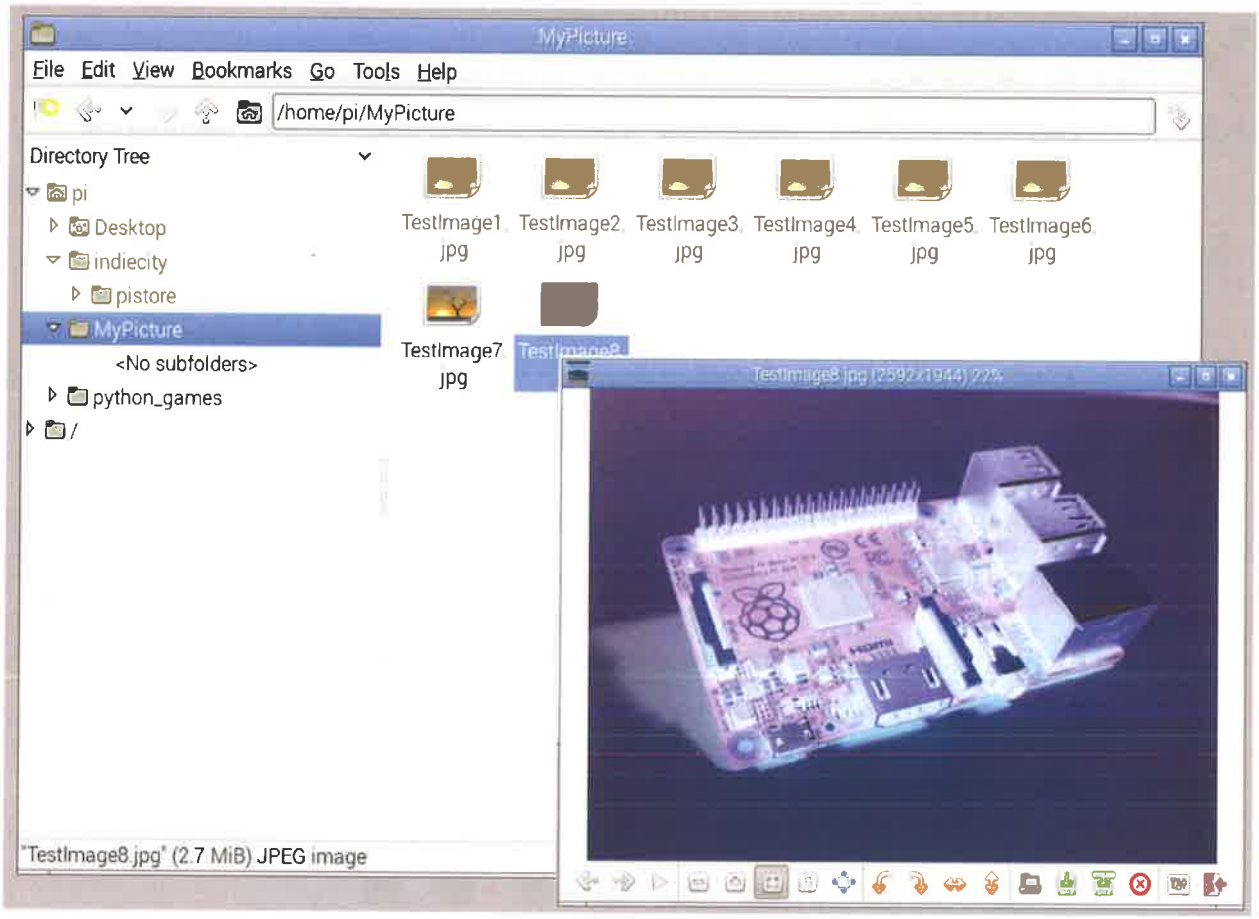
(5) หลังจากถ่ายรูปแล้ว ผู้ใช้งานสามารถตรวจสอบรูปถ่ายได้ โดยเลือกเข้าสู่ระบบปฏิบัติการในโหมดกราฟิก จากนั้นเลือก **Menu > Accessories > File Manager > MyPicture**



