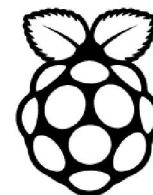


บทที่ 13

Raspberry Pi 2 กับการพัฒนาอุปกรณ์ Internet of Things (IoT)



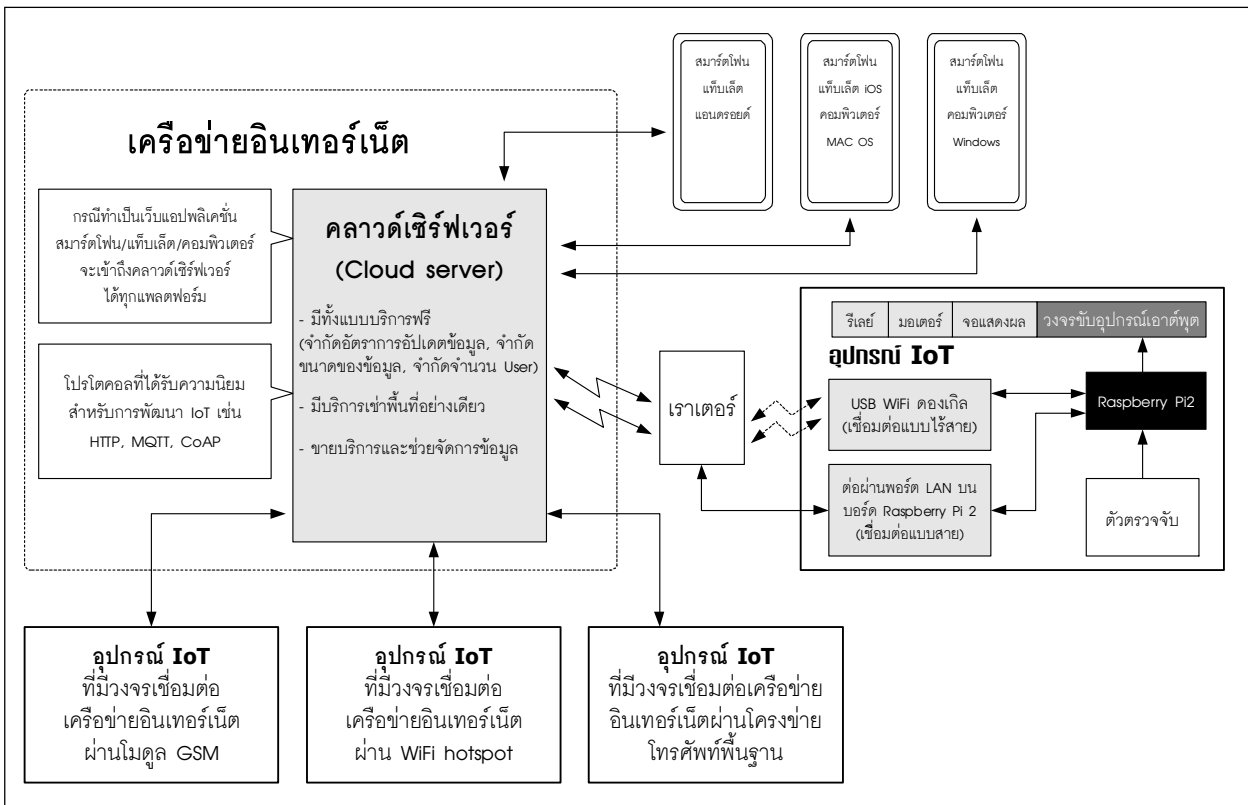
บอร์ด Raspberry Pi 2 ไม่เพียงมีความสามารถในการเชื่อมต่อกับระบบเครือข่ายผ่าน LAN หรือ WiFi เพื่อสร้างระบบควบคุมผ่านเครือข่ายดังตัวอย่างที่นำเสนอในบทที่ 12 เท่านั้น ด้วยช่องทางการติดต่อระบบเครือข่ายที่มีอยู่ยังสามารถทำให้บอร์ด Raspberry Pi 2 สามารถเชื่อมต่อกับเครือข่ายอินเทอร์เน็ต นำไปสู่การสร้างระบบบันทึกและตรวจสอบข้อมูลหรือการสร้างระบบควบคุมอุปกรณ์ไฟฟ้าผ่านเครือข่ายอินเทอร์เน็ตได้ นั่นหมายความว่า บอร์ด Raspberry Pi 2 มีความสามารถอย่างเพียงพอต่อการพัฒนาเป็นอุปกรณ์ IoT (Internet of Things) หรือเชื่อมต่อกับฐานข้อมูลก้อนเมฆหรือคลาวด์เซิร์ฟเวอร์อันเป็นหัวใจสำคัญสูงสุดของระบบ IoT หรืออินเทอร์เน็ตของสรรพสิ่ง

ในบทนี้นำเสนอข้อมูล ขั้นตอนวิธีการในการเชื่อมต่อบอร์ด Raspberry Pi 2 กับอุปกรณ์ต่างๆ รวมถึงคลาวด์เซิร์ฟเวอร์เพื่อสร้างเป็นระบบหรืออุปกรณ์ IoT เบื้องต้น (แต่ความสามารถของบอร์ด Raspberry Pi 2 ทำได้มากกว่านั้น) โดยจะเริ่มจากการให้ข้อมูลเบื้องต้นของ IoT ตามด้วยการแนะนำคลาวด์เซิร์ฟเวอร์ BeeBotte จากนั้นจึงเข้าสู่ขั้นตอนการทดลองเพื่อพัฒนาเป็นอุปกรณ์ IoT ของบอร์ด Raspberry Pi 2 และทำการทดสอบเพื่อแสดงให้เห็นผลการทำงานที่เกิดขึ้นจริง

13.1 อะไรคือ Internet of Things (IoT)

Internet of Things คำนี้เกิดขึ้นมาตั้งแต่ปี ค.ศ. 1999 โดย *Kevin Ashton* แห่ง *MIT's Media center* เขาได้นำเสนอแนวคิดว่ามันคือ การนำสิ่งของต่างๆ ไม่ว่าจะเป็นคอมพิวเตอร์, เครื่องจักร และตัวตรวจจับมาเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตเพื่อรายงานสถานะการทำงาน สถานะข้อมูล และรับรู้คำสั่งควบคุม สิ่งที่น่าประหลาดใจคือ ในช่วงเวลานั้น โลกเพิ่งรู้จักและใช้งานอินเทอร์เน็ตได้ไม่นาน แต่ Kevin มองเห็นอนาคตและพัฒนาการของสรรพสิ่งที่จะต้องเชื่อมโยงถึงกันผ่านเครือข่ายอินเทอร์เน็ต

แม้ว่าแนวคิดของ IoT ถูกนำเสนอตั้งแต่ปี ค.ศ. 1999 แต่ไม่ได้รับการตอบรับมากนัก อาจมาจากสาเหตุที่ว่า ในเวลานั้นอินเทอร์เน็ตเป็นเรื่องกลุ่มคนเฉพาะ ค่อนข้างยาก และต้องการทรัพยากรมาก แต่ก็มีคนนำแนวคิด IoT ไปสานต่อ และมีชื่อเรียกแตกต่างกันไป อาทิ Machine-to-machine (M2M), Ubiquitous Computing, Embedded Computing, Smart Service, Industrial Internet จนกระทั่งวันนี้เมื่ออินเทอร์เน็ตเข้าถึงทุกคน ทุกบ้าน ทำให้แนวคิด Internet of Things ได้รับการยอมรับ และเรียกขานเทคโนโลยีด้วยชื่อเดิมที่ถูกคิดมาตั้งแต่ปี ค.ศ. 1999



รูปที่ 13-1 ไดอะแกรมการทำงานเบื้องต้นของระบบ Internet of Things หรือ IoT

IoT หรือ Internet of Things หมายถึง เทคโนโลยีที่ก่อให้เกิดการเชื่อมโยงกันของสิ่งของ ผู้คน ข้อมูล และการบริการเข้ากับเครือข่ายอินเทอร์เน็ต ปัจจัยสำคัญในการทำให้เกิด IoT ได้คือ การบรรจุ อุปกรณ์สมองกลฝังตัวหรือ embedded system device เข้าไปใน “สิ่งของ” หรือเครื่องมือ เครื่องใช้ต่างๆ มีตัวตรวจจับหรือเซนเซอร์เพื่อตรวจวัดค่าที่สนใจ แล้วส่งมายังส่วนสมองกล เพื่อส่งต่อมายังส่วนประมวลผลกลางและฐานข้อมูลผ่านเครือข่ายอินเทอร์เน็ต ในส่วนหลังนี้มีชื่อเรียกด้วยศัพท์สมัยใหม่ว่า **คลาวด์เซิร์ฟเวอร์ (cloud server)**

ด้วยการนำอุปกรณ์สมองกลฝังตัวบรรจุลงใน “สิ่งของ” ต่างๆ ทำให้ “สิ่งของ” เหล่านั้นทำงานในแบบอัจฉริยะได้ อุปกรณ์เครื่องใช้ต่างๆ ในบ้าน ในโรงงาน ในที่ทำงาน ในยานพาหนะ ล้วนแล้วแต่ใช้ระบบสมองกลฝังตัวมากขึ้น ทำให้มันทำงานได้ด้วยตัวเอง และ/หรือรวมเข้าเป็นส่วนหนึ่งของระบบใหญ่ เกิดการเชื่อมโยงการทำงานเป็นระบบได้ การทำให้ “สิ่งของ” ทำงานร่วมกันผ่านเครือข่ายอินเทอร์เน็ต จึงทำให้เกิดนิยามของเทคโนโลยีนี้ขึ้น Internet of Things หรือ IoT เป็นการขยายขอบเขตการทำงานของอินเทอร์เน็ตให้กว้างและลึกไกลไปถึงการเชื่อมต่อเพื่อสื่อสารข้อมูลกับ “สิ่งของ” ทำให้เกิดการรับส่งข้อมูลและตอบสนองในแบบทุกที่ ทุกเวลา และทุกสิ่งของได้ในที่สุด

ระบบหรือเทคโนโลยี IoT จะเกิดขึ้นได้ต้องมีองค์ประกอบครบดังนี้

1. สิ่งของ
2. อุปกรณ์ (ตัวควบคุม, ตัวตรวจจับ และอุปกรณ์ขับโหลดหรืออุปกรณ์เอาต์พุต)
3. ระบบเชื่อมต่ออินเทอร์เน็ต (จะเป็นแบบมีสายหรือไร้สายก็ได้)
4. ข้อมูล
5. ระบบจัดการฐานข้อมูลคลาวด์เซิร์ฟเวอร์ (Cloud server)

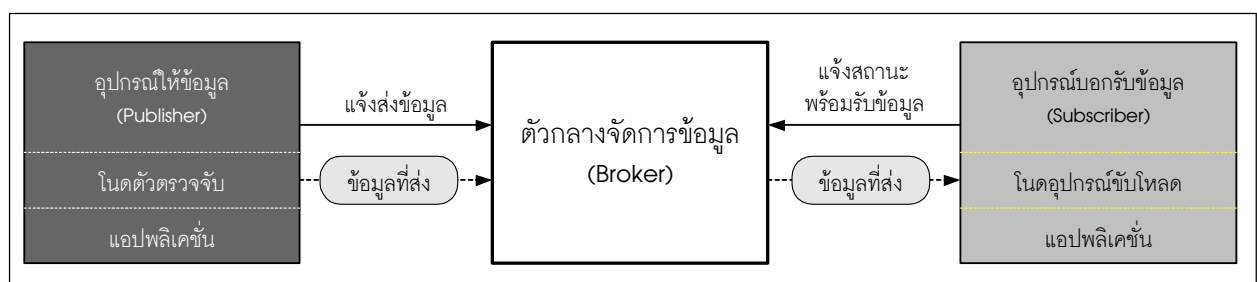
รูปที่ 13-1 แสดงไดอะแกรมการทำงานอย่างง่ายของ Internet of Things หรือ IoT ที่มีองค์ประกอบครบ ทั้งยังแสดงให้เห็นแนวทางในการนำบอร์ด Raspberry Pi 2 มาพัฒนาเป็นอุปกรณ์ IoT ด้วย ซึ่งจะได้นำเสนอตัวอย่างขั้นต้นในบทนี้

13.2 แนะนำคลาวด์เซิร์ฟเวอร์ที่ชื่อ Beebotte

13.2.1 เกี่ยวกับคลาวด์เซิร์ฟเวอร์

IoT จะเกิดขึ้นอย่างสมบูรณ์ไม่ได้หากขาดองค์ประกอบที่เรียกว่า คลาวด์เซิร์ฟเวอร์ (cloud server) โดยคลาวด์เซิร์ฟเวอร์เป็นตัวกลางของการรับส่งข้อมูลดังแสดงไดอะแกรมการทำงานอย่างง่ายในรูปที่ 13-2 จะเห็นว่ามีส่วนประกอบสำคัญ 3 ส่วนคือ

1. อุปกรณ์ให้ข้อมูลหรือพบลิชเซอร์ (Publisher) เป็นอุปกรณ์ที่ส่งข้อมูลตั้งต้นเข้ามาในระบบ อาจมองได้ง่ายขึ้นหากยกตัวอย่างเป็นโนคตัวตรวจจับ (sensor node) เมื่อตัวตรวจจับวัดค่าทางกายภาพได้ จะส่งต่อไปให้อุปกรณ์ควบคุมหลักซึ่งในที่นี้มักเป็น ไมโครคอนโทรลเลอร์ แล้วส่งข้อมูลไปยังคลาวด์เซิร์ฟเวอร์ ซึ่งทำหน้าที่เป็นตัวจัดการข้อมูล
2. ตัวกลางจัดการข้อมูลหรือโบรกเกอร์ (Broker) คลาวด์เซิร์ฟเวอร์จะทำหน้าที่หลักในส่วนนี้เพื่อรับข้อมูลจากอุปกรณ์ให้ข้อมูลหรือพบลิชเซอร์ นำมาพักไว้หรือบันทึกไว้เพื่อรอการร้องขอหรือกระจายข้อมูลไปยังอุปกรณ์รับข้อมูลหรือซับสไครเบอร์ รวมถึงการส่งต่อไปแสดงผลด้วยแดชบอร์ด (dashboard) หรือหน้าปัดแสดงผลบนเว็บไซต์ด้วย



รูปที่ 13-2 ไดอะแกรมการทำงานเบื้องต้นของคลาวด์เซิร์ฟเวอร์

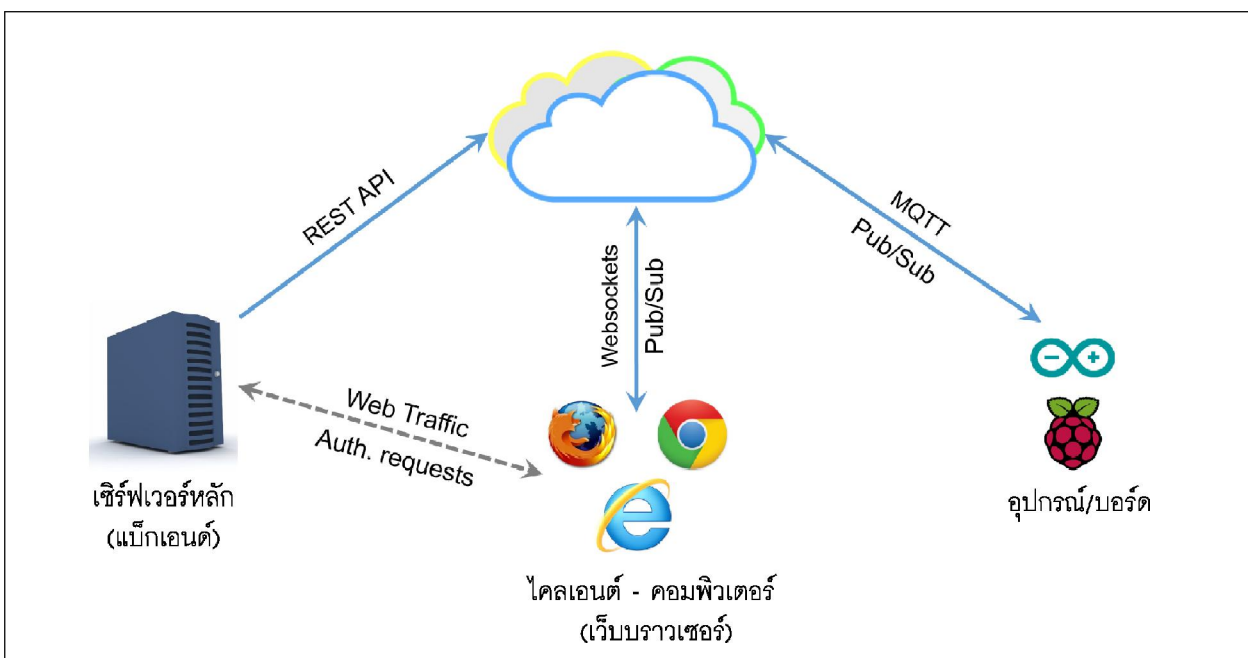
3. อุปกรณ์รับข้อมูลหรือซัพสไครเบอร์ (Subscriber) เป็นอุปกรณ์ที่ต้องการข้อมูลเพื่อนำมาดำเนินการอย่างใดอย่างหนึ่ง เช่น นำมาแสดงผล นำมาบันทึก นำมาควบคุมอุปกรณ์หรือโหลด อาจยกตัวอย่างเป็น ปลั๊กไฟ IoT ที่นำข้อมูลจากคลาวด์เซิร์ฟเวอร์มาทำการเปิดปิดอุปกรณ์ไฟฟ้าที่ต่ออยู่

ดังนั้นจะเห็นว่า ในกระบวนการ IoT แทบจะไม่มี “คน” เข้าไปร่วมทำงาน หากแต่ “คน” จะเป็นผู้สร้างระบบ กำหนดรูปแบบการสื่อสารข้อมูล จากนั้นอุปกรณ์ทั้งหมดจะ “คุยกันเอง” ผ่านเครือข่ายอินเทอร์เน็ต โดยมีคลาวด์เซิร์ฟเวอร์เป็นตัวกลาง

13.2.2 คลาวด์เซิร์ฟเวอร์ Beebotte

Beebotte (ออกเสียงว่า บีบอต - มาจากภาษาฝรั่งเศส) เป็นคลาวด์เซิร์ฟเวอร์ที่พัฒนาขึ้นจากนักพัฒนา (developer) เพื่อนักพัฒนาด้วยกัน สำนักงานใหญ่อยู่ในกรุงปารีส ประเทศฝรั่งเศส เว็บไซต์สำหรับให้ข้อมูลและลงทะเบียนเพื่อใช้งานคือ <http://beebotte.com> โดยมีความสามารถและข้อเด่นดังนี้

- เตรียมการเชื่อมต่ออุปกรณ์และแสดงผลข้อมูลได้ในแบบเวลาจริง (real time) จากอุปกรณ์ IoT ในทุกที่ ทุกเวลา
- มีวิดเจ็ต (widget) ให้เลือกใช้หลากหลาย (วิดเจ็ตคือชุดคำสั่ง โปรแกรมขนาดเล็กที่รองรับการอินเตอร์เฟสกับแอปพลิเคชันหรือระบบปฏิบัติการ ที่พบกันบ่อยๆ เช่น ปุ่ม ไอคอน และแถบเมนู
- แสดงผลข้อมูลที่นำขึ้นมาเก็บไว้ได้แบบเวลาจริงในรูปแบบกราฟิกและตัวเลข
- มีรูปแบบการให้บริการทั้งแบบไม่มีค่าใช้จ่ายและแบบมีค่าใช้จ่ายตามความสามารถและขนาดของข้อมูล



รูปที่ 13-3 การทำงานของคลาวด์เซิร์ฟเวอร์ Beebotte กับคอมพิวเตอร์และฮาร์ดแวร์สมองกลฝังตัว

รูปแบบการทำงานของ Beebotte แสดงดังในรูปที่ 13-3 จะเห็นว่า Beebotte มี API (Application Programming Interface) และ โพรโทคอล MQTT สำหรับเชื่อมต่อกับบอร์ด Raspberry Pi และฮาร์ดแวร์ระบบเปิดชื่อดังอย่าง Arduino ด้วย รองรับการเข้าถึงเพื่อดู อ่าน และตรวจสอบข้อมูลผ่านเว็บเบราว์เซอร์ทุกแพลตฟอร์มหากใช้งานด้วยเครื่องคอมพิวเตอร์

รูปแบบการให้บริการของ Beebotte มี 4 รูปแบบดังแสดงในตารางที่ 13-1 สำหรับนักทดลองหรือผู้เริ่มต้น ขอแนะนำให้ผู้เริ่ม XS ก่อน เพราะไม่มีค่าใช้จ่ายและมีคุณสมบัติที่ดีพอสมควร ทั้งการไม่จำกัดจำนวนช่อง, รองรับการส่งผ่านข้อความได้มากถึง 50,000 ข้อความต่อวัน, บันทึกข้อความได้ 5,000 ข้อความ และที่น่าสนใจมากๆ คือ ตัวคลาวเซิร์ฟเวอร์นี้ยังเก็บรักษาข้อมูลย้อนหลังให้ยาวนานถึง 3 เดือน ซึ่งหาได้ยากมากสำหรับคุณสมบัติข้อนี้ในคลาวเซิร์ฟเวอร์ที่ให้บริการแบบไม่มีค่าใช้จ่าย รวมถึงการปกป้องข้อมูลแบบ SSL (Secure Socket Layer)

รูปแบบการบริการ	จำนวนช่อง	จำนวนข้อความ (ต่อวัน)	การบันทึกข้อความ (ต่อวัน)	ระยะเวลาเก็บข้อมูลย้อนหลัง	การปกป้องข้อมูล	ค่าบริการ (ต่อเดือน)
XS	ไม่จำกัด	50,000	5,000	3 เดือน	SSL	ไม่มีค่าใช้จ่าย
Small		200,000	15,000	12 เดือน		US\$10
Medium		1,000,000	50,000	ไม่จำกัด		US\$30
Large		5,000,000	200,000			US\$120

ข้อมูลนี้นับจนถึงเดือนมกราคม 2559 - รูปแบบการให้บริการและค่าใช้จ่ายบริการอาจเปลี่ยนแปลงได้
ติดตามข้อมูลล่าสุดได้ที่ <http://beebotte.com>

ตารางที่ 13-1 รูปแบบการให้บริการของ Beebotte

ความรู้เกี่ยวกับ SSL

(ข้อมูลจาก : <http://www.sutenm.com>)

SSL คือ โพรโทคอลที่อยู่ระหว่าง Application layer และ Transport layer โดย SSL รองรับการทำงานกับแอปพลิเคชัน โพรโทคอลต่างๆ เช่น HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), Telnet, POP3, SMTP หรือแม้แต่ VPN ได้ SSL ทำงานโดยอาศัยหลักการของการเข้ารหัสข้อมูล (encryption), Message Digests และลายเซ็นอิเล็กทรอนิกส์ (digital signature) โดยแบ่งหน้าที่ออกเป็น 3 ส่วนคือ

1. ตรวจสอบ server ว่าเป็นตัวจริง
2. ตรวจสอบว่า Client เป็นตัวจริงหรือไม่
3. เข้ารหัสลับการเชื่อมต่อ