(6) จะพบไฟล์ของรูปถ่ายที่บันทึกไว้ คลิกเมาส์ปุ่มขวาที่รูปภาพ แล้วเลือก **Image Viewer**



(7) โปรแกรมจะแสดงรูปภาพขึ้นมา



11.5 ทดสอบถ่ายภาพเคลื่อนไหว

หลังจากทดสอบถ่ายภาพนิ่งแล้ว ลำดับต่อไปเป็นการทดสอบการถ่ายภาพเคลื่อนไหว จะมีขั้นตอนคล้ายๆ กับการถ่ายภาพนิ่ง ดังนี้

(1) เปิดเทอร์มินอล พิมพ์คำสั่ง **mkdir MyVideo** เพื่อสร้างไดเร็กตอรี่สำหรับเก็บภาพเคลื่อนไหว ซึ่งอยู่ในพาธ **home/pi/MyVideo** จากนั้นพิมพ์คำสั่ง **cd MyVideo** เพื่อเข้าไปยังไดเร็กตอรี่ที่สร้างไว้

(2) หันกล้องไปพื้นที่เป้าหมายที่ต้องการบันทึกภาพ พิมพ์ชุดคำสั่งที่ใช้ถ่ายภาพเคลื่อนไหว เช่น

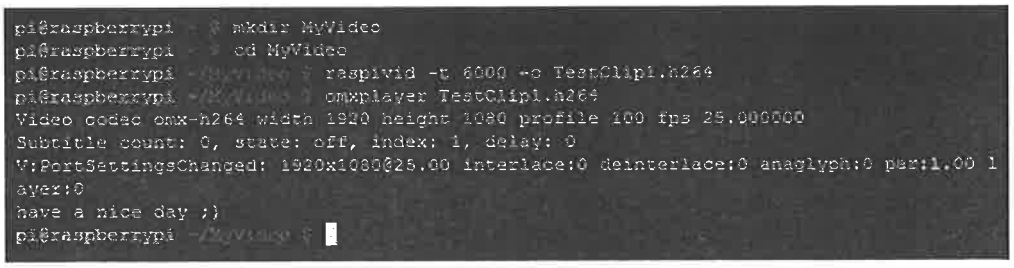
raspivid -t 60000 -o TestClip1.h264

เปิดโมดูลกล้อง 60 วินาที แล้วถ่ายภาพเคลื่อนไหว บันทึกไว้ในไฟล์ชื่อ TestClip1.h264 โดยความละเอียดจะเป็นค่ามาตราฐานอยู่ที่ 1080p 30fps

raspivid -t 60000 -o TestClip2.h264 -b 10000000 -fps 10

เปิดโมดูลกล้อง 60 วินาที ตั้งค่าบิตเรตที่ 10Mbit/s ใช้อัตราการบันทึกภาพ 10 fps แล้วถ่ายภาพเคลื่อนไหว บันทึกไว้ในไฟล์ชื่อ TestClip2.h264

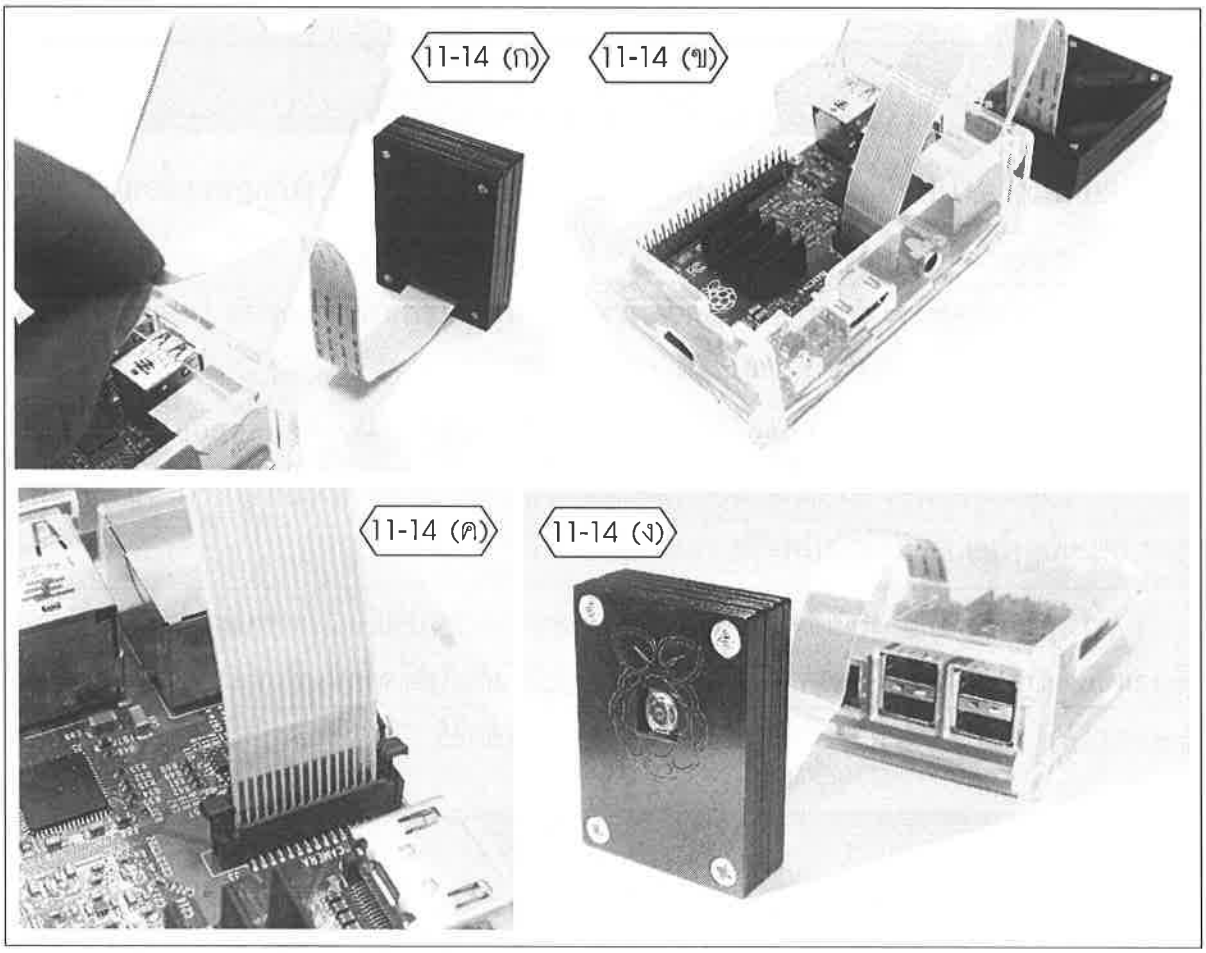
(3) หลังจากบันทึกภาพเคลื่อนไหวแล้ว ผู้ใช้งานสามารถเรียกไฟล์ขึ้นมาแสดงผลผ่านเทอร์มินอลได้ โดยพิมพ์คำสั่ง **cd MyVideo** เพื่อเข้าไปยังไดเร็กตอรี่ที่เก็บไฟล์ จากนั้นพิมพ์คำสั่ง **omxplayer TestClip1.h264** ก็จะเล่นไฟล์ภาพเคลื่อนไหวที่ได้บันทึกไว้



ในกรณีที่ระบบปฏิบัติการ Raspbian ไม่ได้ติดตั้งโปรแกรม **omxplayer** ไว้ ให้เชื่อมต่อบอร์ด Raspberry Pi 2 เข้ากับเครือข่ายอินเทอร์เน็ต แล้วเปิดเทอร์มินอล พิมพ์คำสั่ง **sudo apt-get install omxplayer** เพื่อติดตั้งโปรแกรม