โดยข้อมูลทั้งหมดที่ถูกส่งระหว่างไคลเอนต์และเซิร์ฟเวอร์จะถูกเข้ารหัสลับ โดยโปรแกรมที่ส่งข้อมูลเป็นผู้เข้ารหัสและโปรแกรมที่รับข้อมูลเป็นผู้ถอดรหัส (โดยใช้วิธี Public key) นอกจากการเข้ารหัสลับในลักษณะนี้แล้ว SSL ยังสามารถป้องกันความถูกต้องสมบูรณ์ของข้อมูลได้อีกด้วย อีกทั้ง ตัวโปรแกรมรับข้อมูลจะทราบได้หากข้อมูลถูกเปลี่ยนแปลงไปในขณะกำลังเดินทางจากผู้ส่งไปยังผู้รับ

13.3 เตรียมการบอร์ด Raspberry Pi 2 เพื่อติดต่อกับ Beebotte

การติดต่อกับคลาวด์เซิร์ฟเวอร์ Beebotte สำหรับคอมพิวเตอร์และบอร์ด Raspberry Pi 2 ทำได้โดยใช้เว็บเบราว์เซอร์ สำหรับบอร์ด Raspberry Pi 2 แนะนำให้ใช้เว็บเบราว์เซอร์ที่ชื่อ *Iceweasel* ซึ่งโดยปกติแล้วในระบบปฏิบัติการ Raspbian ที่ติดตั้งใน SD การ์ดจะไม่ได้ติดตั้งมาให้ โดยส่วนใหญ่จะเป็น Chromium ดังนั้นเมื่อจะใช้บอร์ด Raspberry Pi 2 กับ Beebotte จึงต้องมีการติดตั้งเว็บเบราว์เซอร์ Iceweasel เพื่อใช้งานแทน มีขั้นตอนดังนี้

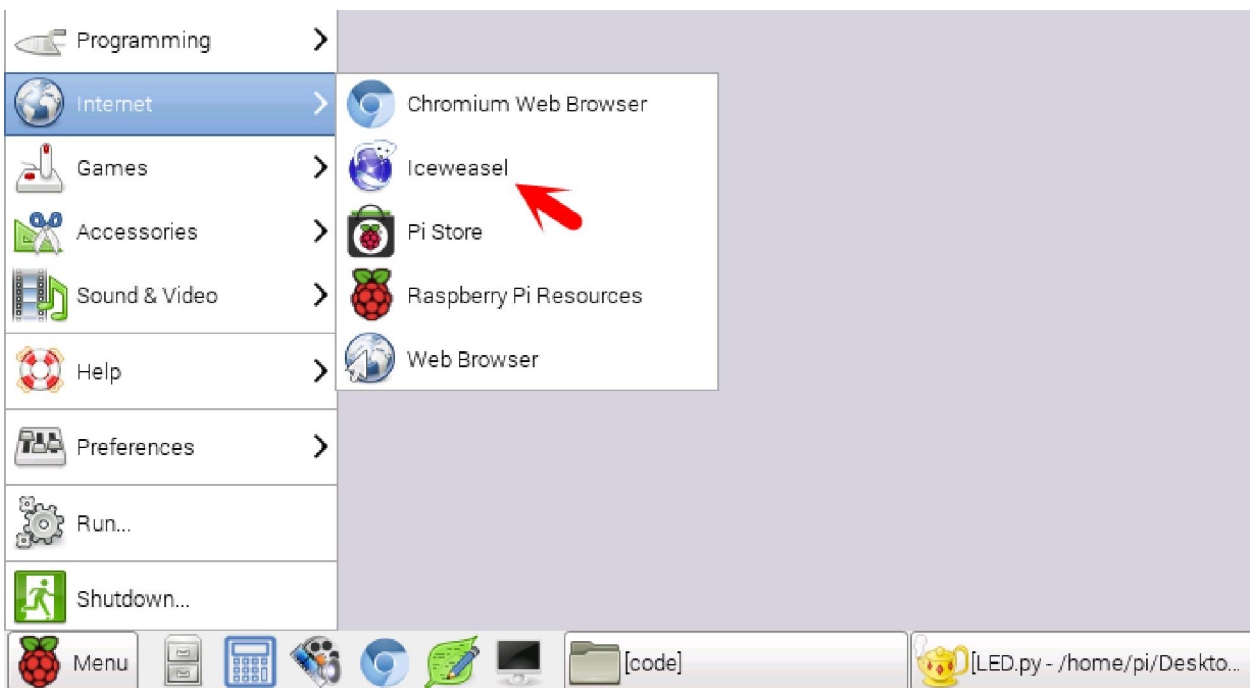
(1) ต่อบอร์ด Raspberry Pi 2 กับเมาส์, คีย์บอร์ด, จอภาพ แล้วจ่ายไฟเลี้ยง อาจใช้วิธีการรีโมตเข้าไปทำงานก็ได้

(2) เชื่อมต่อบอร์ด Raspberry Pi 2 เข้ากับเครือข่ายอินเทอร์เน็ต

(3) ไปที่คอมมานด์พรอมพ์ แล้วพิมพ์คำสั่ง

```
sudo apt-get install iceweasel
```

(4) จะเริ่มต้นการติดตั้งเว็บเบราว์เซอร์ Iceweasel เมื่อเรียบร้อยแล้ว ให้เข้าสู่โหมดกราฟิกด้วยการพิมพ์ startx เลือก Menu เข้าไปที่รายการ Internet จะเห็นไอคอนของเว็บเบราว์เซอร์ Iceweasel ดังรูปที่ 13-4



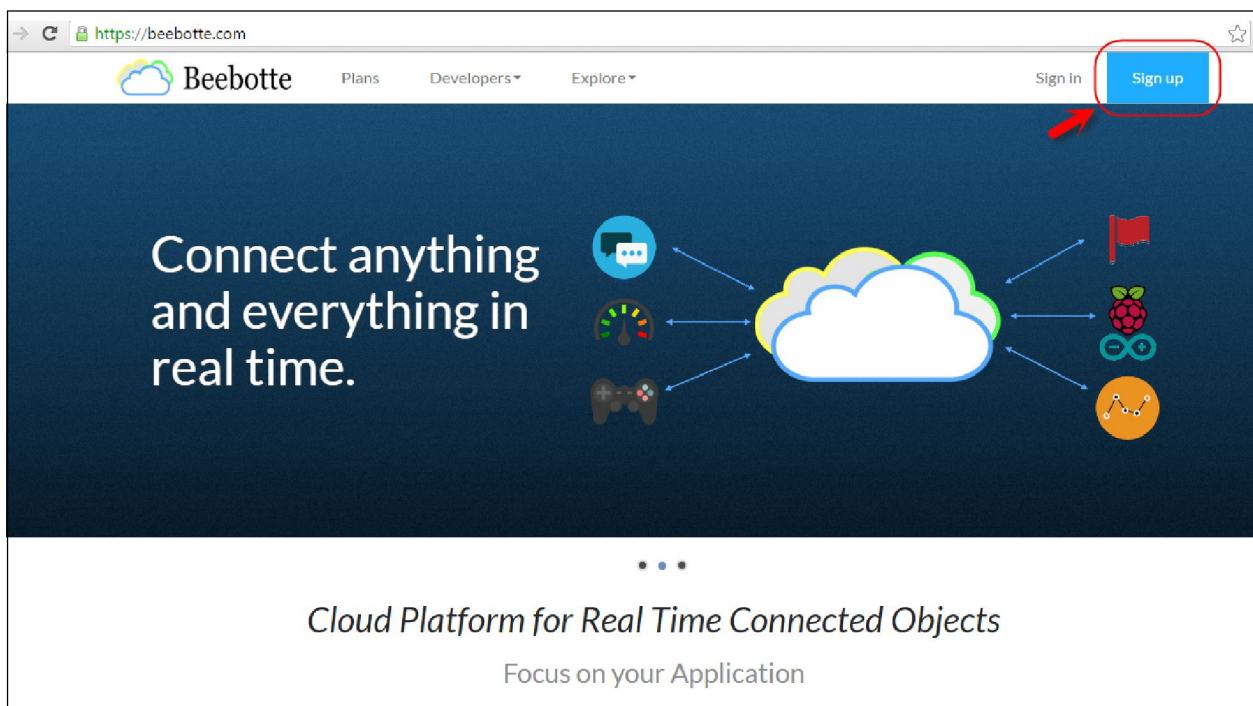
รูปที่ 13-4 การเรียกดูเว็บเบราว์เซอร์ Iceweasel เมื่อติดตั้งลงในบอร์ด Raspberry Pi 2 แล้ว

13.4 การสมัครใช้งาน Beebotte

การสมัครและใช้งาน Beebotte สำหรับบอร์ด Raspberry Pi 2 จะต้องมีการเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตตลอดเวลา ดังนั้นก่อนการใช้งานจะต้องเชื่อมต่อบอร์ด Raspberry Pi 2 กับเครือข่ายอินเทอร์เน็ต และต้องแน่ใจว่า การเชื่อมต่อสมบูรณ์และสามารถเข้าถึงเว็บไซต์ต่างๆ ได้ โดยใช้ได้ทั้งการเชื่อมต่อแบบสายผ่านพอร์ตอีเทอร์เน็ต (LAN) และแบบไร้สายผ่าน WiFi โดยใช้ USB WiFi dongle

เมื่อเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตเรียบร้อยแล้ว จะเข้าสู่การสมัครใช้งาน Beebotte มีขั้นตอนดังนี้

(1) เปิดเว็บเบราว์เซอร์ Iceweasel จากนั้นไปที่เว็บไซต์ <https://beebotte.com/> จะปรากฏหน้าโฮมเพจดังรูปที่ 13-5 จากนั้นคลิกที่ปุ่ม **Sign up** เพื่อลงทะเบียน



รูปที่ 13-5 เว็บไซต์ของ Beebotte จาก <http://beebotte.com> (อาจเปลี่ยนแปลงได้ตามการปรับปรุงของผู้ให้บริการ)

(2) กำหนดชื่อผู้ใช้งานตามต้องการ, อีเมล, กำหนดรหัสผ่าน ดังรูปที่ 13-6 จากนั้นคลิกที่ปุ่ม **SIGN UP** เพื่อยืนยัน

รูปที่ 13-6 ตั้งค่าเพื่อลงทะเบียนใช้งาน Beebotte

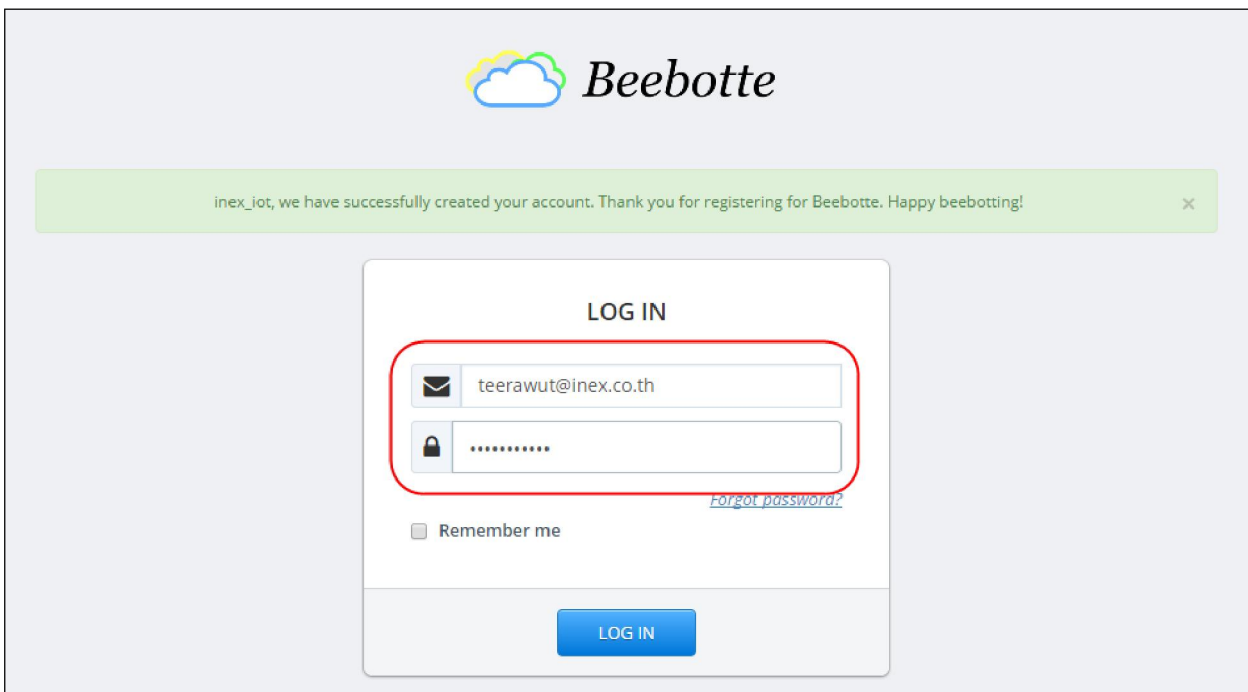


(3) ตรวจสอบอีเมลที่ทำการลงทะเบียนไว้ จะปรากฏข้อความตอบรับจาก Beebotte ดังรูปที่ 13-7 ทำการคลิกลิงก์ที่ได้รับแจ้งมา



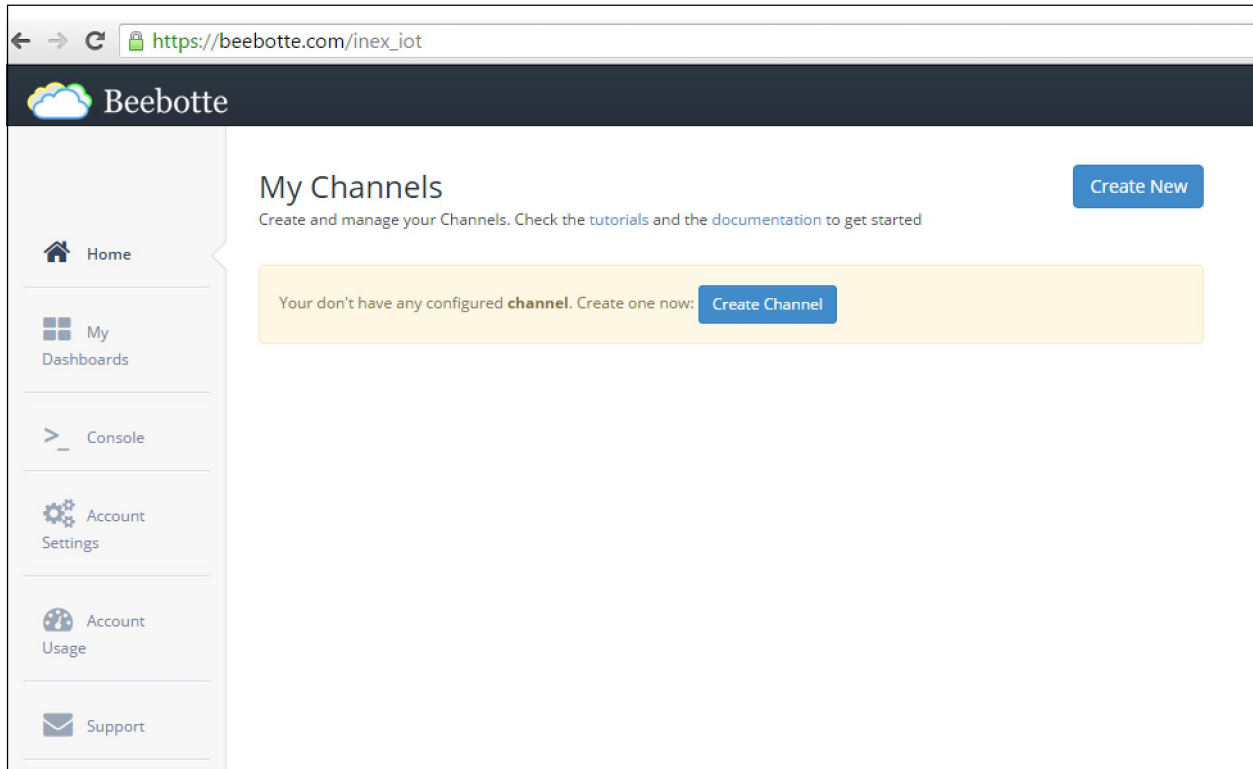
รูปที่ 13-7 แสดงอีเมลตอบรับการลงทะเบียนและลิงก์สำหรับการยืนยันเพื่อใช้งาน Beebotte

(4) ระบบจะนำกลับมายังหน้า LOG IN เพื่อลงชื่อใช้งาน ให้ใส่อีเมลและรหัสผ่านที่ได้ลงทะเบียนไว้ ดังรูปที่ 13-8 จากนั้นคลิกปุ่ม LOG IN



รูปที่ 13-8 หน้า LOG IN ของ Beebotte เพื่อลงชื่อเข้าใช้งาน

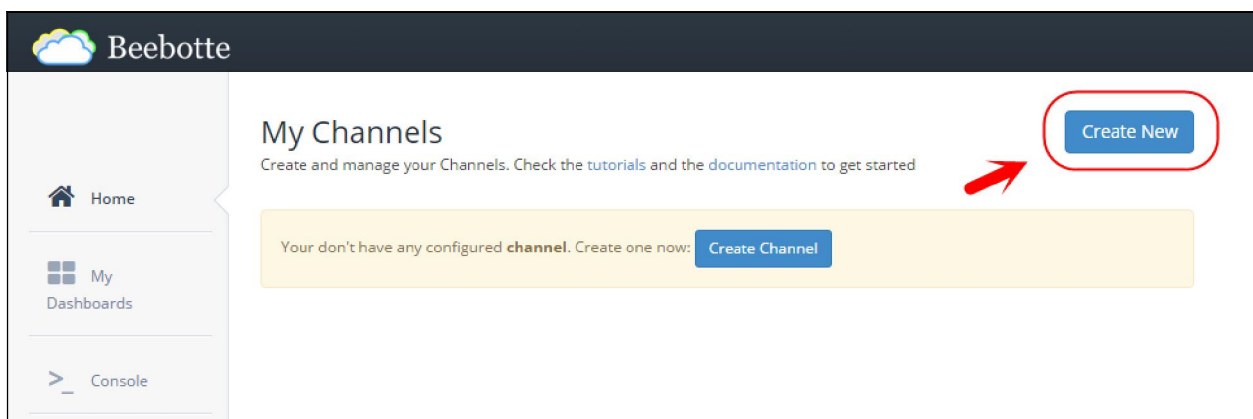
(5) จะปรากฏเว็บเพจดังรูปที่ 13-9 เป็นการยืนยันว่า การลงทะเบียนเพื่อใช้งานเสร็จสมบูรณ์ เริ่มต้นใช้งาน Beebotte ได้ทันที



รูปที่ 13-9 เว็บเพจ My Channels ของ Beebotte เป็นการยืนยันว่า การลงทะเบียนถูกต้อง เริ่มใช้งานได้ตามต้องการ

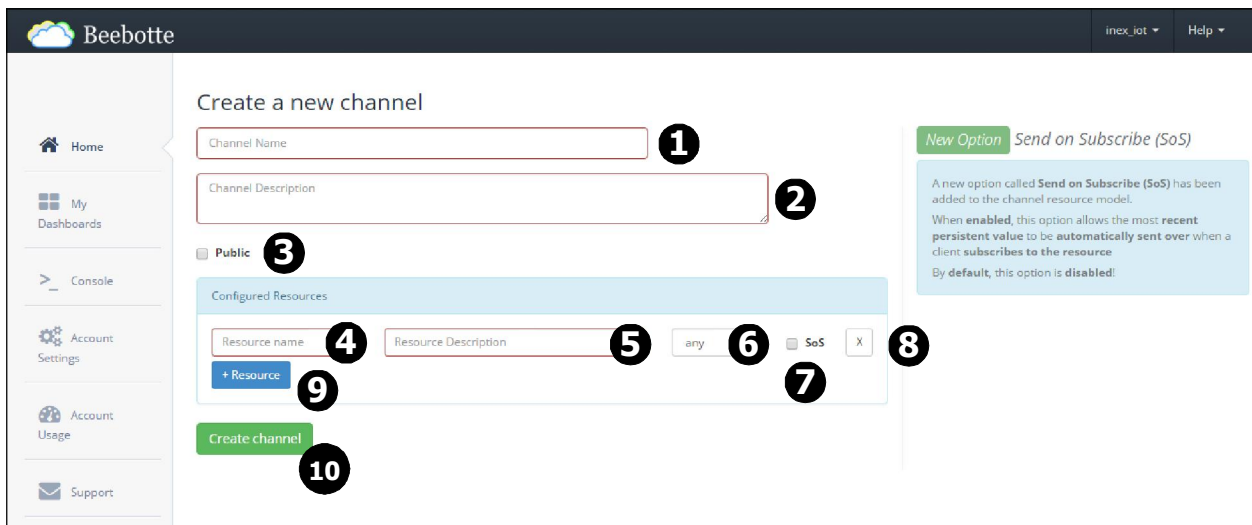
13.5 ส่วนประกอบสำคัญของ Beebotte

การทำงานของ Beebotte จะอยู่ในลักษณะการแบ่งเป็นกลุ่มที่เรียกว่า **ช่องเก็บข้อมูล** หรือ **Channel** และในแต่ละช่องมีส่วนประกอบหรือเรียกว่า **Resource** เมื่อทำการลงชื่อเข้าใช้งานหรือ LOG IN จะมีปุ่ม **Create New** ปรากฏขึ้นที่มุมขวาบน ดังรูปที่ 13-10



รูปที่ 13-10 เว็บเพจเพื่อเริ่มต้นสร้างช่องเก็บข้อมูลใหม่เพื่อติดต่อกับ Beebotte

คลิกที่ปุ่ม **Creat New** เพื่อเริ่มต้นสร้างช่องเก็บข้อมูล เว็บเพจ **Create a new channel** จะปรากฏขึ้น มาดูรูปที่ 13-11



รูปที่ 13-11 เว็บเพจ **Create a new channel** สำหรับป้อนข้อมูลเพื่อสร้างช่องเก็บข้อมูล

- ❶ ชื่อของช่องเก็บข้อมูลหรือ Channel
- ❷ คำอธิบายของช่องเก็บข้อมูล
- ❸ กำหนดช่องเก็บข้อมูลให้มีสถานะเป็นสาธารณะ
- ❹ ชื่อของส่วนประกอบหรือ Resource
- ❺ คำอธิบายส่วนประกอบ
- ❻ กำหนดประเภทข้อมูลของส่วนประกอบ
- ❼ กำหนดให้สามารถร้องขอเพื่อรับข้อมูลหรือซบสไครบ์ได้ (subscribe) ได้
- ❽ คลิกเพื่อลบส่วนประกอบที่ไม่ต้องการ
- ❾ เพิ่มส่วนประกอบ
- ❿ ปุ่มสร้างช่องเก็บข้อมูลเมื่อทำการตั้งค่าต่าง ๆ เรียบร้อยแล้ว

13.6 ตัวอย่างการออกแบบคลาวด์เซิร์ฟเวอร์ สำหรับแสดงค่าความชื้นสัมพัทธ์และอุณหภูมิ

ลำดับต่อไปเป็นการแนะนำขั้นตอนการออกแบบคลาวด์เซิร์ฟเวอร์บน Beebotte เพื่อเก็บและแสดงข้อมูลของความชื้นสัมพัทธ์และอุณหภูมิที่บอร์ด Raspberry Pi 2 อ่านได้จาก DHT11 โมดูลวัดความชื้นสัมพัทธ์และอุณหภูมิ โดยมีขั้นตอนหลัก 3 ขั้นตอนดังนี้

- (1) สร้างช่องเก็บข้อมูลที่มีชื่อว่า **RaspberryPi**
- (2) สร้างส่วนประกอบที่มีชื่อว่า **Temp** ชนิดข้อมูลเป็น **temperature** เพื่อเก็บค่าอุณหภูมิ
- (3) สร้างส่วนประกอบที่มีชื่อว่า **humi** ชนิดข้อมูลเป็น **humidity** เพื่อเก็บข้อมูลค่าความชื้นสัมพัทธ์

13.6.1 เริ่มต้นสร้างช่องเก็บข้อมูล

(13.6.1.1) กำหนดชื่อช่องเก็บข้อมูลหรือ Channel เป็น *RaspberryPi*, ชื่อส่วนประกอบหรือ Configured Resource เป็น *Temp*, ชนิดข้อมูลเป็น *temperature* และค่าอื่นๆ ดังรูปที่ 13-12 จากนั้นคลิกที่ปุ่ม **+ Resource** เพื่อเพิ่มส่วนประกอบอีกหนึ่งตัว

The screenshot shows a web interface for creating a new channel. The title is "Create a new channel". There are two input fields: the first contains "RaspberryPi" and the second contains "raspberrypi Cloud Example". Below these is a checkbox labeled "Public" which is checked. A section titled "Configured Resources" contains a table with one resource: "Temp" with the type "temperature data". To the right of this resource are a dropdown menu showing "tempere", a "SoS" checkbox, and an "X" button. At the bottom left of this section is a blue button labeled "+ Resource" with a red arrow pointing to it.