11.6 ตัวอย่างการควบคุมโมดูลกล้องด้วยโปรแกรมภาษา Python

ในทุกขั้นตอนของตัวอย่างการควบคุมและใช้งานโมดูลกล้อง Raspberry Pi ผู้พัฒนาจะต้องเชื่อมต่อโมดูลกล้องและทำการติดต่อในเบื้องต้นให้เรียบร้อยก่อน ดังรูปที่ 11-14 ซึ่งได้บรรจุบอร์ด Raspberry Pi 2 ลงในกล่องอะครีลิกที่ทาง INEX จัดทำขึ้นแล้วรวมไว้ในชุด Raspberry Pi 2 Education kit ทำการเชื่อมต่อสายของโมดูลกล้องเข้ากับบอร์ด Raspberry Pi 2 ผ่านทางจุดต่อ CSI ตามที่แสดงในรูป



รูปที่ 11–14 การเตรียมความพร้อมและเชื่อมต่อโมดูลกล้อง Raspberry Pi เข้ากับบอร์ด Raspberry PI 2 ที่บรรจุลงในกล่องแล้ว

11.6.1 ทดลองถ่ายภาพด้วย Python

ในตัวอย่างนี้เป็นการทดสอบเบื้องต้นของการเขียนโปรแกรมภาษา Python เพื่อควบคุมการทำงานของโมดูลกล้องผ่านบอร์ด Raspberry Pi 2

(1) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(2) พิมพ์คำสั่งตามโปรแกรมที่ 11-1

(3) บันทึกไฟล์ชื่อ CameraControl.py

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

ผลลัพธ์ที่ได้คือ เมื่อทำการ Remote Desktop เข้าไปดู รูปถ่ายของโมดูลกล้องจะอยู่ที่โฟลเดอร์ /home/pi/Desktop/PythonDemo/Photo หรือ /home/pi/Desktop/Photo ขึ้นอยู่กับการกำหนดพาธและที่ตั้งของโฟลเดอร์ โดยในโฟลเดอร์ Photo ถูกสร้างขึ้นจากคำสั่งในโปรแกรมที่ 11-1 จะมีไฟล์อยู่ 2 ไฟล์ ซึ่งชื่อไฟล์จะไม่เหมือนกัน โดยไฟล์หนึ่งใช้ค่าของวันเวลาที่ถ่ายมากำหนดชื่อไฟล์ ส่วนอีกไฟล์ภาพจะใช้ชื่อที่ผู้ใช้งานกำหนดเอง ดังรูปที่ 11-15

```
import picamera
import datetime
img_name="Photo/"
img_name+=str(datetime.datetime.now())
img_name+=".jpg"
camera = picamera.PiCamera()
camera.capture('Photo/IMG.JPG')
camera.capture(img_name)
```