รูปที่ 13-12 กำหนดชื่อช่องเก็บข้อมูลใหม่และส่วนประกอบต่างๆ

(13.6.1.2) ส่วนประกอบที่เพิ่มขึ้นใหม่ กำหนดชื่อเป็น *humi*, ชนิดข้อมูลเป็น *humidity* และค่าอื่นๆ ตามรูปที่ 13-13 จากนั้นคลิกที่ปุ่ม **Create channel** เพื่อสร้างช่องเก็บข้อมูลตามที่ต้องการ

The screenshot shows the 'Create a new channel' form. The 'Name' field contains 'RaspberryPi' and the 'Channel Name' field contains 'raspberry Pi Cloud Example'. The 'Public' checkbox is checked. Under the 'Configured Resources' section, there are two resource entries: 'Temp' with 'temperature data' and 'humi' with 'humidity data'. The 'humi' entry is circled in red. A red arrow points to the 'Create channel' button at the bottom.

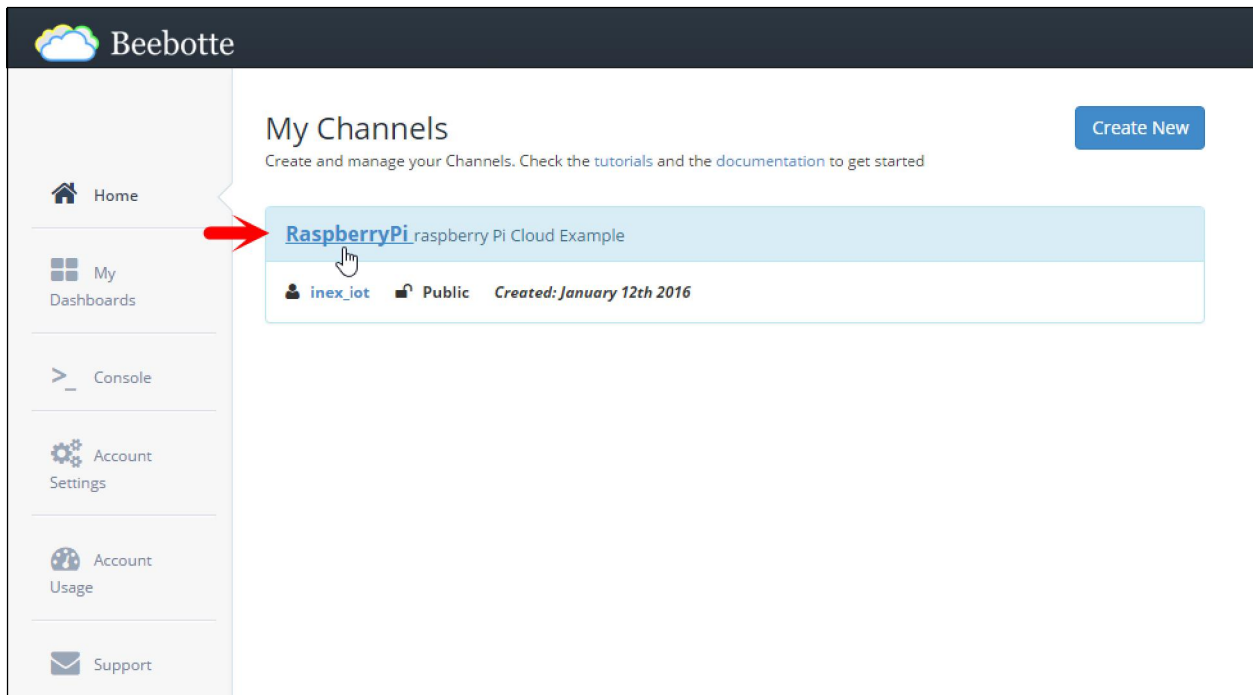
รูปที่ 13-13 กำหนดค่าส่วนประกอบที่เพิ่มขึ้นใหม่

(13.6.1.3) ระบบจะกลับมายังเว็บเพจ **My Channels** จะแสดงชื่อช่องเก็บข้อมูล **RaspberryPi** และคำอธิบายดังรูปที่ 13-14

The screenshot shows the 'My Channels' page. The page title is 'My Channels' and it includes a 'Create New' button. Below the title, there is a list of channels. One channel is highlighted with a red circle: 'RaspberryPi raspberrry Pi Cloud Example'. The channel is owned by 'inex_iot', is 'Public', and was created on 'January 12th 2016'.

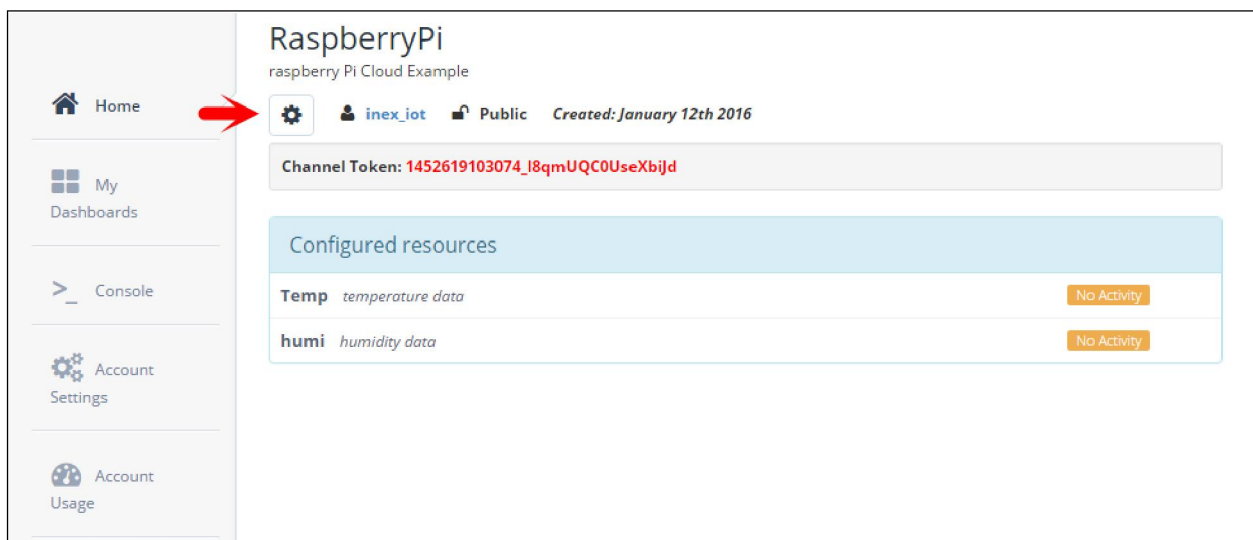
รูปที่ 13-14 เว็บเพจ **My Channels** แสดงรายการช่องเก็บข้อมูลที่สร้างขึ้นใหม่

(13.6.1.4) คลิกที่ชื่อช่องเก็บข้อมูล RaspberryPi ดังรูปที่ 13-15 เพื่อเข้าไปดูรายละเอียดต่างๆ



รูปที่ 13-15 การเลือกเข้าไปดูรายละเอียดของช่องเก็บข้อมูลที่สร้างขึ้น

(13.6.1.5) จะปรากฏรายละเอียดต่างๆ ดังรูปที่ 13-16 ผู้ใช้งานสามารถแก้ไข ลดหรือเพิ่มส่วนประกอบได้ตามต้องการ โดยคลิกที่ปุ่มรูปเฟือง จากรูปที่ 13-16 จะเห็นได้ว่า ไม่มีการส่งข้อมูลมายังคลาวด์เซิร์ฟเวอร์ โดยสังเกตได้จากส่วนประกอบทั้งสองตัวของช่องเก็บข้อมูล มีข้อความแสดงสถานะเป็น **No Activity**

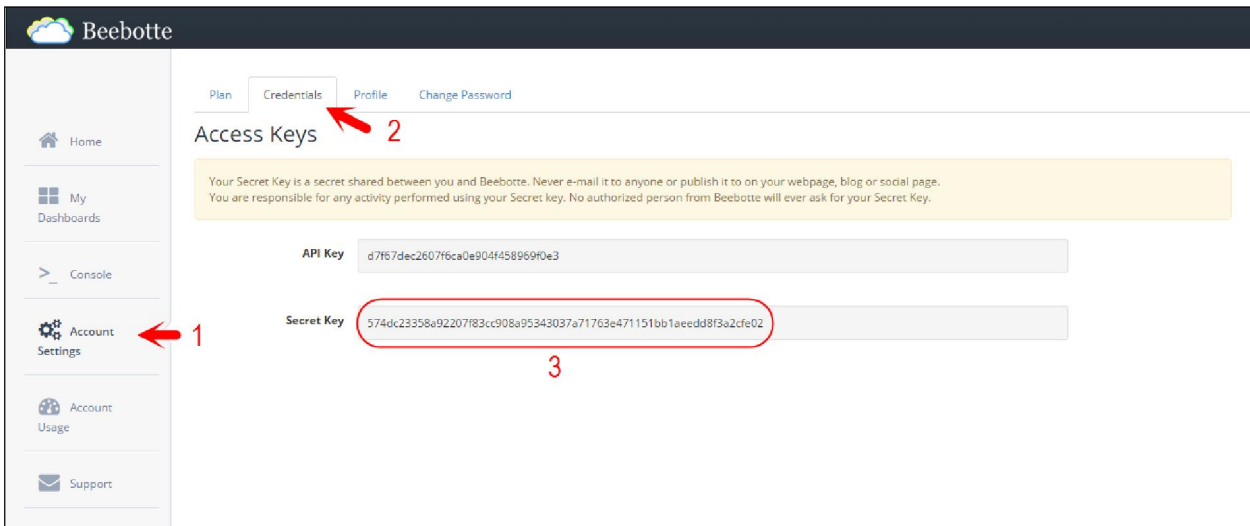


รูปที่ 13-16 แสดงรายละเอียดของช่องเก็บข้อมูลที่สร้างขึ้น และตำแหน่งของปุ่มสำหรับแก้ไข

13.6.2 ทดสอบและตรวจสอบข้อมูลที่ส่งมายัง Beebotte

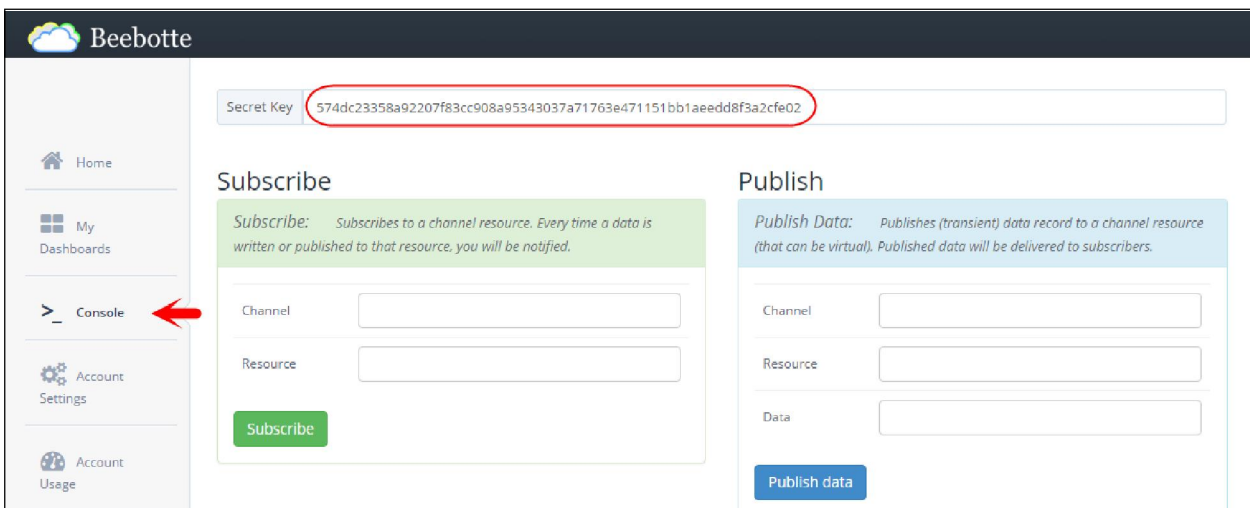
Beebotte มีเว็บเพจสำหรับใช้ในการทดสอบส่งข้อมูลเข้ามายังช่องเก็บข้อมูลที่ผู้ใช้งานสร้างขึ้น และสามารถตรวจสอบข้อมูลที่ส่งเข้ามาได้ มีขั้นตอนการใช้งานดังนี้

(13.6.2.1) สังเกตที่แถบเมนู เลือก **Account Setting** ตามลูกศร 1 จากนั้นคลิกที่ **Credentials** (ลูกศร 2) แล้วคัดลอก **Secret Key** (ลูกศร 3) เก็บไว้ก่อน เพื่อนำไปใช้ในขั้นตอนต่อไป ดังรูปที่ 13-17



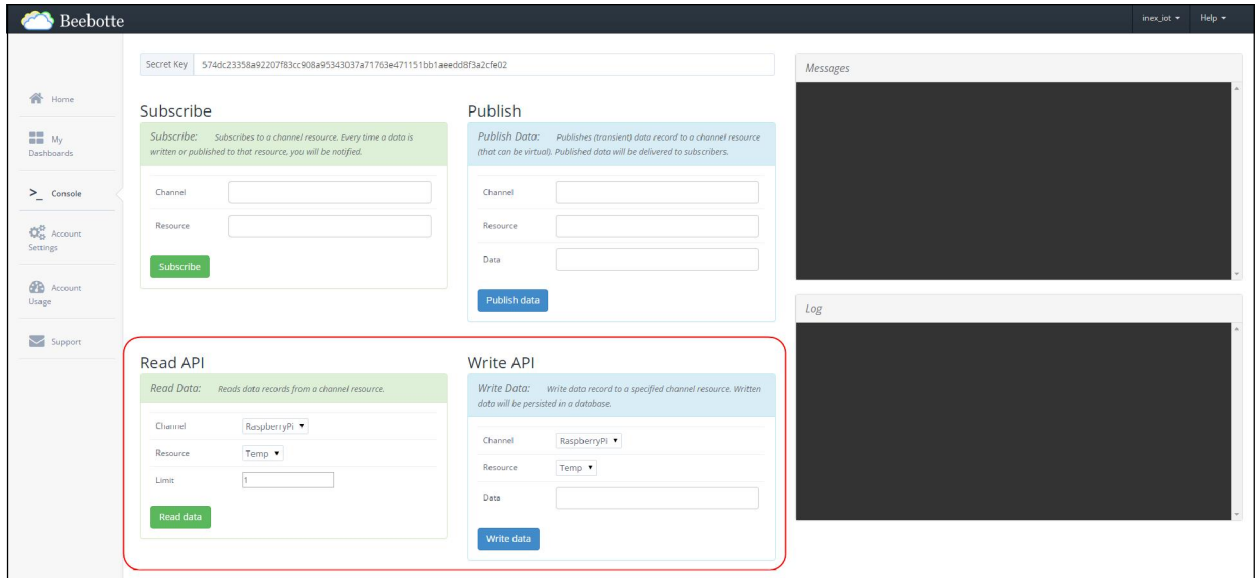
รูปที่ 13-17 แสดงขั้นตอนการเริ่มต้นทดสอบการทำงานของช่องเก็บข้อมูลที่สร้างขึ้นใหม่ โดยเลือกที่ **Account Setting** ที่แถบเมนูด้านข้าง

(13.6.2.2) ที่เมนูทางซ้าย เลือกรายการ **Console** นำ **Secret Key** ที่คัดลอกไว้จากขั้นตอนที่ (1) ใสลงในช่อง **Secret Key** ดังรูปที่ 13-18



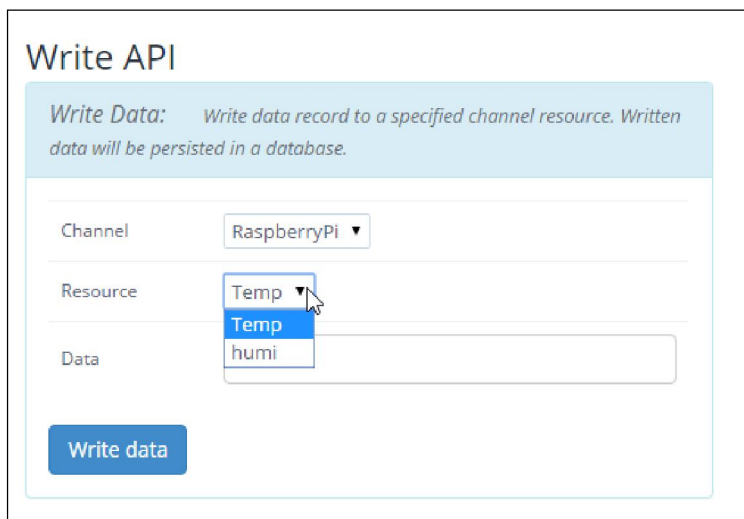
รูปที่ 13-18 แสดงการป้อนรหัส **Secret Key**

(13.6.2.3) ระบบจะไปยังเว็บเพจของการตั้งค่า Subscribe, Publish, Read API, Write API ที่ใช้ในการทดสอบ โดยในที่นี้จะยกตัวอย่างเพียง **Read API** และ **Write API** ก่อนตามรูปที่ 13-19 เพื่อให้สอดคล้องกับการเขียนไฟล์สคริปต์ Python บน Raspberry Pi 2 ซึ่งจะได้อธิบายต่อไป



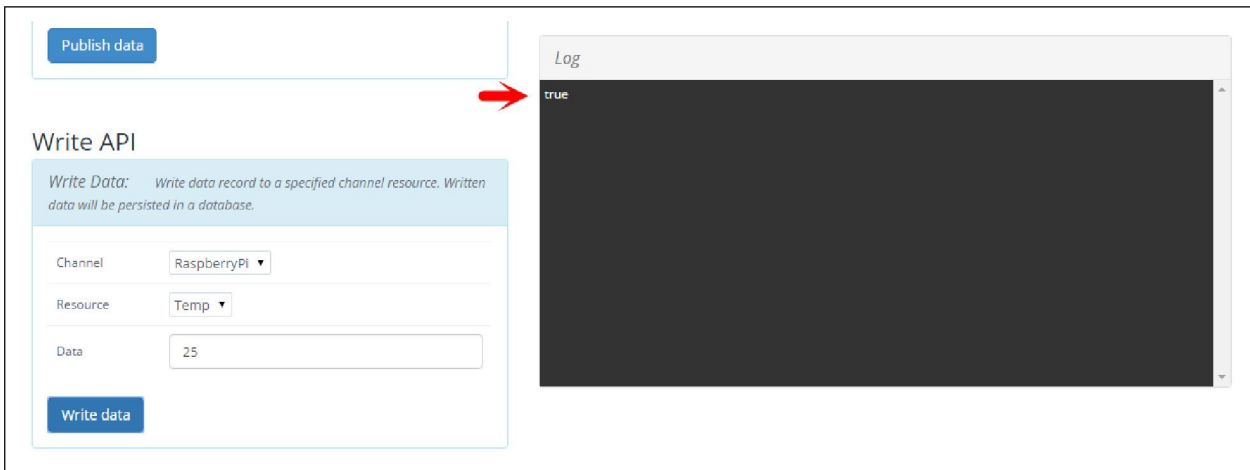
รูปที่ 13-19 เว็บเพจสำหรับกำหนดค่าที่เกี่ยวข้องกับการเขียนโปรแกรมติดต่อกับอุปกรณ์ของ Beebotte โดยเลือกการติดต่อผ่าน API

(13.6.2.4) ทดสอบการส่งข้อมูลมายังช่องเก็บข้อมูล สังเกตที่กรอบ **Write API** จะเห็นว่า มีเพียงตัวเลือกเดียวในขณะนี้คือ **RaspberryPi** ถ้าหากผู้ใช้งานสร้างไว้หลายช่อง ตัวเลือกก็จะมากตามไปด้วย ในขณะที่ส่วนประกอบหรือ Resource จะปรากฏให้เลือกตามจำนวนที่ได้สร้างไว้แล้ว ดังรูปที่ 13-20



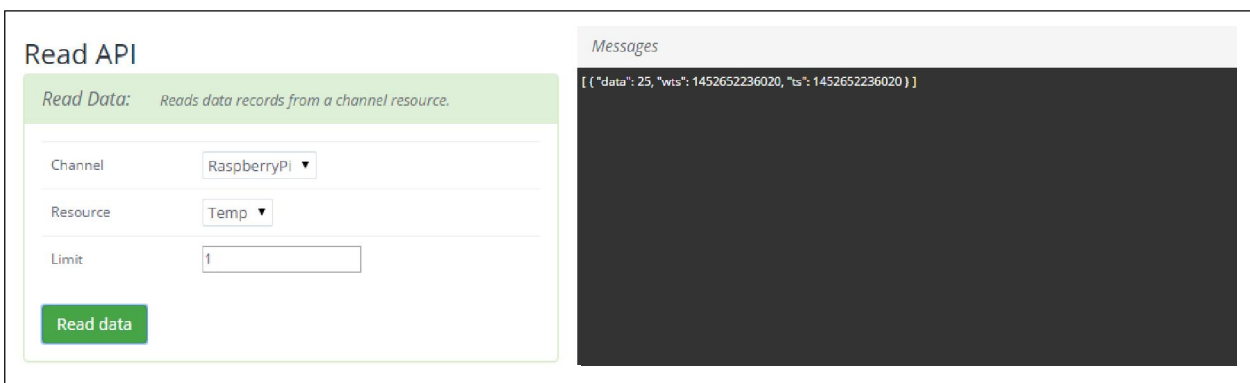
รูปที่ 13-20 แสดงรายละเอียดของช่องเก็บข้อมูลในส่วนของ Write API

(13.6.2.5) หากต้องการส่งค่าอุณหภูมิไปยังส่วนประกอบที่ชื่อว่า **Temp** ให้เลือกรายการของ **Resource** เป็น **Temp** แล้วกำหนดค่าที่ช่อง **Data** เป็น **25** ดังรูปที่ 13-21 จากนั้นคลิกปุ่ม **Write data** สังเกตที่หน้าต่าง **Log** จะมีข้อความว่า **true** นั้นหมายความว่า เกิดการส่งข้อมูลได้ถูกต้อง ดังแสดงในรูปที่ 13-21 ทางขวามือ



รูปที่ 13-21 การกำหนดเพื่อทดสอบเขียนข้อมูลไปยังช่องเก็บข้อมูลด้วยการใช้ Write API

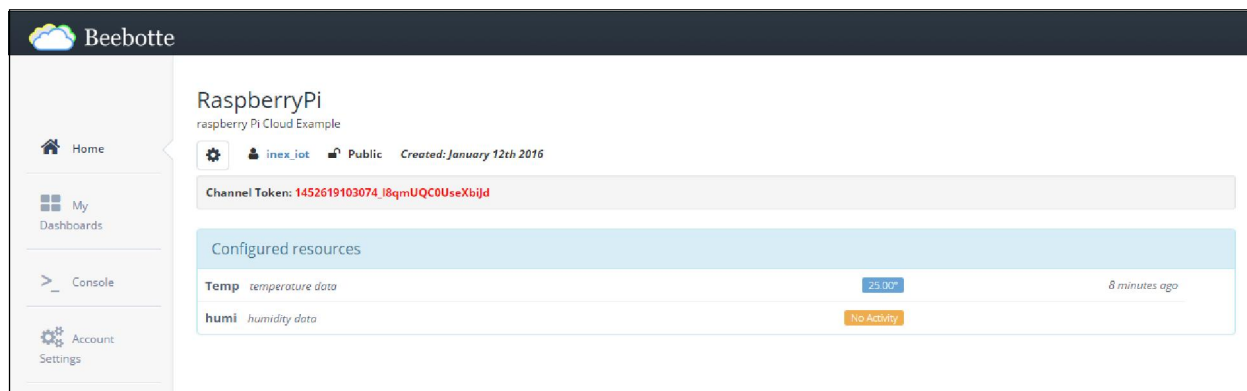
(13.6.2.6) หากต้องการทดสอบอ่านค่าจากช่องเก็บข้อมูลหรือ Channel ให้กำหนดและดูการทำงานที่กรอบ **Read API** หากต้องการอ่านค่าล่าสุดจาก Resource ที่ชื่อว่า **Temp** ดังรูปที่ 13-22 (ก) ที่ช่อง **Limit** ให้ใส่เลข **1** เพื่อเลือกอ่านข้อมูลล่าสุด และแสดงข้อมูลตอบกลับที่ช่อง **Messages** ดังรูปที่ 13-22 (ข) ข้อความที่ตอบกลับจะอยู่ในรูปแบบ json แต่ถ้าเลือกใส่เลข **5** ที่ช่อง **Limit** จะ ได้ข้อมูล 5 ชุดสุดท้ายที่ส่งเข้าไปยังส่วนประกอบหรือ Resource ตัวนี้



รูปที่ 13-22 การกำหนดค่าในกรอบ Read API เพื่อทดสอบอ่านค่าจากส่วนประกอบ Temp ของช่องเก็บข้อมูล RaspberryPi

- (ก) ตัวอย่างการกำหนดค่าเพื่อทดสอบการอ่าน
- (ข) ผลการทำงานที่ได้

(13.6.2.7) การตรวจสอบสถานะการส่งข้อมูลไปยังช่องเก็บข้อมูลหรือ Channel ที่ชื่อ **RaspberryPi** แสดงได้ดังรูปที่ 13-23 สังเกตที่ Resource ชื่อ **Temp** จะมีค่าอุณหภูมิที่ได้ทำการทดสอบส่งค่าจากขั้นตอนที่ (5) แสดงขึ้นมาแทนข้อความ **No Activity**

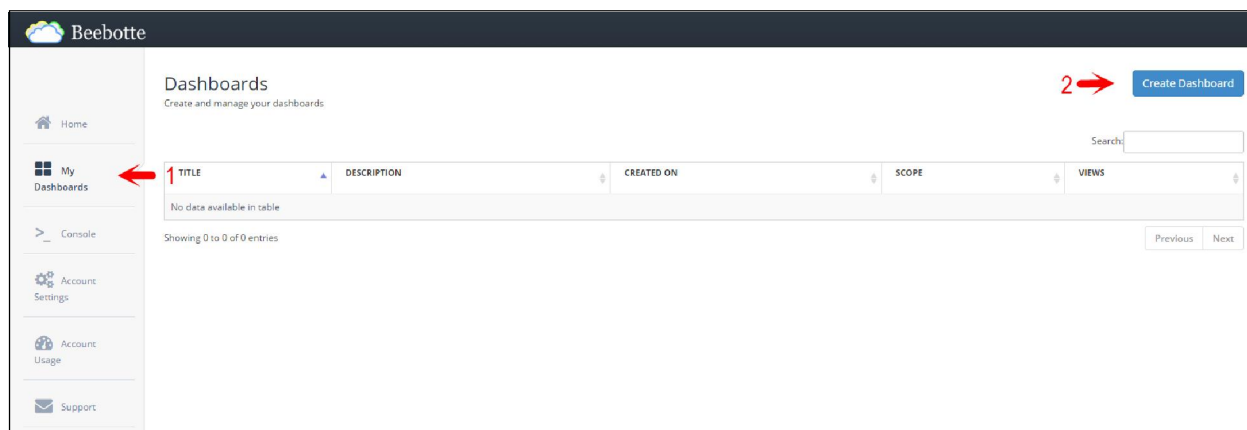


รูปที่ 13-23 เว็บเพจแสดงการสถานะการส่งข้อมูลมายังช่องเก็บข้อมูลหรือ Channel ที่ชื่อ **RaspberryPi**

13.7 การสร้างแดชบอร์ดหรือหน้าปัดแสดงผล

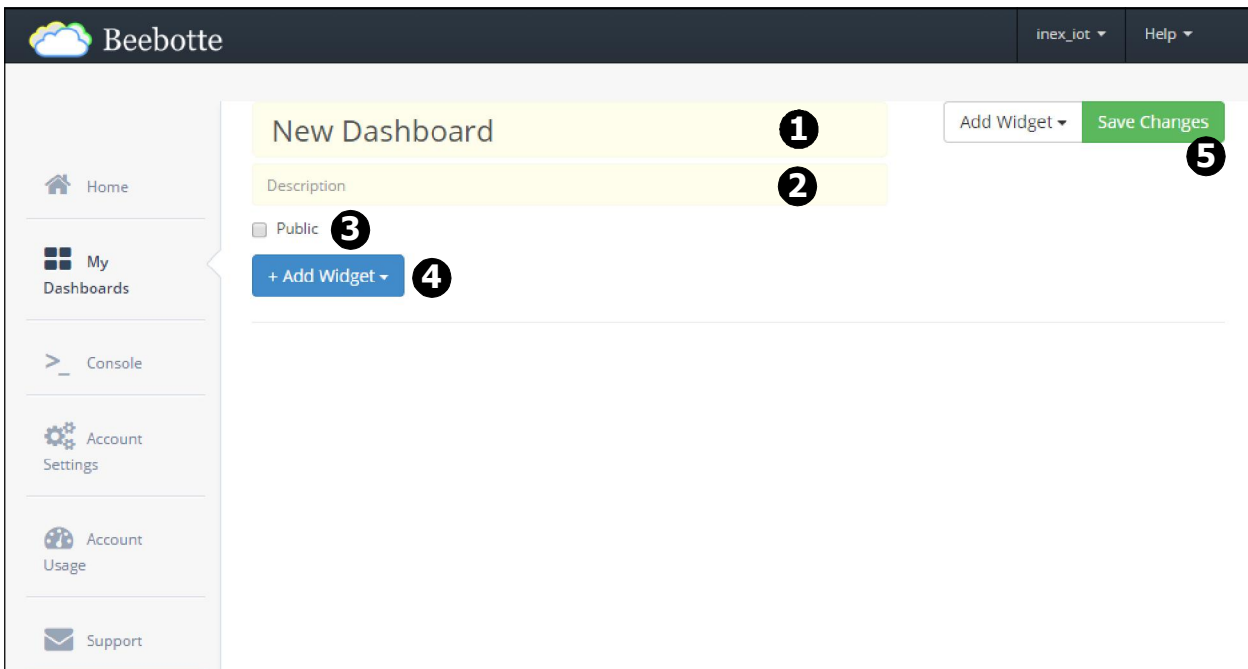
จากหัวข้อ 13.6 นำเสนอตัวอย่างการออกแบบและทดสอบคลาวด์เซิร์ฟเวอร์ที่ใช้เก็บและแสดงค่าความชื้นสัมพัทธ์และอุณหภูมิ ซึ่งเพียงพอแล้วสำหรับการทำงานของอุปกรณ์ IoT ทว่าในความเป็นจริง การแสดงผลของข้อมูลในรูปแบบหน้าปัดหรือแดชบอร์ด (dashboard) เป็นสิ่งที่ช่วยให้การดูค่าหรือการควบคุมผ่านระบบคลาวด์เซิร์ฟเวอร์มีความน่าสนใจและใช้งานง่ายขึ้น Beebotte เองก็มีความสามารถในการสร้างหน้าปัดแสดงผลหรือแดชบอร์ดได้ ในหัวข้อนี้จึงนำเสนอตัวอย่างการสร้างแดชบอร์ดอย่างง่ายจากเครื่องมือที่ทาง Beebotte สนับสนุน

(13.7.1) ที่แถบเมนูทางซ้ายเลือก **My Dashboards** (ลูกศร 1) จากนั้นคลิก **Create Dashboard** (ลูกศร 2) ดังรูปที่ 13-24



รูปที่ 13-24 เริ่มต้นสร้างแดชบอร์ดของ **Beebotte**

(13.7.2) จะปรากฏเว็บเพจสำหรับออกแบบแดชบอร์ดดังรูปที่ 13-25

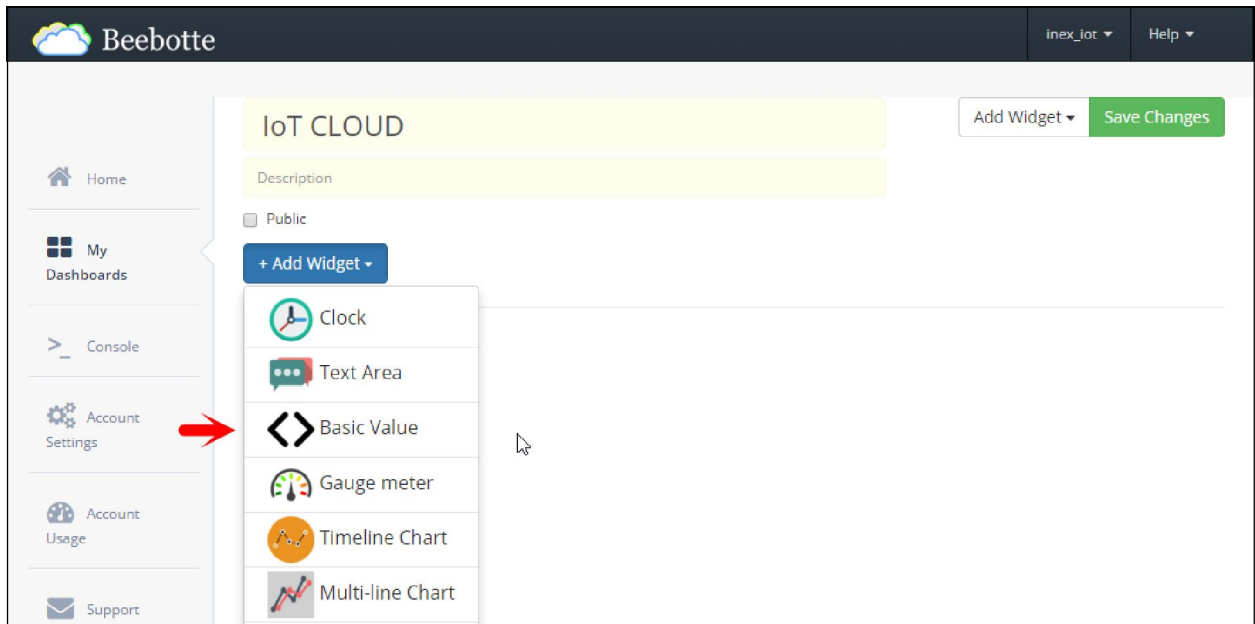


รูปที่ 13-25 เว็บเพจสำหรับสร้างแดชบอร์ดของ Beebotte

- ❶ กำหนดชื่อแดชบอร์ด
- ❷ กำหนดคำอธิบายของแดชบอร์ด
- ❸ เลือกความเป็นส่วนตัวของแดชบอร์ด หากคลิกที่ช่อง Public จะเป็นการเลือกให้เผยแพร่แดชบอร์ดนี้สู่สาธารณะ
- ❹ เลือกเครื่องมือสำหรับใช้ในการแสดงผลหรือวิดเจ็ตบนแดชบอร์ด
- ❺ ปุ่มบันทึกแดชบอร์ดที่สร้างขึ้น

(13.7.3) ตัวอย่างแดชบอร์ดที่แนะนำให้สร้างในหัวข้อนี้จะสัมพันธ์กับการทำงานของช่องเก็บข้อมูลในหัวข้อ 13.6 โดยกำหนดชื่อแดชบอร์ดว่า **IoT CLOUD** และเลือกเครื่องมือหรือวิดเจ็ต (widget) สำหรับแสดงค่าอุณหภูมิและความชื้นในรูปแบบ Basic Value, Gauge meter และ Multi-line Chart เริ่มจากกำหนดชื่อแดชบอร์ดโดยเปลี่ยนจากข้อความ **New Dashboard** เป็น **IoT CLOUD**

(13.7.4) เพิ่ม **Basic Value** โดยคลิกที่ **+Add Widget** จะปรากฏรายการให้เลือก ทำการเลือก **Basic Value** ดังรูปที่ 13-26

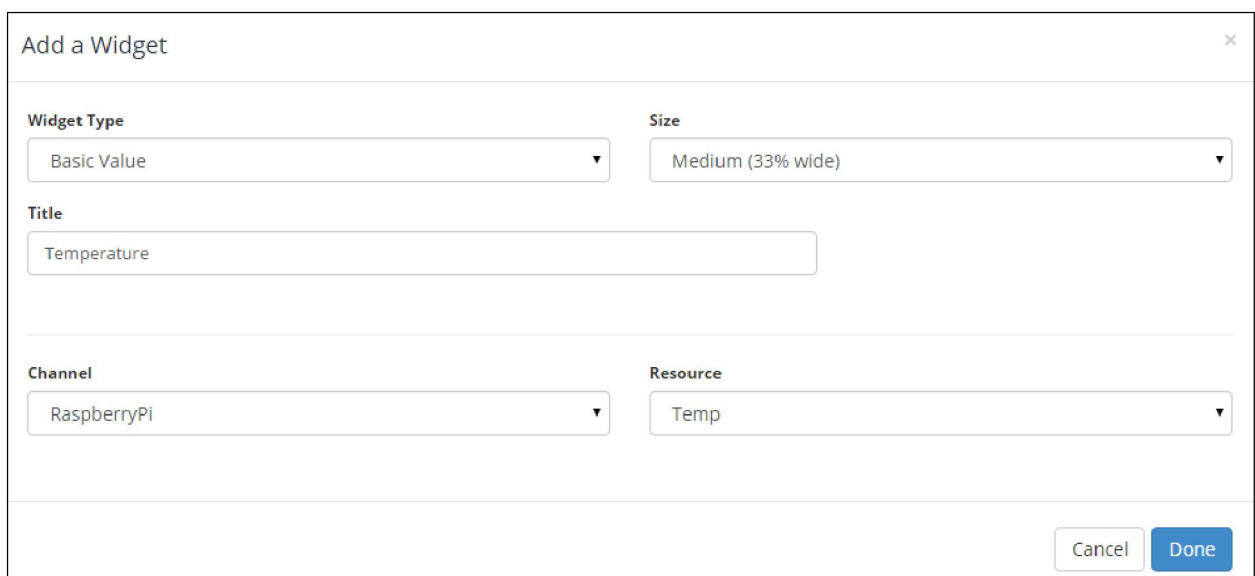


รูปที่ 13-26 เลือกเครื่องมือแสดงผล **Basic Value**

(13.7.5) จะปรากฏหน้าต่างการตั้งค่าต่างๆ ของวิดเจ็ต **Basic Value** ดังรูปที่ 13-27 กำหนดค่าดังนี้

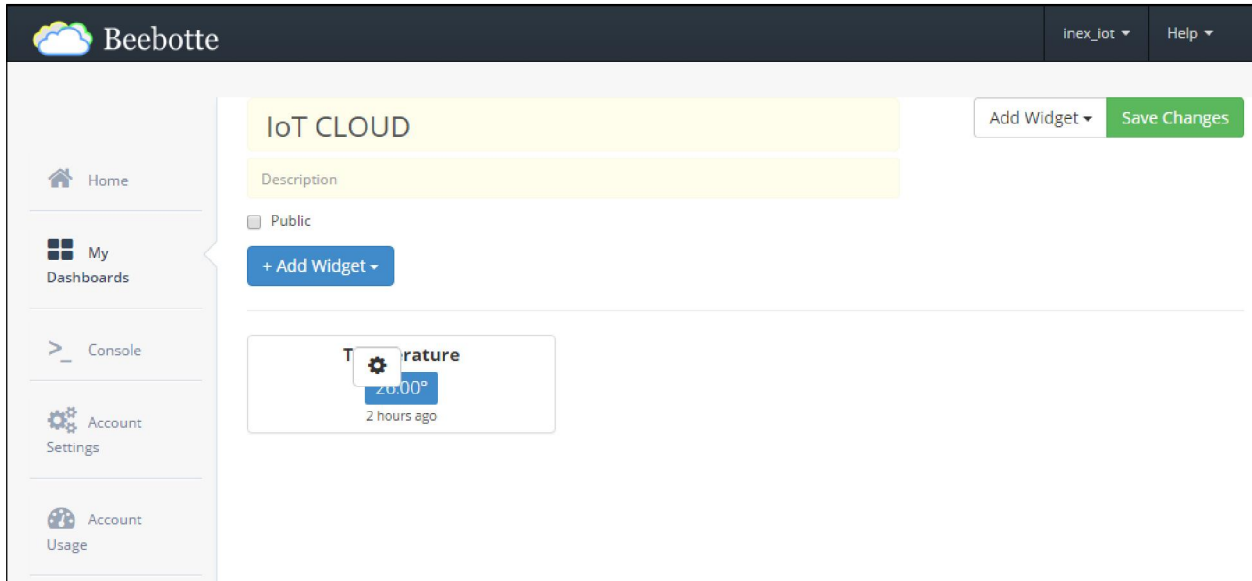
- **Title** กำหนดเป็น *Temperature*
- **Channel** กำหนดเป็น *RaspberryPi*
- **Resource** เป็น *Temp*

จากนั้นคลิกปุ่ม **Done** เพื่อยืนยันการตั้งค่า



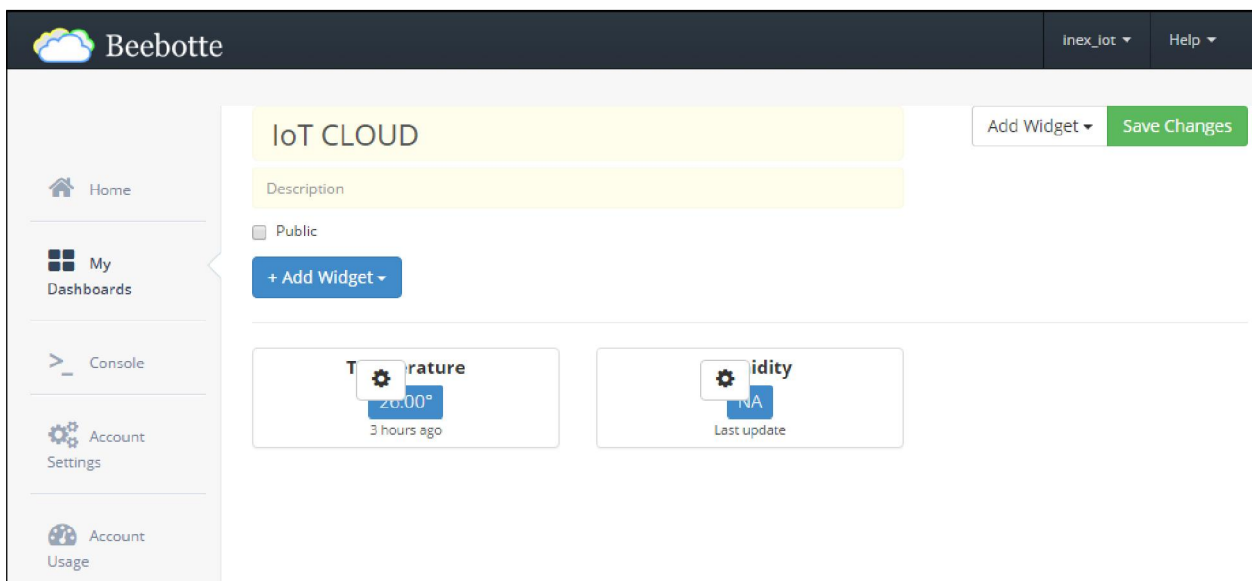
รูปที่ 13-27 การตั้งค่าวิดเจ็ต **Basic Value**

(13.7.6) จะปรากฏการแสดงผลดังรูปที่ 13-28 มีรูปเฟืองบ่งชี้ว่าแก้ไขได้ ถ้าต้องการแก้ไขใหม่ ให้คลิกที่รูปเฟือง จะมีเมนูให้เลือกคือ **Edit** หากต้องการแก้ไข และ **Delete** หากต้องการลบ