คำอธิบายโปรแกรมเพิ่มเติม

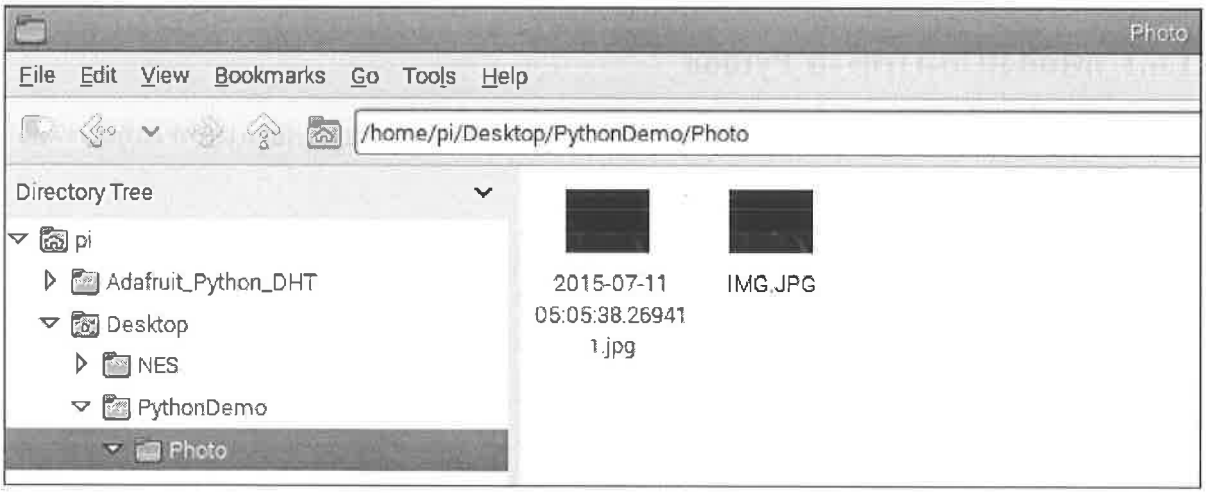
ในโปรแกรมนี้กำหนดให้เก็บไฟล์ภาพถ่ายไว้ที่โฟลเดอร์ Photo มีนามสกุลเป็น .jpg กำหนดชื่อไฟล์ไว้ 2 แบบคือ

1. ใช้ข้อมูลของวันและเวลา จากคำสั่ง `img_name+=str(datetime.datetime.now())`

2. กำหนดชื่อเองโดยผู้พัฒนาโปรแกรม หลังจากที่สั่งถ่ายภาพด้วยคำสั่ง `camera.capture('Photo/IMG.JPG')`

คำสั่งที่ใช้ในการถ่ายภาพคือ คำสั่ง `camera.capture`

โปรแกรมที่ 11-1 โค้ดภาษา Python ไฟล์ CamreraControl.py สำหรับบอร์ด Raspberry Pi 2 เพื่อควบคุมโมดูลกล้อง Raspberry Pi ให้ถ่ายภาพนิ่ง



รูปที่ 11-15 แสดงตำแหน่งของไฟล์ภาพถ่ายที่ได้จากการทำงานของโปรแกรมที่ 11-1

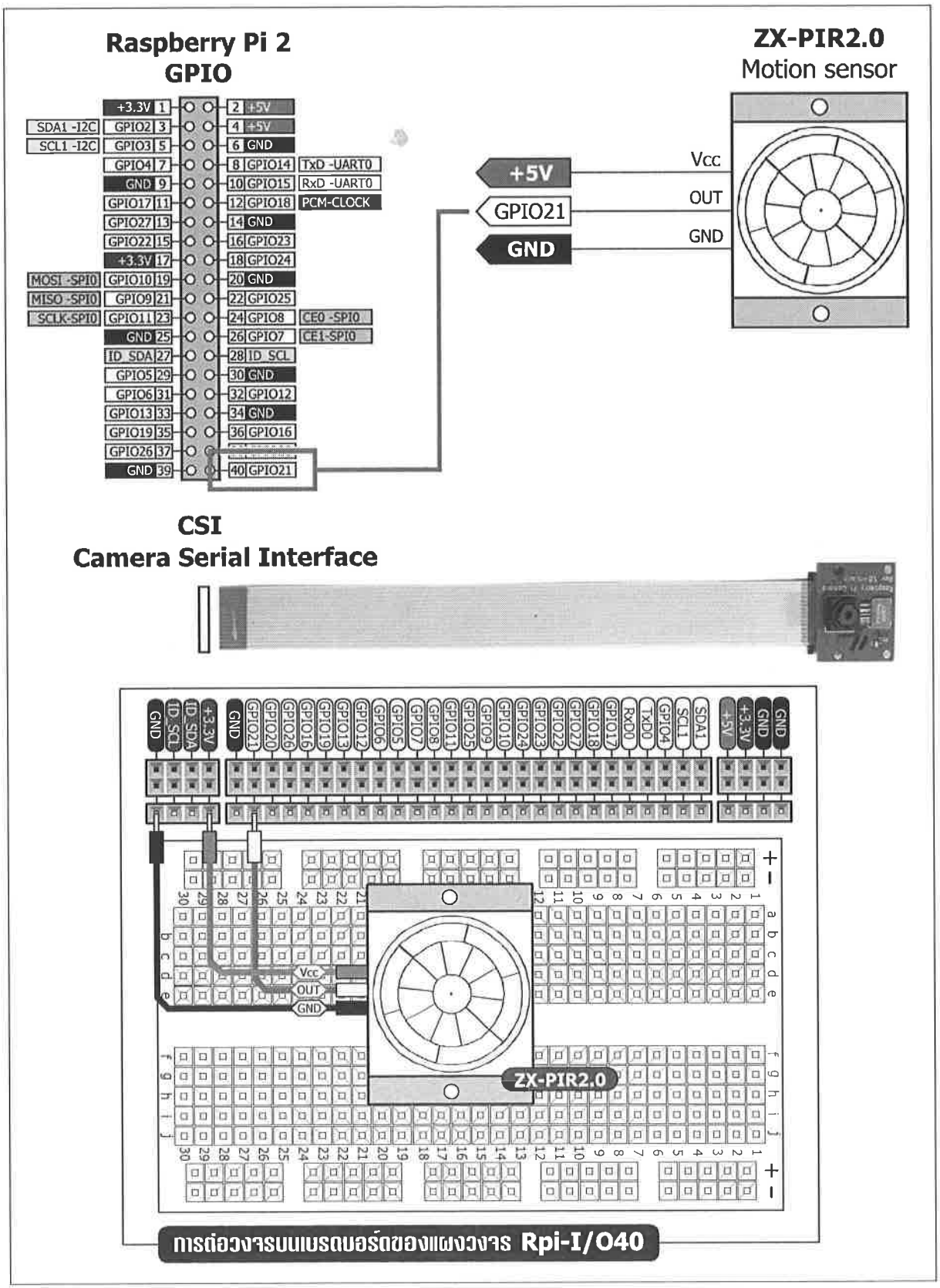
11.6.2 ควบคุมการถ่ายภาพจากการตรวจจับความเคลื่อนไหว

ในตัวอย่างนี้นำโมดูลตรวจจับความเคลื่อนไหว ZX-PIR2.0 มาช่วยในการสั่งให้ถ่ายภาพแบบอัตโนมัติ มีขั้นตอนการทดลองและทดสอบดังนี้

(1) ต่อวงจรตามรูปที่ 11-16

```
import picamera
import datetime
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(21, GPIO.IN)
camera = picamera.PiCamera()
i=0
while (1):
  state=GPIO.input(21)
  while (state):
    img_name="Photo/"
    img_name+=str(datetime.datetime.now())
    img_name+=".jpg"
    camera.capture(img_name)
    i=i+1
    time.sleep(1)
    print("Takephoto=",i)
    state=GPIO.input(21)
```

โปรแกรมที่ 11-2 โค้ดภาษา Python ไฟล์ CamreraPIR.py สำหรับบอร์ด Raspberry Pi 2 เพื่อควบคุมโมดูลกล้อง Raspberry Pi ให้ถ่ายภาพนิ่งโดยมีตัวตรวจจับความเคลื่อนไหวเป็นตัวสั่งการให้ถ่ายภาพโดยอัตโนมัติ



รูปที่ 11-16 วงจรและตัวอย่างการเชื่อมต่อบอร์ด Raspberry Pi 2 กับโมดูลกล้อง Raspberry Pi และโมดูล ZX-PIR2.0 เพื่อสร้างระบบถ่ายภาพนิ่งอัตโนมัติจากการตรวจจับความเคลื่อนไหวของมนุษย์