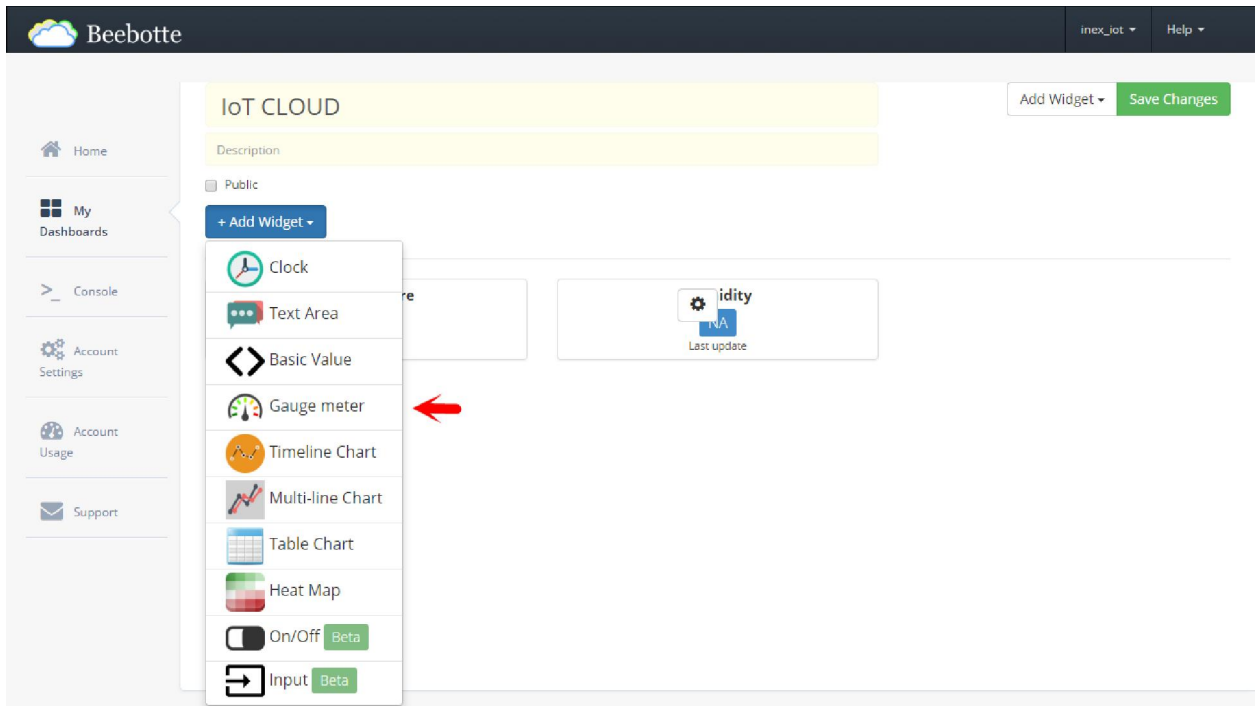
รูปที่ 13-28 แดชบอร์ดที่เริ่มต้นวางเครื่องมือ

(13.7.7) เพิ่มวิดเจ็ตแบบ **Basic Value** อีกหนึ่งตัว เพื่อแสดงค่าความชื้นสัมพัทธ์ ขั้นตอนการเพิ่มเหมือนขั้นตอนที่ (13.7.4) และ (13.7.5) ของหัวข้อนี้ เปลี่ยนชื่อเป็น **Humidity** เลือก Resource เป็น **humi** เมื่อคลิก **Done** จะได้ผลการออกแบบดังรูปที่ 13-29



รูปที่ 13-29 แดชบอร์ดที่มีการเพิ่มเครื่องมือ

(13.7.8) เพิ่มวิดเจ็ต **Gauge meter** โดยคลิกที่ **+Add Widget** เลือก **Gauge meter** ดังรูปที่ 13-30

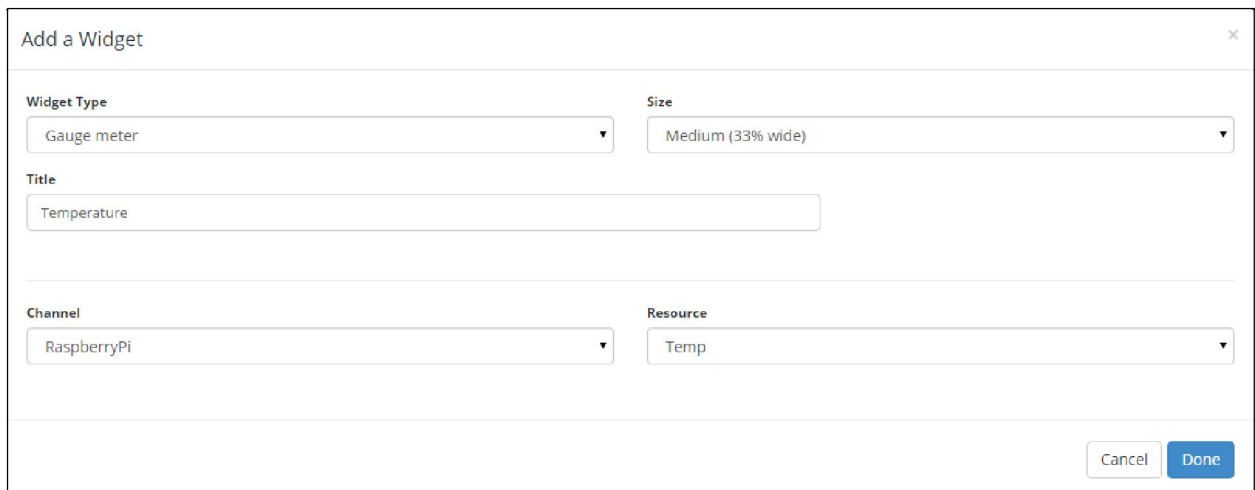


รูปที่ 13-30 การเพิ่มวิดเจ็ต **Gauge meter**

(13.7.9) หน้าต่างตั้งค่าการใช้งาน **Gauge meter** ปรากฏขึ้น ให้ทำการตั้งค่าดังนี้

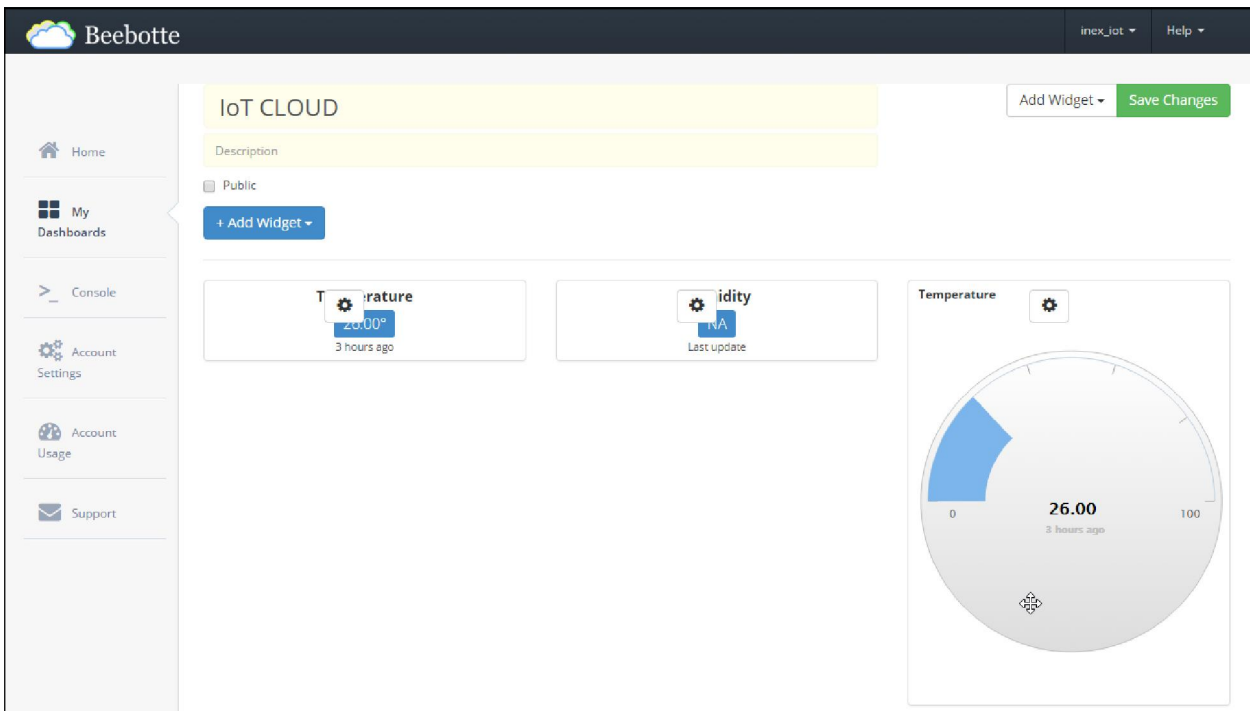
- **Title** กำหนดชื่อเป็น *Temperature* เพื่อแสดงค่าอุณหภูมิ
- **Channel** กำหนดเป็น *RaspberryPi*
- **Resource** กำหนดเป็น *Temp*

จากนั้นคลิกปุ่ม **Done** เพื่อยืนยันการตั้งค่า ดังแสดงการตั้งค่าในรูปที่ 13-31



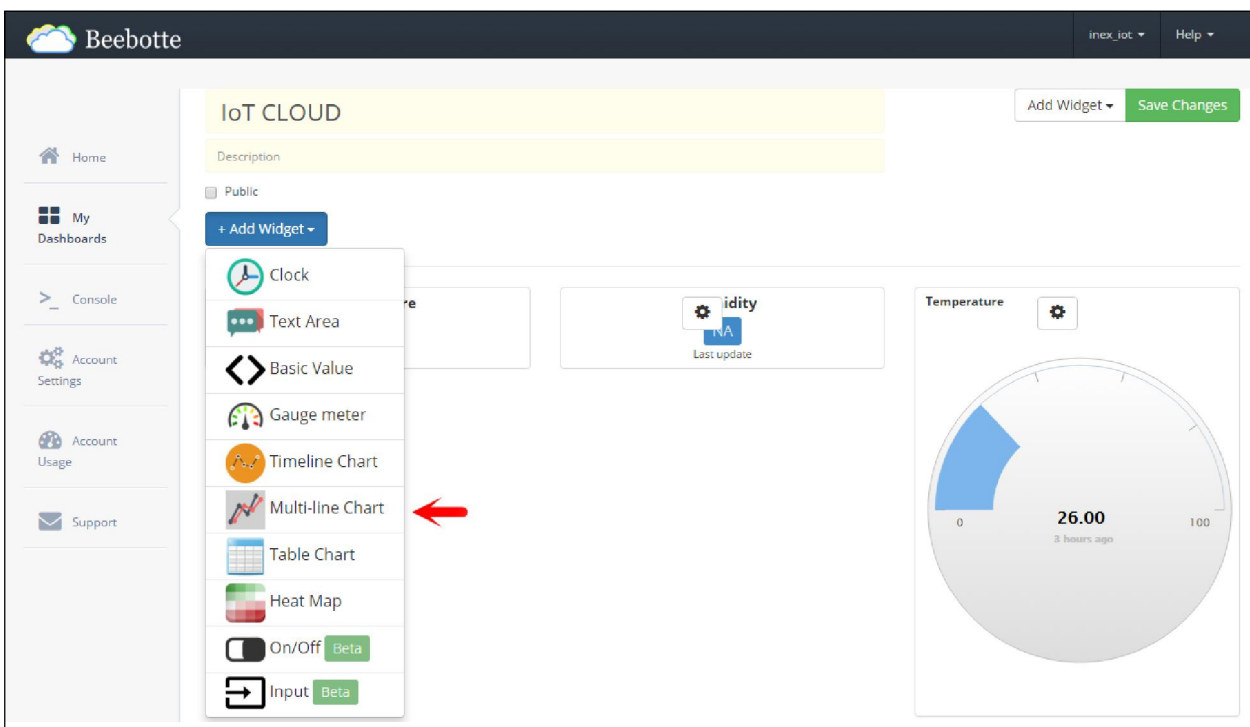
รูปที่ 13-31 ตั้งค่าการใช้งานวิดเจ็ต **Guage meter**

(13.7.10) ที่เว็บเพจ IoT CLOUD จะแสดง Gauge meter สำหรับแสดงค่าอุณหภูมิดังรูปที่ 13-32



รูปที่ 13-32 แสดง Gauge meter สำหรับแสดงค่าอุณหภูมิ

(13.7.11) เพิ่มวิดเจ็ต Multi-line Chart โดยคลิกที่ +Add Widget แล้วเลือก Multi-line Chart ดังรูปที่ 13-33



รูปที่ 13-33 แสดงการเลือกวิดเจ็ต Multi-line Chart

(13.7.12) หน้าต่างตั้งค่าการใช้งาน **Multi-line Chart** ปรากฏขึ้น ทำการตั้งค่าดังนี้

- **Title** กำหนดชื่อเป็น *Temperature & Humidity*
- **Channel** กำหนดเป็น *RaspberryPi*
- **Resource** กำหนดเป็น *Temp*
- **Label** ใช้สำหรับบอกชื่อเส้นกราฟ

ทั้งหมดเป็นการตั้งค่าสำหรับส่วนประกอบตัวแรกที่ใช้แสดงค่าอุณหภูมิ ดังรูปที่ 13-34

The screenshot shows a dialog box titled "Add a Multi-line Widget". It has several input fields and buttons. The "title" field contains "Temperature & Humidity". The "Size" dropdown is set to "Large (50% wide)". Below these, there are four columns: "Channel" with a dropdown set to "RaspberryPi", "Resource" with a dropdown set to "Temp", "Label" with a text field containing "Temperature", and "Color" with a blue button labeled "7CB5EC" and a "Remove" button with an "X". At the bottom left is a "+ Resource" button, and at the bottom right are "Cancel" and "Done" buttons.

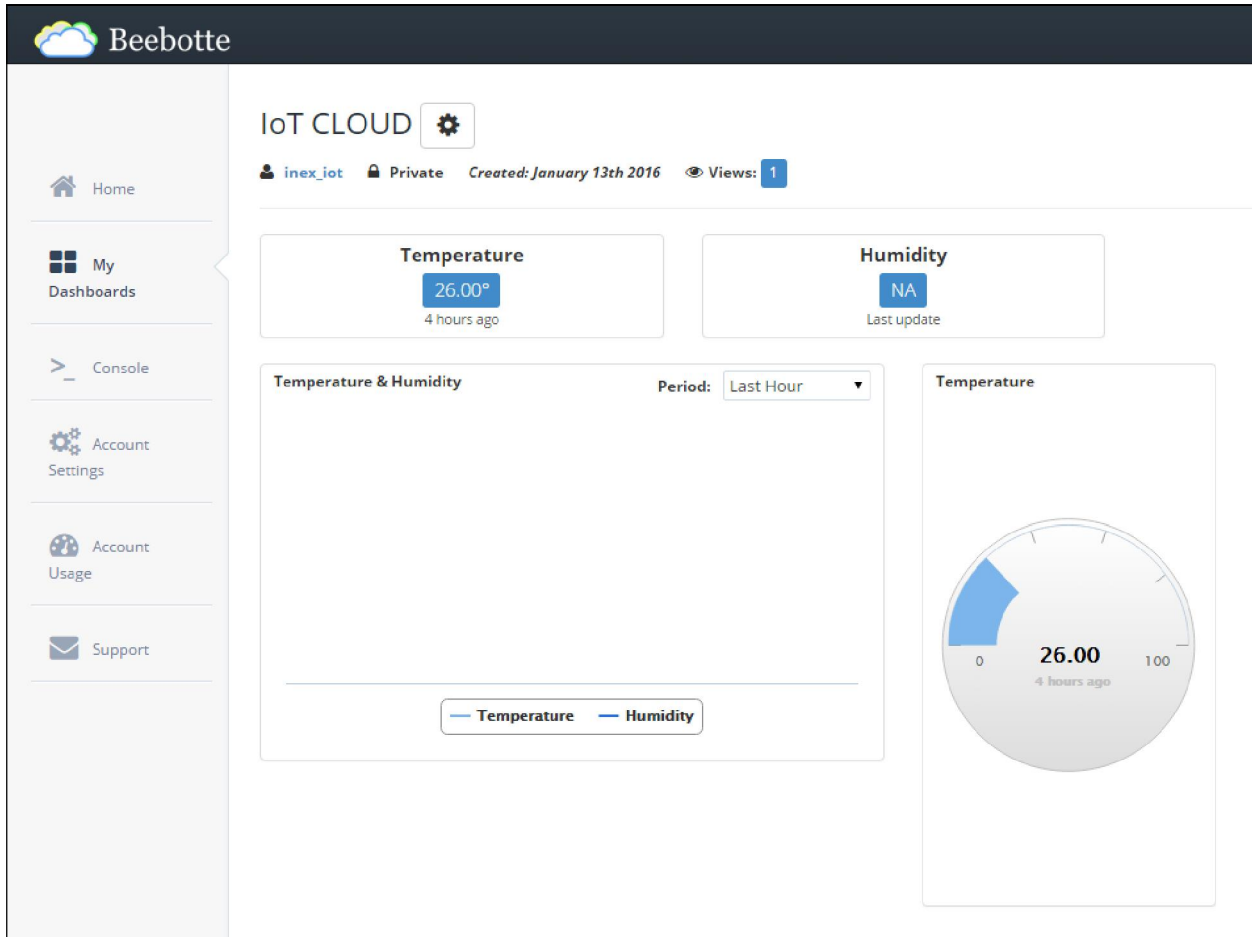
รูปที่ 13-34 ตั้งค่าการใช้งานวิดเจ็ต **Multi-line Chart**

(13.7.13) จากนั้นคลิกที่ปุ่ม **+ Resource** เพื่อเพิ่มเส้นกราฟสำหรับแสดงค่าความชื้นสัมพัทธ์แล้ว ตั้งค่าดังรูปที่ 13-35 ผู้ใช้งานสามารถเลือกเปลี่ยนสีเส้นกราฟได้ โดยคลิกที่ช่อง **Color** เมื่อตั้งค่าเรียบร้อยแล้ว คลิกปุ่ม **Done** เพื่อบันทึกการตั้งค่า

This screenshot shows the same dialog box as Figure 13-34, but with an additional entry. The second entry has "Channel" set to "RaspberryPi", "Resource" set to "humi", and "Label" set to "Humidity". Its "Color" field is "266EEC" and has a color picker open over it. The color picker shows a rainbow spectrum with a white crosshair. The "Remove" button for this entry also has an "X". The "+ Resource" button is still visible at the bottom left, and "Cancel" and "Done" buttons are at the bottom right.

รูปที่ 13-35 แสดงการเลือกสีเส้นกราฟของ **Multi-line chart**

(13.7.14) ที่เว็บเพจ **IoT CLOUD** จะแสดงวิดเจ็ต **Multi-line Chart** คลิกปุ่ม **Save Changes** แดชบอร์ด **IoT CLOUD** พร้อมสำหรับการแสดงผลข้อมูลดังรูปที่ 13-36



รูปที่ 13-36 แดชบอร์ด IoT CLOUD ที่พร้อมรับข้อมูลเพื่อแสดงผล

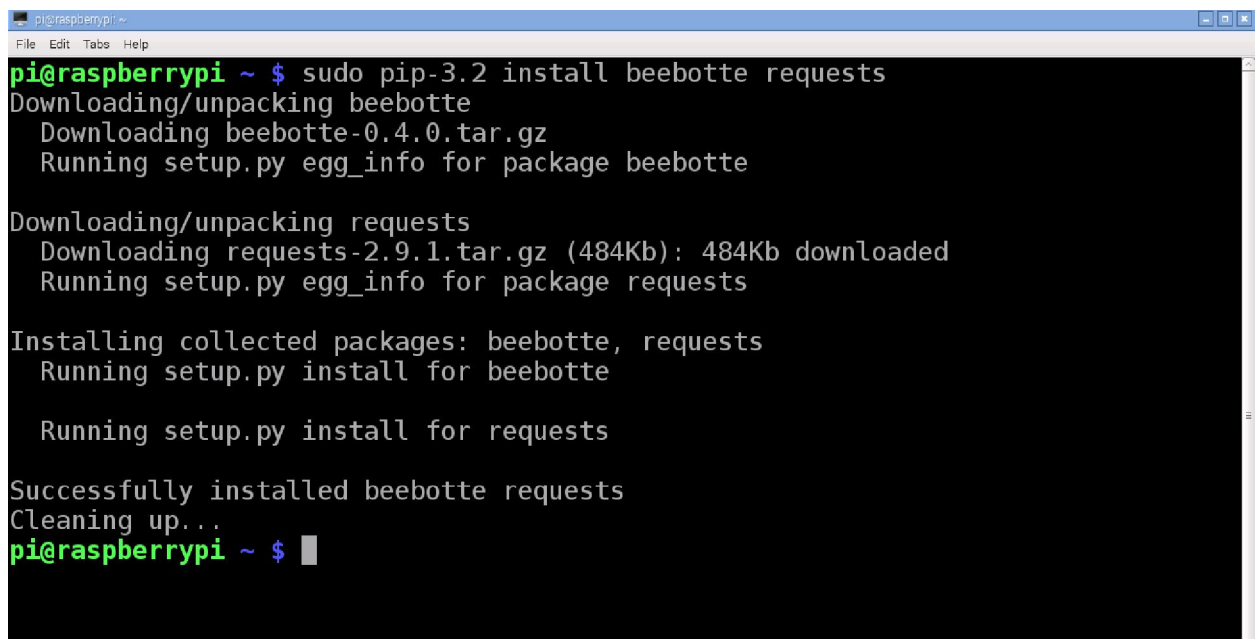
13.8 ติดตั้งไลบรารี beebotte บน Python

หลังจากทราบถึงการตั้งค่าคลาวด์เซิร์ฟเวอร์ และขั้นตอนการสร้างแดชบอร์ดเพื่อแสดงผลกลับมาฝั่งของบอร์ด Raspberry Pi 2 จะต้องมีการเขียนโปรแกรมเป็นไฟล์สคริปต์เพื่ออ่านค่าจากตัวตรวจจับและส่งผ่านเครือข่ายอินเทอร์เน็ตไปยังคลาวด์เซิร์ฟเวอร์ Beebotte จึงต้องอาศัยไฟล์ไลบรารีของการติดต่อกับ beebotte ผ่าน API หรือผ่านโปรโตคอล MQTT

สำหรับในที่นี้ขอแนะนำการติดต่อด้วย **REST API** จึงต้องทำการติดตั้งไลบรารีก่อน โดยกำหนดให้ Raspberry Pi 2 ทำงานในโหมดคอมมานด์พรอมพ์ แล้วพิมพ์คำสั่ง

```
sudo pip-3.2 install beebotte requests
```

เพื่อติดตั้งไลบรารี beebotte ให้กับ Python รอจนกระทั่งการติดตั้งเสร็จสมบูรณ์ ดังรูปที่ 13-37



```
pi@raspberrypi ~ $ sudo pip-3.2 install beebotte requests
Downloading/unpacking beebotte
  Downloading beebotte-0.4.0.tar.gz
  Running setup.py egg_info for package beebotte

Downloading/unpacking requests
  Downloading requests-2.9.1.tar.gz (484Kb): 484Kb downloaded
  Running setup.py egg_info for package requests

Installing collected packages: beebotte, requests
  Running setup.py install for beebotte

  Running setup.py install for requests

Successfully installed beebotte requests
Cleaning up...
pi@raspberrypi ~ $
```

รูปที่ 13-37 หน้าต่างคอมมานด์ไลน์ของ Raspberry Pi 2 แสดงการติดตั้งไลบรารี beebotte

13.9 ตัวอย่างโปรแกรมทดสอบใช้งาน Beebotte กับ Raspberry Pi 2

ตัวอย่างที่นำเสนอในบทนี้คือ การควบคุมให้บอร์ด Raspberry Pi 2 อ่านค่าอุณหภูมิและความชื้นสัมพัทธ์จากโมดูลตรวจจับ ZX-DHT11 แล้วนำข้อมูลนั้นส่งไปยังคลาวด์เซิร์ฟเวอร์ Beebotte ด้วย REST API เพื่อบันทึกค่าและแสดงผล

ผู้พัฒนาควรทำการออกแบบคลาวด์เซิร์ฟเวอร์ตามขั้นตอนในหัวข้อ 13.6 โดยมีขั้นตอนหลัก ดังนี้