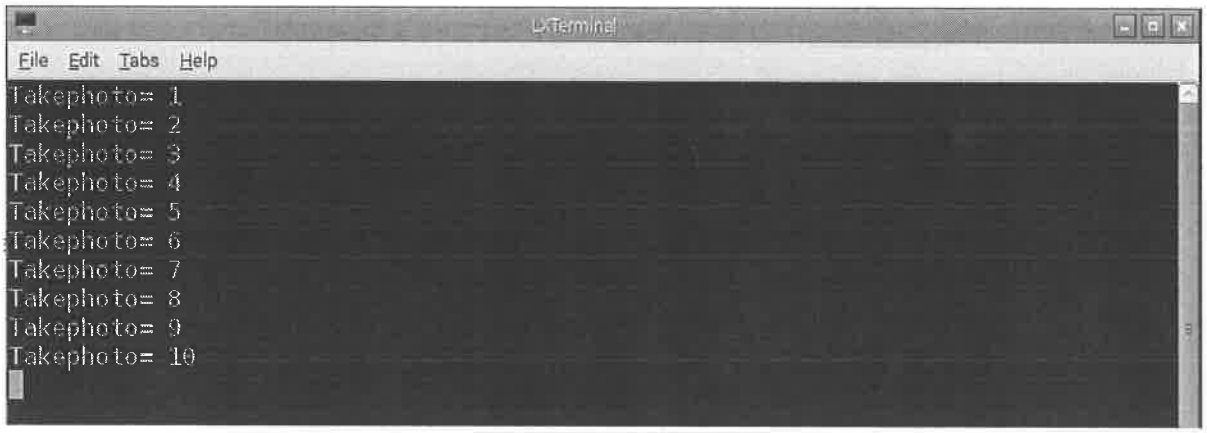
(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(2) พิมพ์คำสั่งตามโปรแกรมที่ 11-2

(3) บันทึกไฟล์ชื่อ CameraPIR.py

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

โมดูลกล้องจะเริ่มถ่ายภาพเมื่อโมดูล ZX-PIR ตรวจจับพบความเคลื่อนไหว โดย Raspberry Pi 2 จะแสดงจำนวนครั้งที่ทำการถ่ายภาพ และบันทึกไฟล์ภาพถ่ายไว้ในโฟลเดอร์ Photo โดยใช้ค่าของวันเวลาที่ถ่ายภาพในการตั้งชื่อไฟล์ ส่งผลให้โอกาสที่จะมีชื่อไฟล์เหมือนกันเกิดขึ้นได้ยากมาก ดังรูปที่ 11-17



รูปที่ 11-17 หน้าต่างเทอร์มินอลแสดงผลการทำงานของโปรแกรมที่ 11-2

11.6.3 ควบคุมการบันทึกภาพเคลื่อนไหวจากการตรวจจับความเคลื่อนไหว

ในตัวอย่างนี้ต่อยอดจากโปรแกรมที่ 11-2 เปลี่ยนจากการถ่ายภาพนิ่ง มาเป็นการบันทึกภาพเคลื่อนไหว มีขั้นตอนการทดลองและทดสอบดังนี้

(1) ใช้การเชื่อมต่ออุปกรณ์ตามรูปที่ 11-15 ในการทดสอบ

(2) สร้างไฟล์สคริปต์ Python ด้วย Geany หรือ nano เท็กซ์เอดิเตอร์

(3) พิมพ์คำสั่งตามโปรแกรมที่ 11-3

(4) บันทึกไฟล์ชื่อ CameraVideo.py

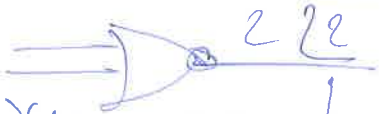
```
import picamera
import datetime
import RPi.GPIO as GPIO
import time
input_pin=21
GPIO.setmode(GPIO.BCM)
GPIO.setup(input_pin, GPIO.IN)
GPIO.setup(input_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
camera = picamera.PiCamera()

while (1):
  GPIO.wait_for_edge(input_pin, GPIO.RISING)
  print('startrecording')
  video_name="Photo/"
  video_name+=str(datetime.datetime.now())
  video_name+=".h264"
  camera.resolution = (640, 480)
  camera.start_recording(video_name)
  GPIO.wait_for_edge(input_pin, GPIO.FALLING)
  camera.stop_recording()
  print('stoprecording')
```