- (1) สร้างช่องเก็บข้อมูลหรือ Channel ชื่อ **RaspberryPi**
- (2) สร้างส่วนประกอบหรือ Resource ชื่อ **Temp** ชนิดข้อมูลเป็น **temperature** เพื่อเก็บค่าอุณหภูมิ
- (3) สร้างส่วนประกอบชื่อ **humi** ชนิดข้อมูลเป็น **humidity** เพื่อเก็บข้อมูลค่าความชื้นสัมพัทธ์

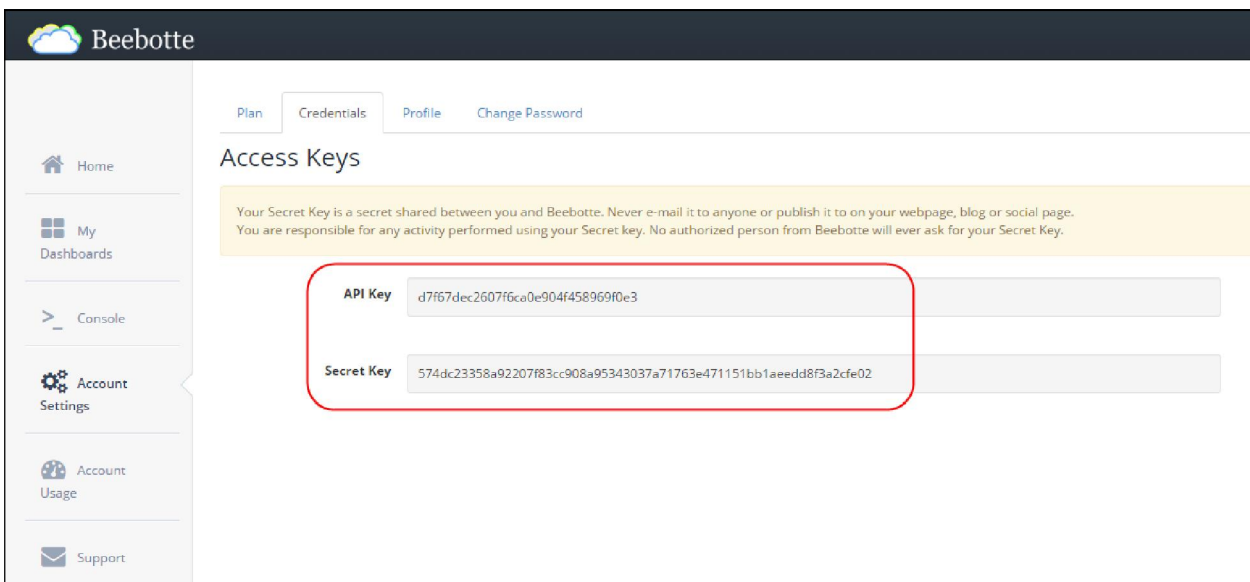
จากนั้นดำเนินการสร้างเว็บเพจของคลาวด์เซิร์ฟเวอร์เพื่อใช้ในการบันทึกและแสดงผล โดยมีขั้นตอนสำคัญดังนี้

(13.9.1) เตรียม **API Key** และ **Secret Key** ที่แถบเมนูทางซ้าย เลือก **Account Setting** จากนั้นคลิกที่ **Credentials** จะได้ผลดังรูปที่ 13-38 มีการกำหนดรหัสใช้งานไว้ 2 รหัสคือ

API Key = **d7f67dec2607f6ca0e904f458969f0e3**

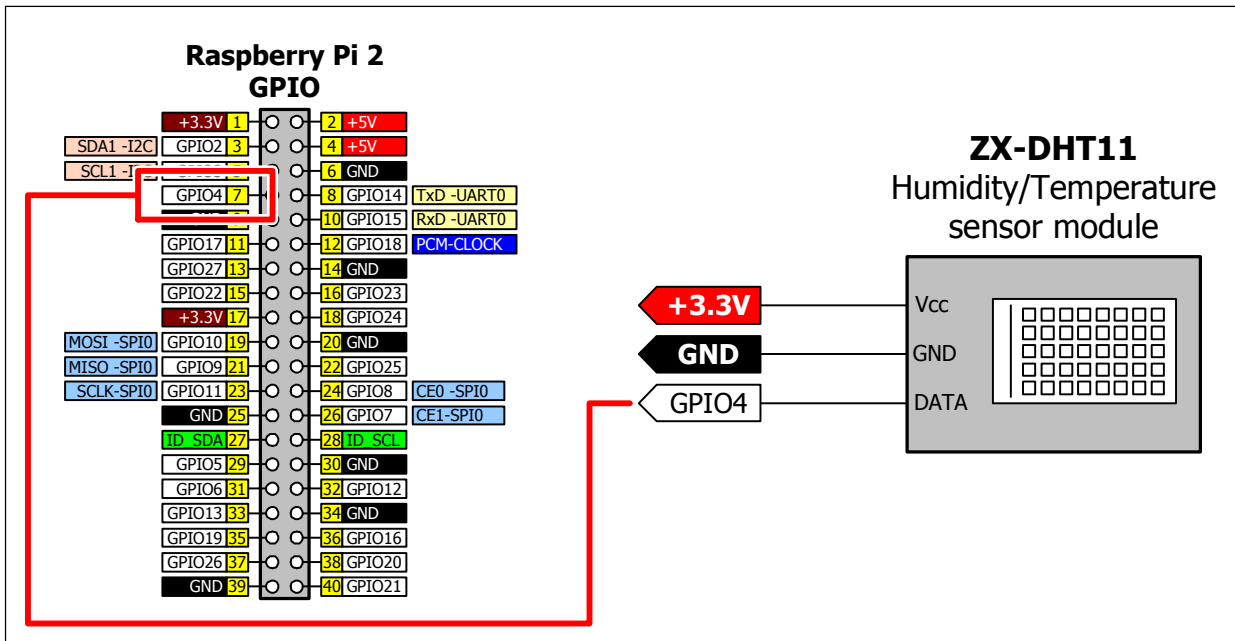
Secret Key = **574dc23358a92207f83cc908a95343037a71763e471151bb1aeedd8f3a2cfe02**

เมื่อได้ **API key** และ **Secret Key** แล้ว ทำการสร้างช่องเก็บข้อมูลและแดชบอร์ดเพื่อรองรับการติดต่อกับบอร์ด Raspberry Pi 2 ต่อไป



รูปที่ 13-38 เว็บเพจแสดงช่องสำหรับป้อนรหัสเพื่อเข้าถึงคลาวด์เซิร์ฟเวอร์ของ Beebotte

(13.9.2) ทำการต่อวงจรตามรูปที่ 13-39 เข้ากับพอร์ต GPIO ของบอร์ด Raspberry Pi 2



รูปที่ 13-39 วงจรเชื่อมต่อบอร์ด Raspberry Pi 2 กับโมดูลตรวจวัดความชื้นสัมพัทธ์และอุณหภูมิ ZX-DHT11 ผ่านพอร์ต GPIO

(13.9.3) เปิดโปรแกรม Geany เพื่อสร้างโปรแกรมของไฟล์สคริปต์ Python โดยพิมพ์โปรแกรมที่ 13-1

(13.9.4) สั่งให้ไฟล์สคริปต์ทำงาน จะได้ผลการทำงานแสดงที่หน้าต่างเทอร์มินอลดังรูปที่ 13-40

```

LXTerminal
File Edit Tabs Help
Temp=27.2*C Humidity=54.6%
Temp=27.0*C Humidity=52.9%
Temp=26.9*C Humidity=52.2%
Temp=26.9*C Humidity=52.2%
Temp=26.9*C Humidity=52.1%
    
```

รูปที่ 13-40 หน้าต่างเทอร์มินอลแสดงผลการทำงานของโปรแกรม cloud.py เป็นการแสดงค่าอุณหภูมิและความชื้นสัมพัทธ์ที่ Raspberry Pi 2 อ่านได้จากโมดูล ZX-DHT11

```
#!/usr/bin/env python
#####
# This code uses the Beebotte API, you must have an account.
# You can register here: http://beebotte.com/register
#####
import time
import json
import Adafruit_DHT
from beebotte import *

API_KEY = 'd7f67dec2607f6ca0e904f458969f0e3'
SECRET_KEY = '574dc23358a92207f83cc908a95343037a71763e471151bb1aeedd8f3a2cfe02'
### Replace API_KEY and SECRET_KEY with those of your account
bbt = BBT(API_KEY,SECRET_KEY)
period = 5 ## Sensor data reporting period (5 seconds)
pin = 4 ## Assuming the DHT11 sensor is connected to GPIO pin number 4
### Change channel name and resource names as suits you
temp_resource = Resource(bbt, 'RaspberryPi', 'Temp')
humid_resource = Resource(bbt, 'RaspberryPi', 'humi')

def run():
    while True:
        ### Assume
        humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, pin)
        if humidity is not None and temperature is not None:
            print('Temp={0:0.1f}*C Humidity={1:0.1f}%'.format(temperature,humidity))
            try:
                #Send temperature to Beebotte
                temp_resource.write(temperature)
                #Send humidity to Beebotte
                humid_resource.write(humidity)
            except Exception:
                ## Process exception here
                print ("Error while writing to Beebotte")
        else:
            print ("Failed to get reading. Try again!")
        #Sleep some time
        time.sleep(period)
run()
```

โปรแกรมที่ 13-1 ไฟล์ cloud.py โปรแกรมภาษา Python สำหรับบอร์ด Raspberry Pi 2 เพื่ออ่านค่าความชื้นสัมพัทธ์และอุณหภูมิจากโมดูล ZX-DHT11 นำไปบันทึกค่าและแสดงผลที่ช่องเก็บข้อมูล RaspberryPi บนคลาวด์เซิร์ฟเวอร์ Beebotte (มีต่อ)

คำอธิบายโปรแกรม

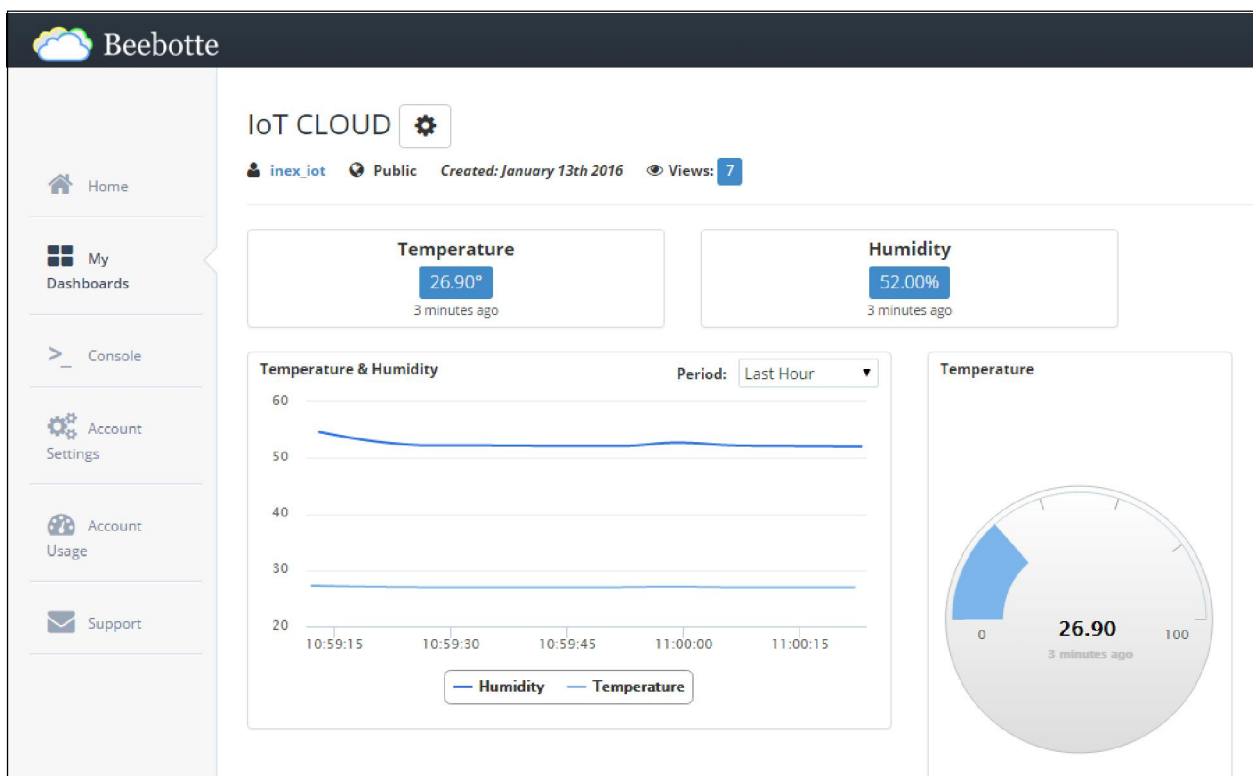
เมื่อสร้างช่องเก็บข้อมูลเพื่อรองรับข้อมูลที่จะส่งขึ้นไปแล้ว รหัส API Key และ Secret key จะเป็นตัวยืนยันตัวตนของอุปกรณ์เพื่อส่งข้อมูลไปยัง Beebotte ดังนั้นจึงต้องกำหนด API Key และ Secret key ไว้ในฟังก์ชัน BBT (API_KEY, SECRET_KEY) และสร้างตัวแปร bbt เพื่อรับออบเจกต์

จากนั้นสร้างออบเจกต์สำหรับส่งค่าไปยังแต่ละส่วนประกอบหรือ resource ด้วยฟังก์ชัน Resource (bbt, 'Name Channel', 'Name resource') ตามด้วยการสร้างตัวแปรมารองรับด้วย เช่น temp_resource = Resource (bbt, 'RaspberryPi', 'Temp')

การส่งค่าไปยังส่วนประกอบหรือ Resource จะใช้คำสั่ง temp_resource.write (temperature) ถ้าหากการส่งข้อมูลไม่สำเร็จ จะแจ้งที่หน้าต่างเทอร์มินอลด้วยคำสั่ง print ("Error while writing to Beebotte")

โปรแกรมที่ 13-1 ไฟล์ cloud.py โปรแกรมภาษา Python สำหรับบอร์ด Raspberry Pi 2 เพื่ออ่านค่าความชื้นสัมพัทธ์และอุณหภูมิจากโมดูล ZX-DHT11 นำไปบันทึกค่าและแสดงผลที่ช่องเก็บข้อมูล RaspberryPi บนคลาวด์เซิร์ฟเวอร์ Beebotte (จบ)

(13.9.5) ตรวจสอบการแสดงผลของแดชบอร์ดที่ออกแบบไว้บน Beebotte จะได้ผลการทำงานตามรูปที่ 13-41



รูปที่ 13-41 แสดงการแสดงผลของแดชบอร์ดของแดชบอร์ดของช่องเก็บข้อมูล RaspberryPi บนคลาวด์เซิร์ฟเวอร์ Beebotte ที่สร้างไว้ตั้งแต่ขั้นตอนในหัวข้อ 13.7

13.10 ตัวอย่างการควบคุม LED ด้วยแดชบอร์ดบน Beebotte

นับจากหัวข้อนี้เป็นการนำเสนอตัวอย่างโครงการเบื้องต้น เพื่อแสดงให้เห็นถึงการทำงานร่วมกันระหว่างบอร์ด Raspberry Pi 2 กับคลาวด์เซิร์ฟเวอร์ Beebotte ในการติดต่อและควบคุมอุปกรณ์อินพุตเอาต์พุตที่ต่อกับพอร์ต GPIO ของบอร์ด Raspberry Pi 2 เพื่อเป็นแนวทางในการพัฒนาอุปกรณ์ IoT อย่างง่ายด้วยบอร์ด Raspberry Pi 2

13.10.1 ภาพรวมของการพัฒนา

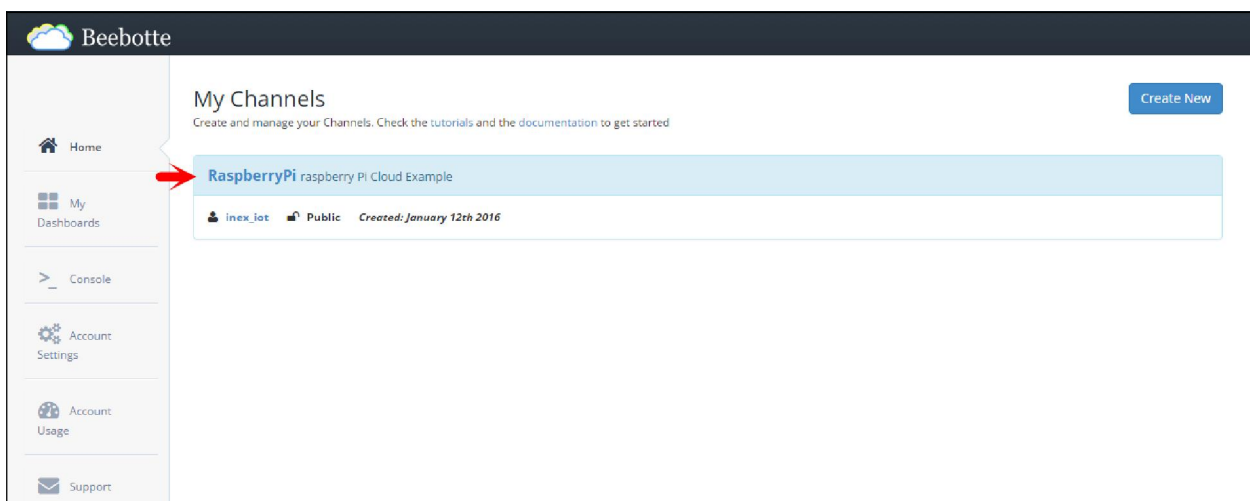
เริ่มต้นด้วยการออกแบบเว็บเพจของคลาวด์เซิร์ฟเวอร์ ให้มีการแสดงรูปสวิตช์สำหรับเปิดปิด LED โดยยังคงใช้ช่องเก็บข้อมูลหรือ Channel เดิมคือ **RaspberryPi** ทำการเพิ่มส่วนประกอบหรือ Resource ใหม่เข้าไป ตั้งชื่อว่า **StLED** จากนั้นสร้างแดชบอร์ด และเขียนโปรแกรมของไฟล์สคริปต์ Python ให้แก่บอร์ด Raspberry Pi 2 ต่อวงจรที่พอร์ต GPIO ของบอร์ด Raspberry Pi 2 แล้วทดสอบการทำงาน

13.10.2 เตรียมการคลาวด์เซิร์ฟเวอร์

มีขั้นตอนดังนี้

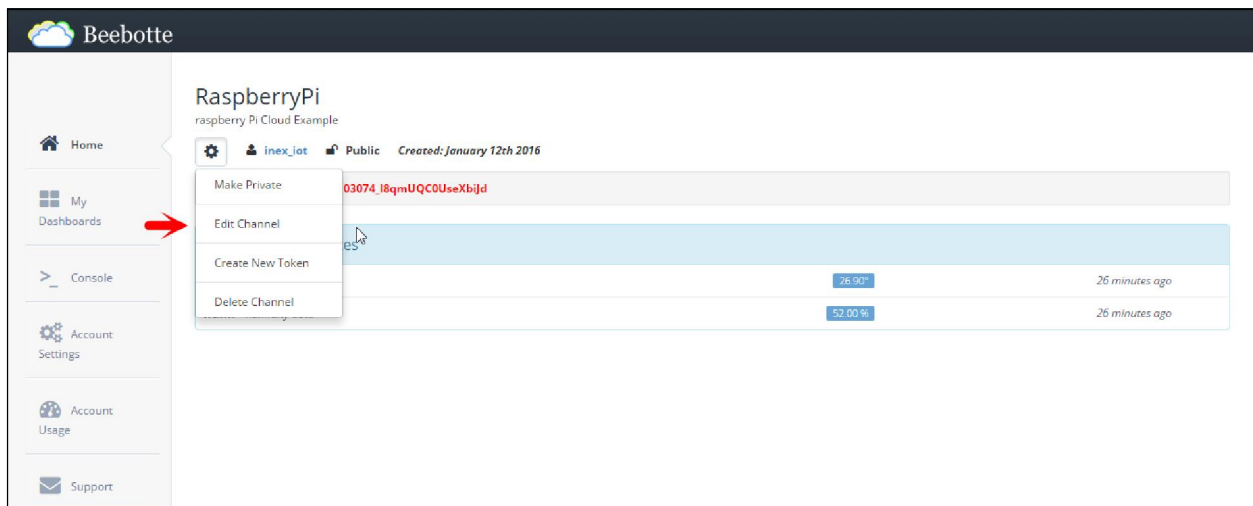
(13.10.2.1) ต่อบอร์ด Raspberry Pi 2 กับเมาส์, คีย์บอร์ด, จอภาพ และเชื่อมต่อเข้ากับเครือข่ายอินเทอร์เน็ต จากนั้นไปยังเว็บไซต์ของ Beebotte ทำการลงชื่อเข้าใช้งาน แล้วเลือกช่องเก็บข้อมูล RaspberryPi ที่เคยทำไว้จากขั้นตอนก่อนหน้านี้ อย่างไรก็ตาม ผู้พัฒนาสามารถสร้างช่องเก็บข้อมูลหรือ Channel ใหม่ได้ตามต้องการ

(13.10.2.2) ที่แถบเมนูทางซ้ายของ Beebotte คลิกที่ **Home** จากนั้นคลิกที่ข้อความ **RaspberryPi** ดังรูปที่ 13-42



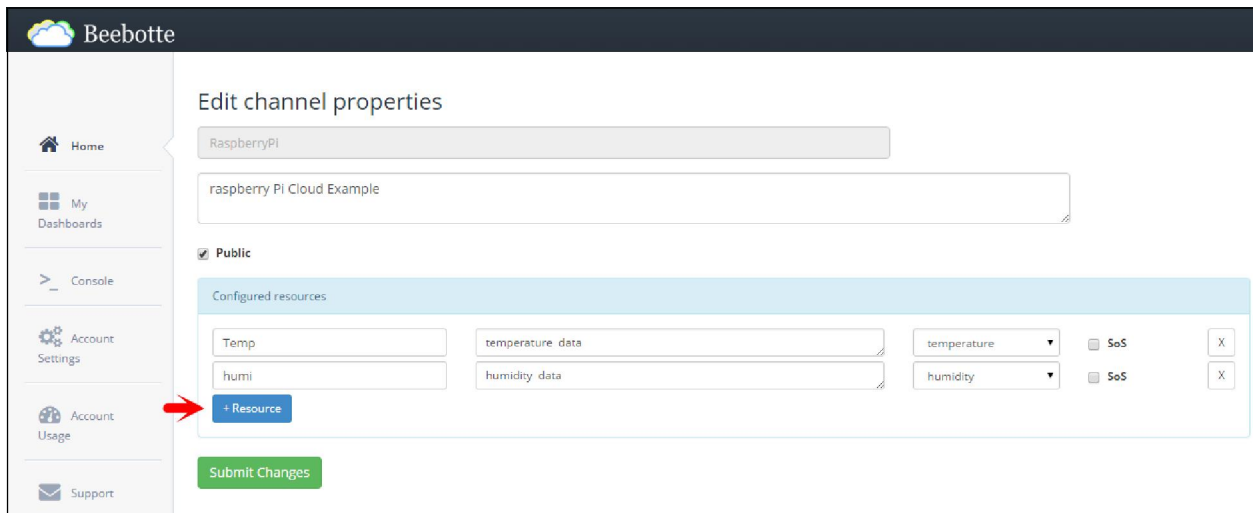
รูปที่ 13-42 เลือกช่องเก็บข้อมูลเพื่อเพิ่มเติมส่วนประกอบ

(13.10.2.3) คลิกที่รูปเฟือง จะปรากฏเมนูต่างๆ ขึ้นมา ให้เลือก **Edit Channel** ดังรูปที่ 13-43



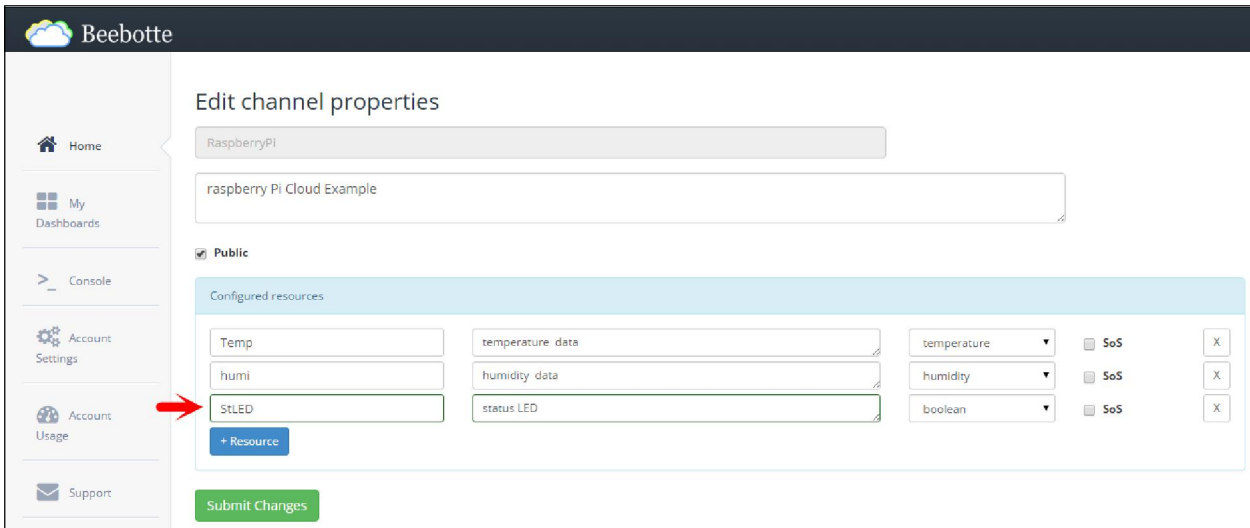
รูปที่ 13-43 เข้าสู่การแก้ไขรายละเอียดของช่องเก็บข้อมูล

(13.10.2.4) คลิกปุ่ม **+ Resource** เพื่อเพิ่มส่วนประกอบหรือ Resource ดังรูปที่ 13-44



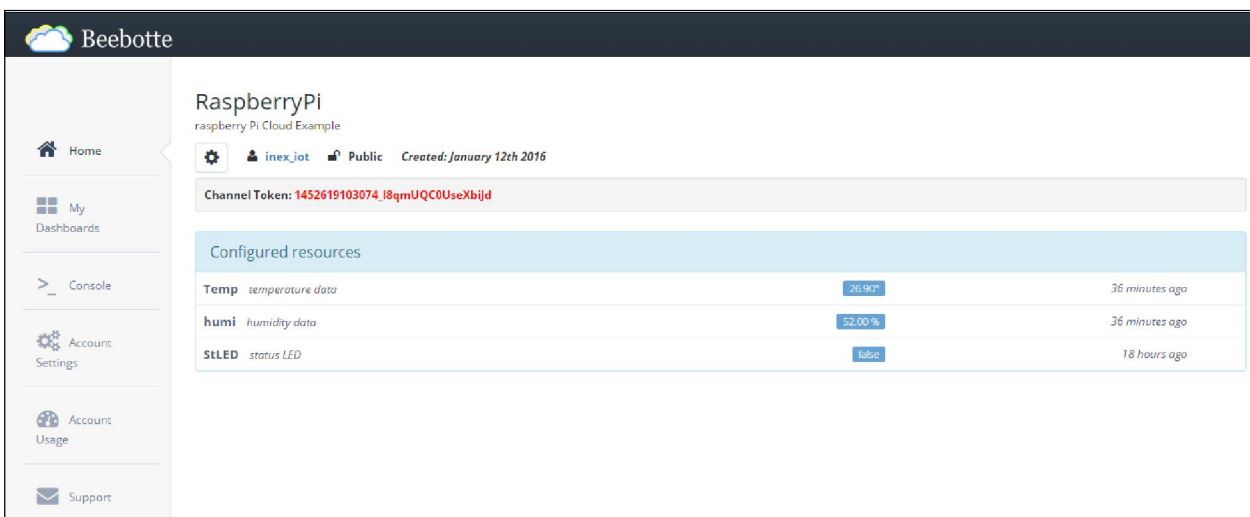
รูปที่ 13-44 เพิ่มส่วนประกอบให้กับช่องเก็บข้อมูล

(13.10.2.5) กำหนดชื่อ Resource เป็น StLED, Resource Description เป็น *status LED* และชนิดข้อมูลเป็น *Boolean* ดังรูปที่ 13-45 จากนั้นคลิก **Submit Changes**



รูปที่ 13-45 เพิ่มส่วนประกอบที่เป็น LED สำหรับแสดงสถานะ

(13.10.2.6) จะปรากฏส่วนประกอบหรือ Resource ของช่องเก็บข้อมูล RaspberryPi เพิ่มขึ้น มาดังรูปที่ 13-46

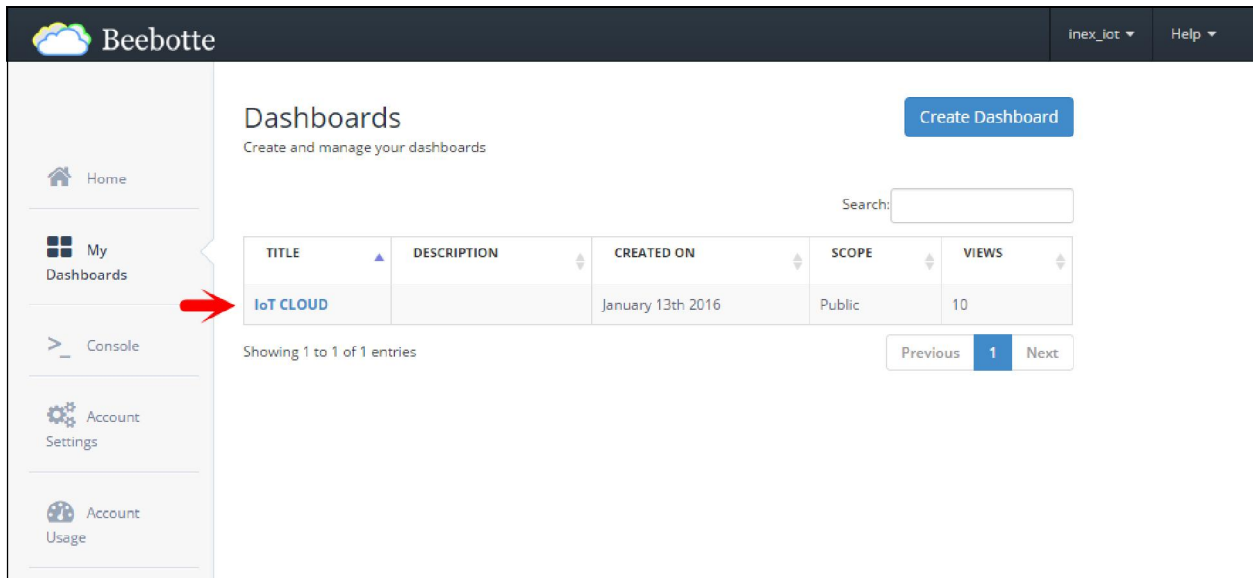


รูปที่ 13-46 แสดงส่วนประกอบที่เพิ่มเข้ามาของช่องเก็บข้อมูล Raspberrypi

13.10.3 ออกแบบแดชบอร์ดที่มีเครื่องมือเป็นสวิตช์เปิดปิด

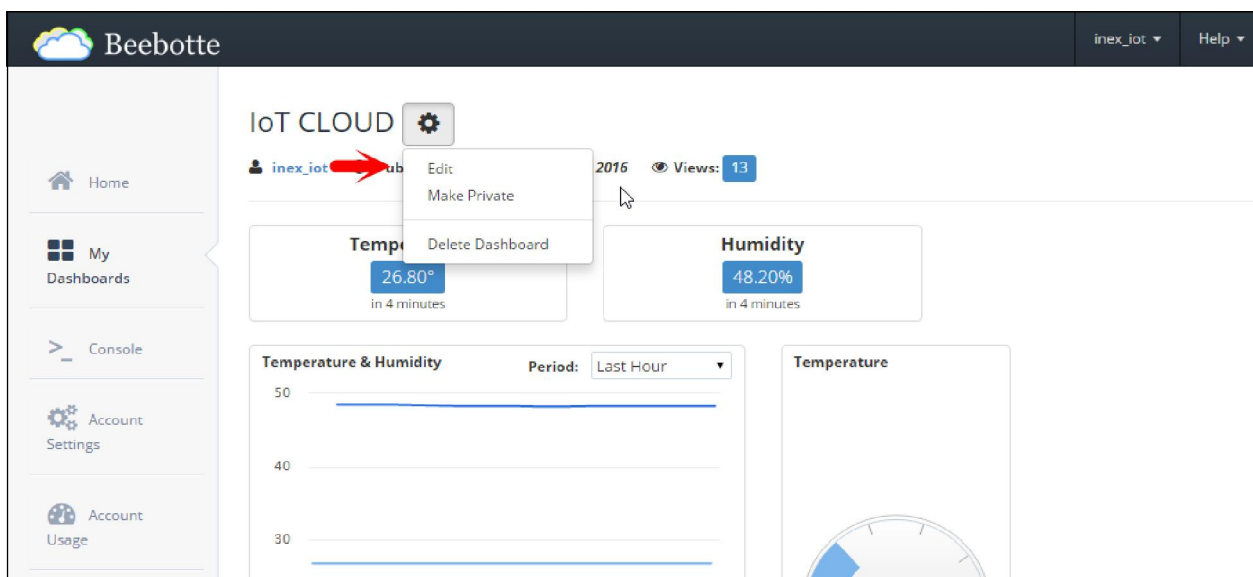
ในที่นี้จะใช้แดชบอร์ดเดิมที่ชื่อ **IoT CLOUD** ซึ่งทำไว้แล้วตั้งแต่หัวข้อ 13.7 มาพัฒนาต่อ โดยเพิ่มสวิตช์สำหรับเปิดปิด LED อีก 1 ตัว โดยมีขั้นตอนดังนี้

(13.10.3.1) ที่แถบเมนู คลิกที่ **My Dashboards** ตามด้วยคลิกที่รายการ **IoT CLOUD** ดังรูปที่ 13-47



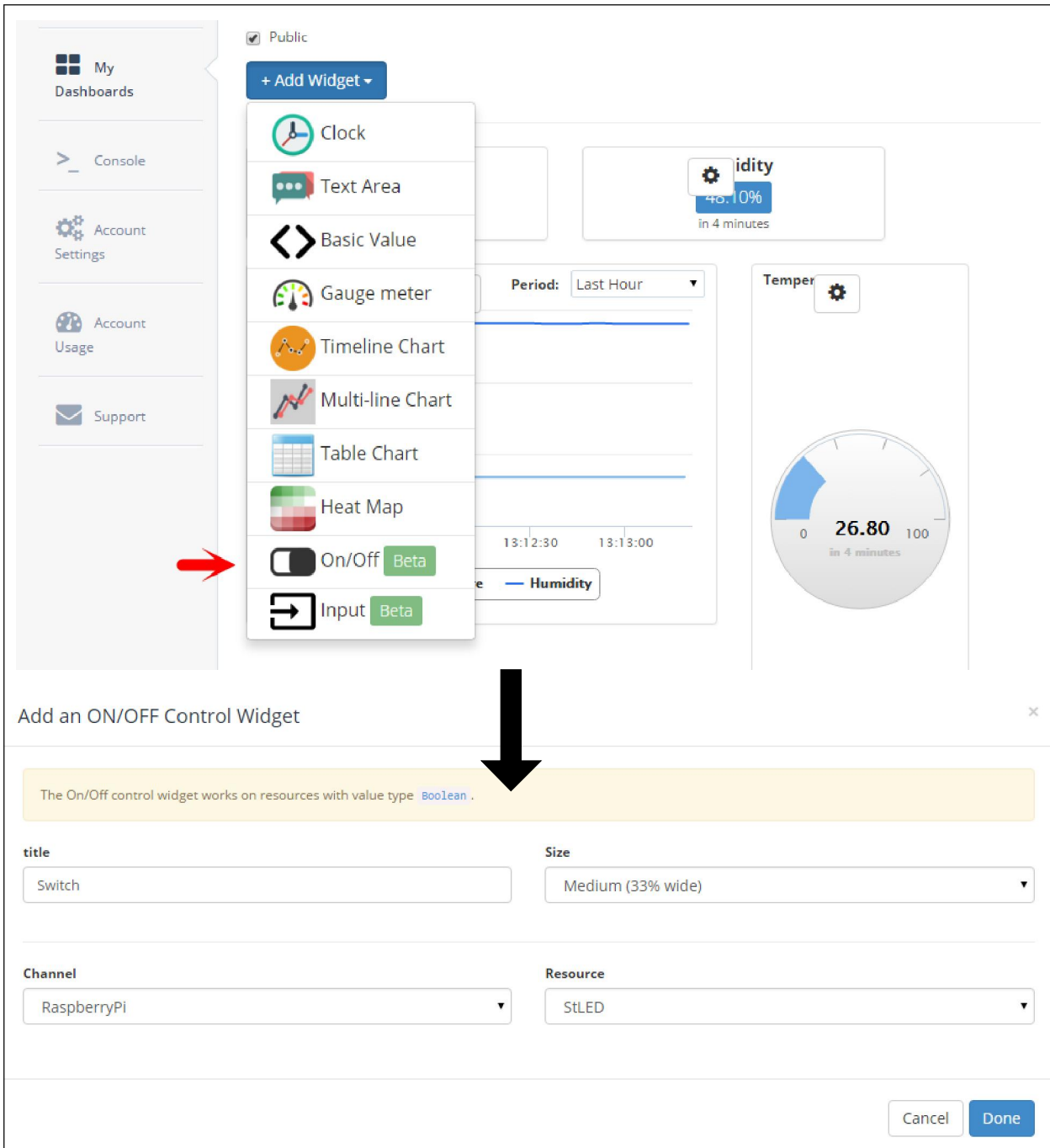
รูปที่ 13-47 เลือกเปิดแดชบอร์ดเดิมที่เคยสร้างไว้แล้ว

(13.10.3.2) คลิกที่รูปเฟือง มีเมนูต่างๆ ปรากฏขึ้นมา ให้เลือก **Edit** ดังรูปที่ 13-48



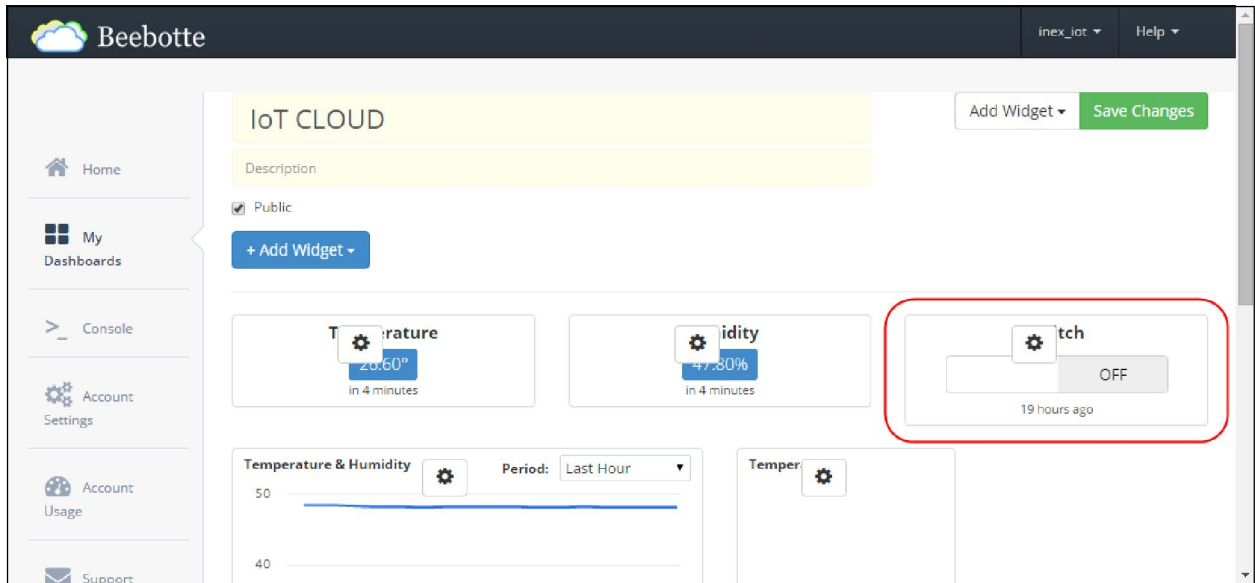
รูปที่ 13-48 การเลือกเพื่อแก้ไขแดชบอร์ดที่เคยสร้างไว้แล้ว

(13.10.3.3) คลิกที่ปุ่ม + Add Widget แล้วเลือก On/Off จากนั้นทำการตั้งค่าต่างๆ โดยกำหนดชื่อเป็น **Switch**, **Channel** คือ **RaspberryPi** และ **Resource** เท่ากับ **StLED** ดังรูปที่ 13-49 แล้วคลิกที่ปุ่ม **Done** ดังรูปที่ 13-49



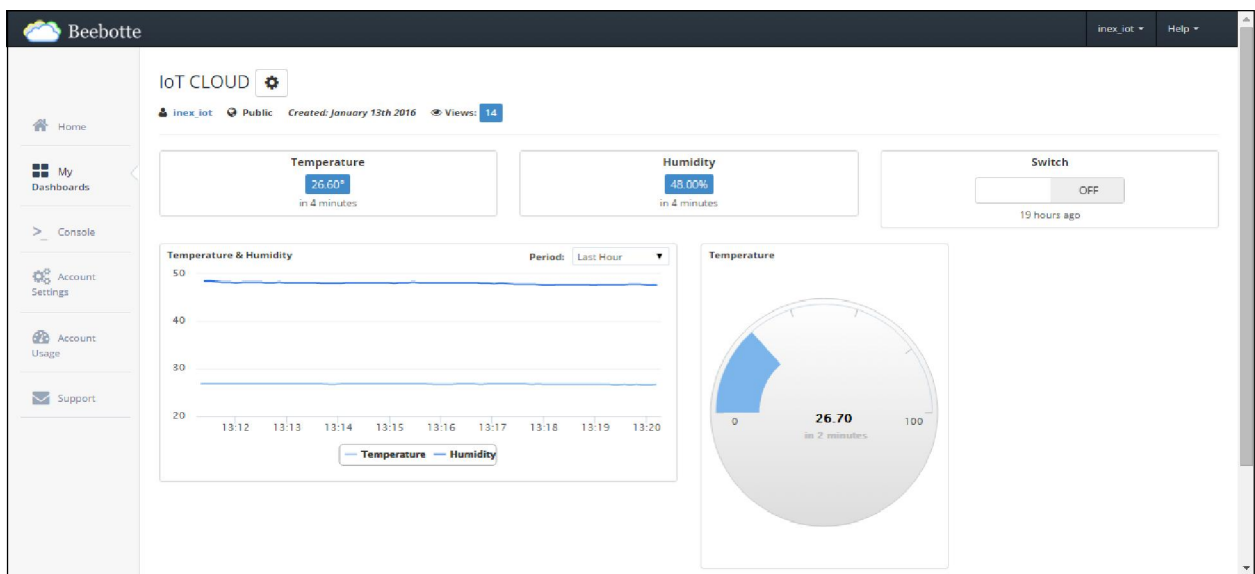
รูปที่ 13-49 เลือกวิดเจ็ตของสวิตช์ On/Off มาใช้งาน

(13.10.3.4) จากนั้นจะปรากฏวิดเจ็ต **Switch** ที่ตั้งค่าแล้วบนหน้าออกแบบของแดชบอร์ดดังรูปที่ 13-50 จากนั้นคลิกปุ่ม **Save Changes**



รูปที่ 13-50 หน้าออกแบบของแดชบอร์ดที่มีวิดเจ็ตรูปสวิตช์เปิดปิดเพิ่มเข้ามา

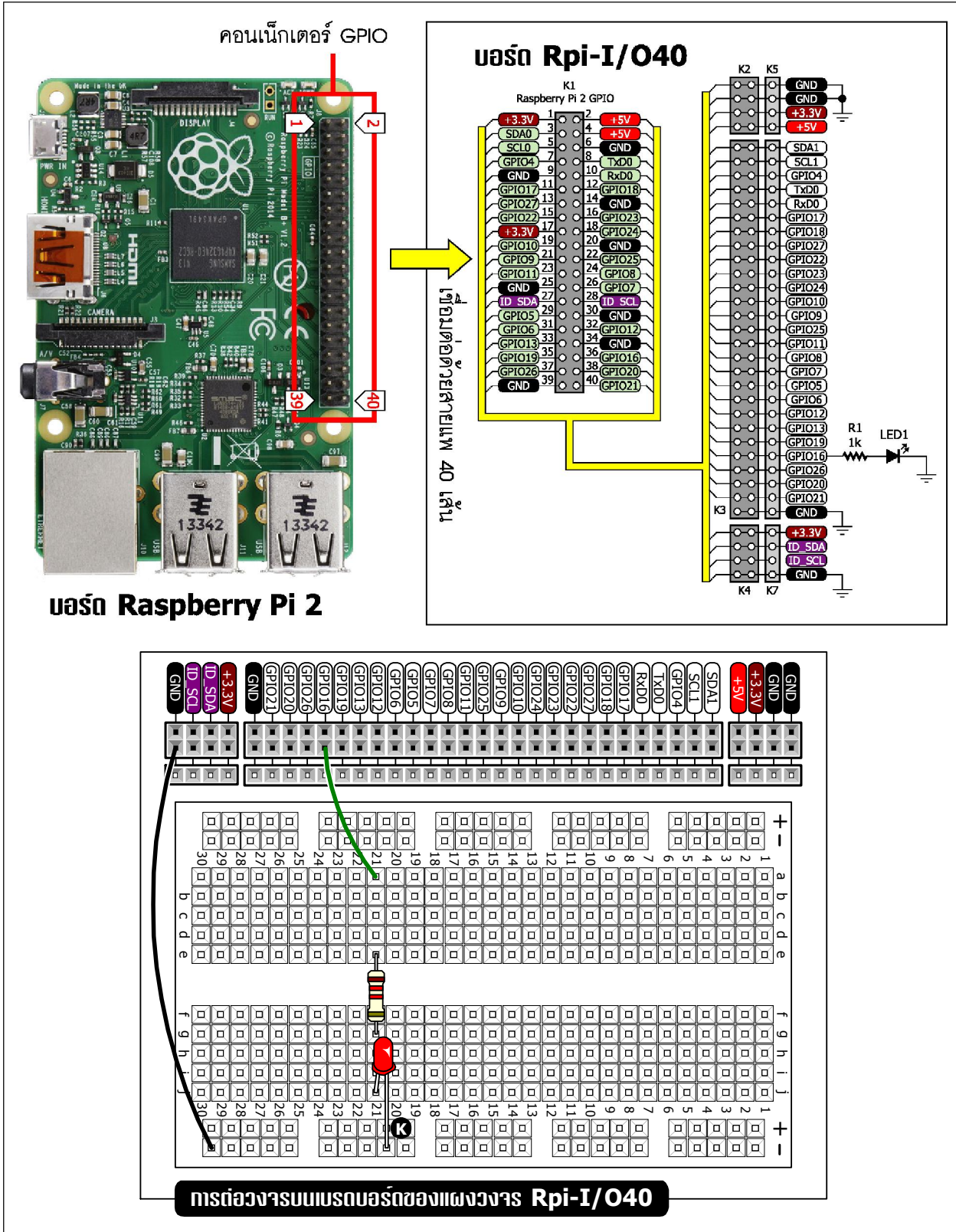
(13.10.3.5) เมื่อบันทึกการเปลี่ยนแปลงเรียบร้อยแล้ว แดชบอร์ดที่สมบูรณ์จะถูกแสดงขึ้นมาดังรูปที่ 13-51 จะเห็นวิดเจ็ตสวิตช์เปิดปิดแสดงที่มุมขวาบน ทดลองคลิกที่รูปสวิตช์ จะเห็นการเปลี่ยนแปลงเป็น **ON** และ **OFF**