โปรแกรมที่ 11-3 โค้ดภาษา Python ไฟล์ CamreraVideo.py สำหรับบอร์ด Raspberry Pi 2 เพื่อควบคุมโมดูลกล้อง Raspberry Pi ให้บันทึกภาพเคลื่อนไหวโดยมีตัวตรวจจับความเคลื่อนไหว ZX-PIR เป็นตัวสั่งการให้บันทึกภาพโดยอัตโนมัติ

(5) สั่งให้ไฟล์สคริปต์ทำงานด้วยการเลือกเมนู **Build** > **Execute** บนโปรแกรม Geany หรือเปิดหน้าต่างเทอร์มินอลแล้วสั่งรันด้วยคำสั่ง **`sudo python3 CameraVideo.py`**

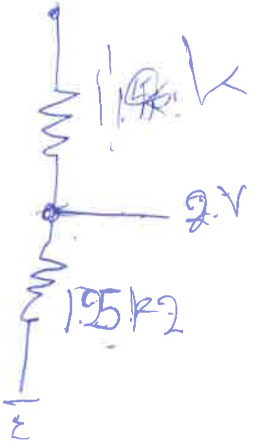
โมดูลกล้องจะเริ่มบันทึกภาพเคลื่อนไหวเมื่อโมดูล ZX-PIR ตรวจจับพบความเคลื่อนไหว โดย Raspberry Pi 2 จะแสดงข้อความ **`startrecording`** *ที่หน้าต่างเทอร์มินอล จนกว่าเป้าหมายจะหยุดการเคลื่อนไหว จากนั้นจะแสดงข้อความ* **`stoprecording`** *เพื่อสิ้นสุดการบันทึกภาพเคลื่อนไหว ดังรูปที่ 11-18*

รูปที่ 11-18 หน้าต่างเทอร์มินอลแสดงผลการทำงานของโปรแกรมที่ 11-3



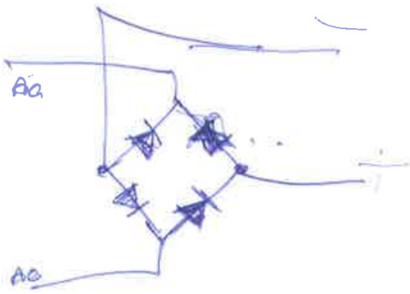
4

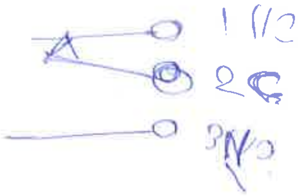
2 | 10
2 | 5 0
2 | 2 1
1 0

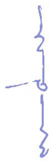


10 | 11001
5 | 3 | 1

8 4 2 1
1 0 1 1
1001

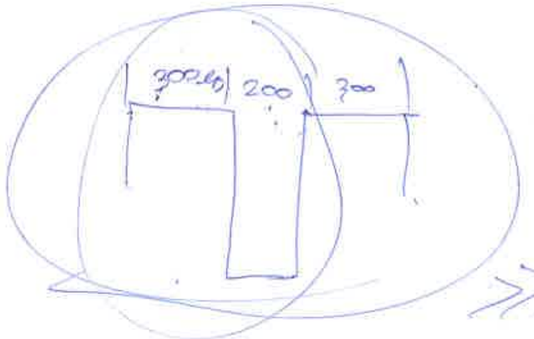






22

010010



15
10

>>> print (4 || 2 || 1)

$f = \frac{1}{T}$

$\frac{1}{500}$

1 ใบ[illegible] [illegible] printf

2 บทที่ 12: [illegible] Html

3 [illegible] ระดับ[illegible].

4 บทที่ 13 อ่าน-เขียน[illegible].

5 บทที่ 12 [illegible].

6 [illegible] text file บทที่ 12.

2 kHz 9701