รูปที่ 13-51 แดชบอร์ด IoT CLOUD ที่เพิ่มสวิตช์เปิดปิดแล้ว

13.10.4 การดำเนินการที่บอร์ด Raspberry Pi 2

(13.10.4.1) ต่อดังรูปที่ 13-52 ที่พอร์ต GPIO ของบอร์ด Raspberry Pi 2



รูปที่ 13-52 วงจรทดลองควบคุม LED ผ่านคลาวเชิร์ฟเวอร์ Beebotte ของบอร์ด Raspberry Pi 2

```
#!/usr/bin/env python
#####
# This code uses the Beebotte API, you must have an account.
# You can register here: http://beebotte.com/register
#####
import RPi.GPIO as GPIO
import time
import json
from beebotte import *

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
LED_pin=16
GPIO.setup(LED_pin,GPIO.OUT)

API_KEY = 'd7f67dec2607f6ca0e904f458969f0e3'
SECRET_KEY= '574dc23358a92207f83cc908a95343037a71763e471151bb1aeedd8f3a2cfe02'
### Replace API_KEY and SECRET_KEY with those of your account
bbt = BBT(API_KEY,SECRET_KEY)

while True:
    try:
        # Read from Beebotte return json
        records = bbt.read('RaspberryPi','StLED')
        print (records)

        # Get data from json
        StLED=records[0]['data']
        print (StLED)
        GPIO.output(16,StLED)
    except Exception:
        print ("Error while writing to Beebotte")
```

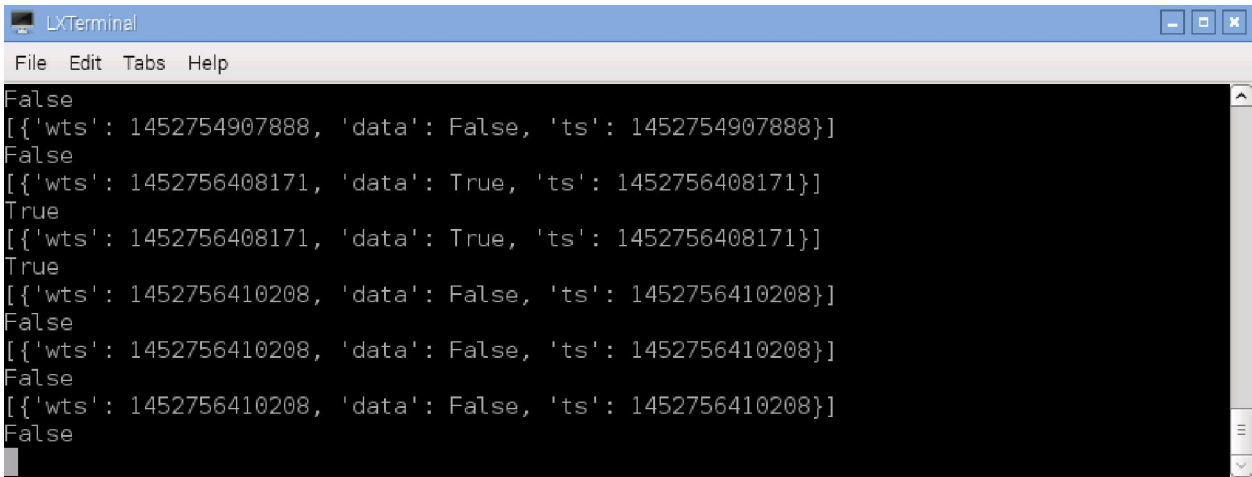
คำอธิบายโปรแกรม

ส่วนสำคัญของโปรแกรมนี้นี้คือ คำสั่ง `records = bbt.read('RaspberryPi','StLED')` เพื่ออ่านค่ามาจากส่วนประกอบหรือ Resource ที่ชื่อว่า `StLED` ผลลัพธ์ที่ส่งกลับมาจะอยู่ในรูปแบบ json โดยมีตัวแปร `records` เก็บผลลัพธ์ไว้ เช่น `[{"wts": 1452682926435,"data": false , "ts": 1452682926435}]` จากตัวอย่าง ข้อมูลที่ต้องการคือ `"data": false` ดังนั้นคำสั่ง `StLED = records[0]['data']` หมายถึง การดึงค่าของตัวแปร `data` มาเก็บไว้ที่ `StLED` เพื่อนำไปเป็นกำหนดสถานะติดดับต่อไป

โปรแกรมที่ 13-2 ไฟล์ `LED.py` โปรแกรมภาษา Python สำหรับบอร์ด Raspberry Pi 2 เพื่อรับการควบคุมอุปกรณ์เอาต์พุตจากคลาวด์เซิร์ฟเวอร์ Beebotte

(13.10.4.2) เปิดโปรแกรม Geany เพื่อสร้างไฟล์ LED.py ตามโปรแกรมที่ 13-2

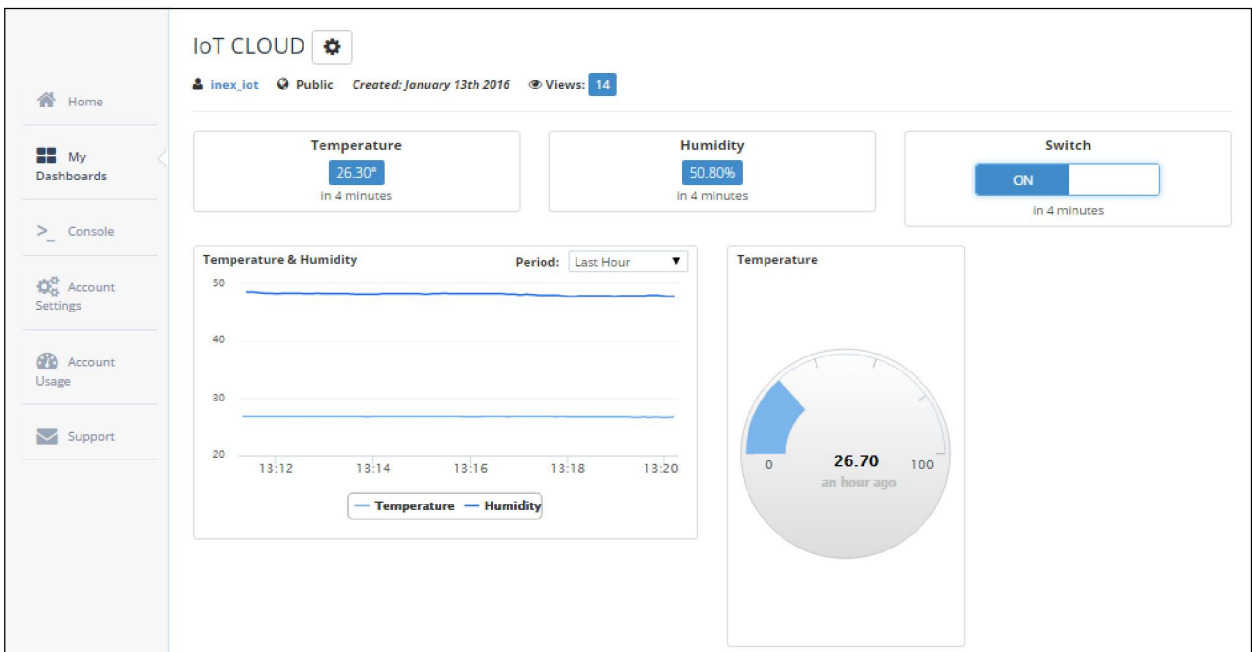
(13.10.4.3) สั่งให้ไฟล์สคริปต์ Python ทำงาน จะได้ผลการทำงานแสดงที่หน้าต่างเทอร์มินอลดังรูปที่ 13-53



```
LXTerminal
File Edit Tabs Help
False
[{'wts': 1452754907888, 'data': False, 'ts': 1452754907888}]
False
[{'wts': 1452756408171, 'data': True, 'ts': 1452756408171}]
True
[{'wts': 1452756408171, 'data': True, 'ts': 1452756408171}]
True
[{'wts': 1452756410208, 'data': False, 'ts': 1452756410208}]
False
[{'wts': 1452756410208, 'data': False, 'ts': 1452756410208}]
False
[{'wts': 1452756410208, 'data': False, 'ts': 1452756410208}]
False
```

รูปที่ 13-53 หน้าต่าง LXTerminal แสดงการทำงานของโปรแกรม LED.py

(13.10.4.4) ทดสอบการทำงานที่แดชบอร์ด IoT CLOUD ที่ทำไว้แล้วบน Beebotte ดังรูปที่ 13-54 โดยคลิกที่รูป Switch แล้วสังเกตการทำงานของ LED ที่ต่อกับพอร์ต GPIO ของบอร์ด Raspberry Pi 2



รูปที่ 13-54 ทดสอบการทำงานที่แดชบอร์ด IoT CLOUD เพื่อควบคุม LED ที่ต่อกับพอร์ต GPIO ของบอร์ด Raspberry Pi 2

13.11 ตัวอย่างโครงการปรับความสว่างของ LED ด้วยแดชบอร์ดบน

Beebotte

13.11.1 ภาพรวมของการพัฒนา

เริ่มต้นด้วยการออกแบบเว็บเพจของคลาวด์เซิร์ฟเวอร์ โดยกำหนดให้มีการแสดงช่องสำหรับรับค่าเพื่อปรับความสว่างของ LED โดยยังคงใช้ช่องเก็บข้อมูลหรือ Channel เดิมคือ **RaspberryPi** ทำการเพิ่มส่วนประกอบหรือ Resource ใหม่เข้าไป ตั้งชื่อว่า **valDuty** จากนั้นสร้างแดชบอร์ดและเขียนโปรแกรมของไฟล์สคริปต์ Python ให้แก่บอร์ด Raspberry Pi 2 เพื่อรับค่าคิวตี้ไซเคิลสำหรับสร้างสัญญาณ PWM จากแดชบอร์ดสำหรับขับ LED ที่ต่อกับขาพอร์ต GPIO ของบอร์ด Raspberry Pi 2

13.11.2 เตรียมการคลาวด์เซิร์ฟเวอร์และแดชบอร์ด

มีขั้นตอนดังนี้

(13.11.2.1) เพิ่มส่วนประกอบหรือ Resource ชื่อ valDuty ชนิดข้อมูลเป็น rate ในช่องเก็บข้อมูลหรือ Channel ที่ชื่อว่า RaspberryPi เมื่อเพิ่มเสร็จแล้ว จะได้ดังรูปที่ 13-55

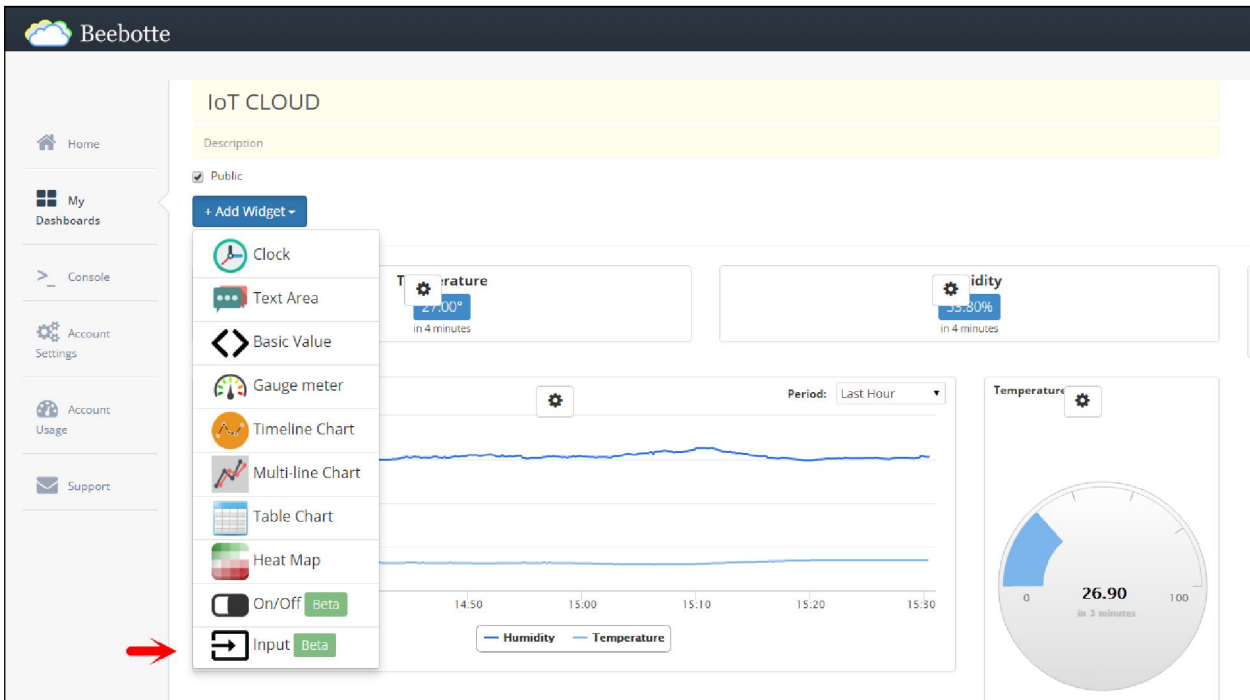
The screenshot shows the Beebotte interface for a Raspberry Pi cloud example. The main content area displays 'Configured resources' with the following data:

Resource Name	Resource Type	Value	Last Update
Temp	temperature data	26.40°	a few seconds ago
humi	humidity data	51.70 %	a few seconds ago
StLED	status LED	true	20 minutes ago
valDuty	duty cycle value	No Activity	

A red arrow points to the 'valDuty' resource in the table.

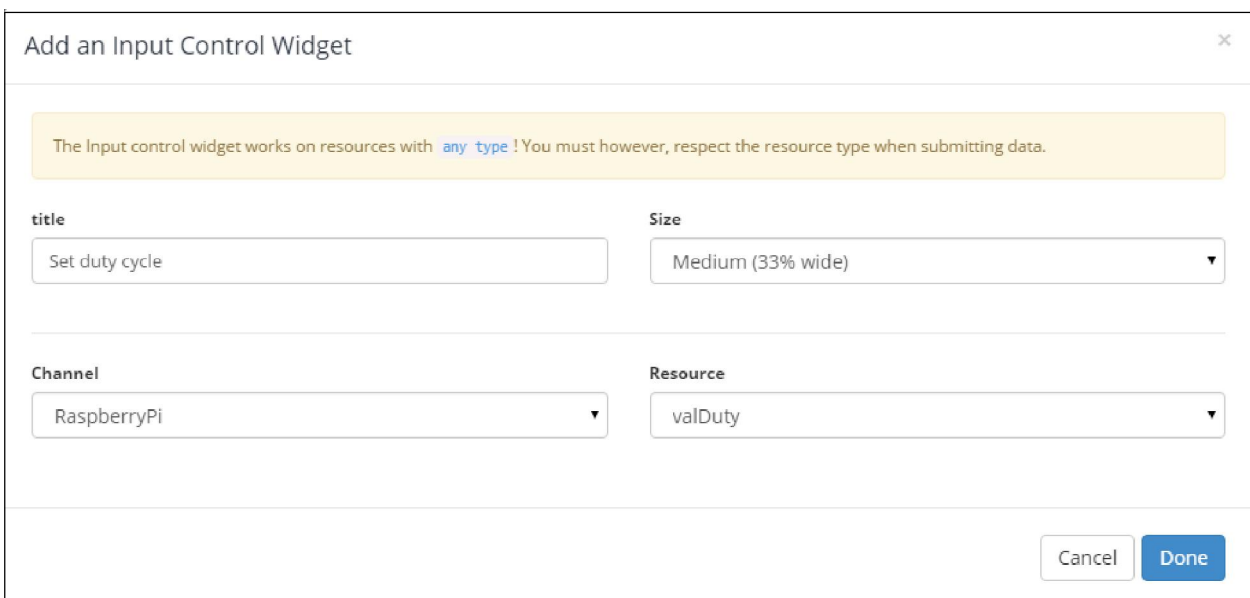
รูปที่ 13-55 เพิ่มส่วนประกอบหรือ Resource ชื่อ valDuty ลงในช่องเก็บข้อมูล RaspberryPi บน Beebotte

(13.11.2.2) ไปที่แดชบอร์ดที่ชื่อว่า **IoT CLOUD** ซึ่งออกแบบมาก่อนหน้านี้แล้ว จากนั้นเพิ่มวิดเจ็ตที่ชื่อว่า **Input** ดังรูปที่ 13-56



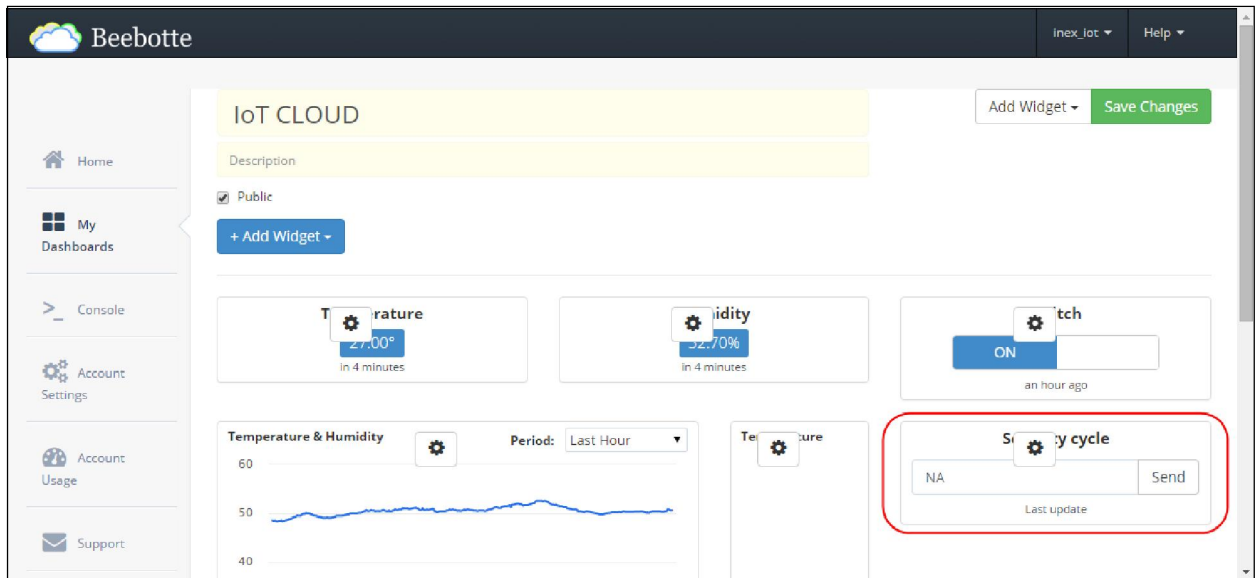
รูปที่ 13-56 เพิ่มวิดเจ็ต Input ลงในแดชบอร์ด IoT CLOUD

(13.11.2.3) กำหนดชื่อวิดเจ็ตเป็น **Set duty cycle** เลือก Resource เป็น **valDuty** ดังรูปที่ 13-57 คลิกปุ่ม **Done** เพื่อบันทึก



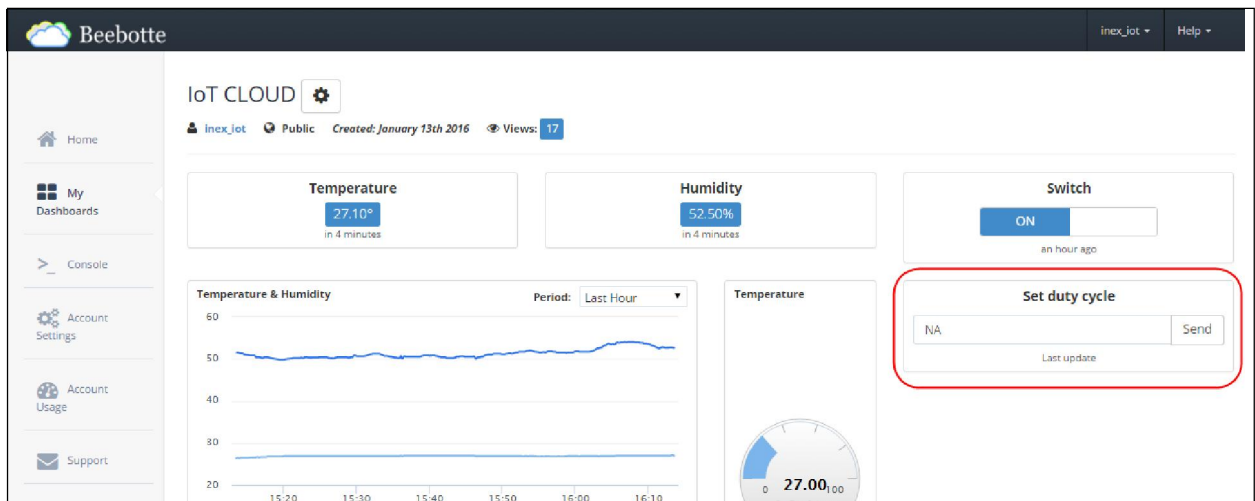
รูปที่ 13-57 กำหนดรายละเอียดของวิดเจ็ต Input

(13.11.2.4) จากนั้นจะปรากฏวิดเจ็ต **Set duty cycle** บนหน้าตาต่างออกแบบแดชบอร์ด ดังรูปที่ 13-58 คลิกปุ่ม **Save Changes** เพื่อยืนยันการเปลี่ยนแปลงรูปที่ 13-58 หน้าออกแบบของแดชบอร์ดที่มีวิดเจ็ตช่องป้อนข้อมูลเพิ่มเข้ามา

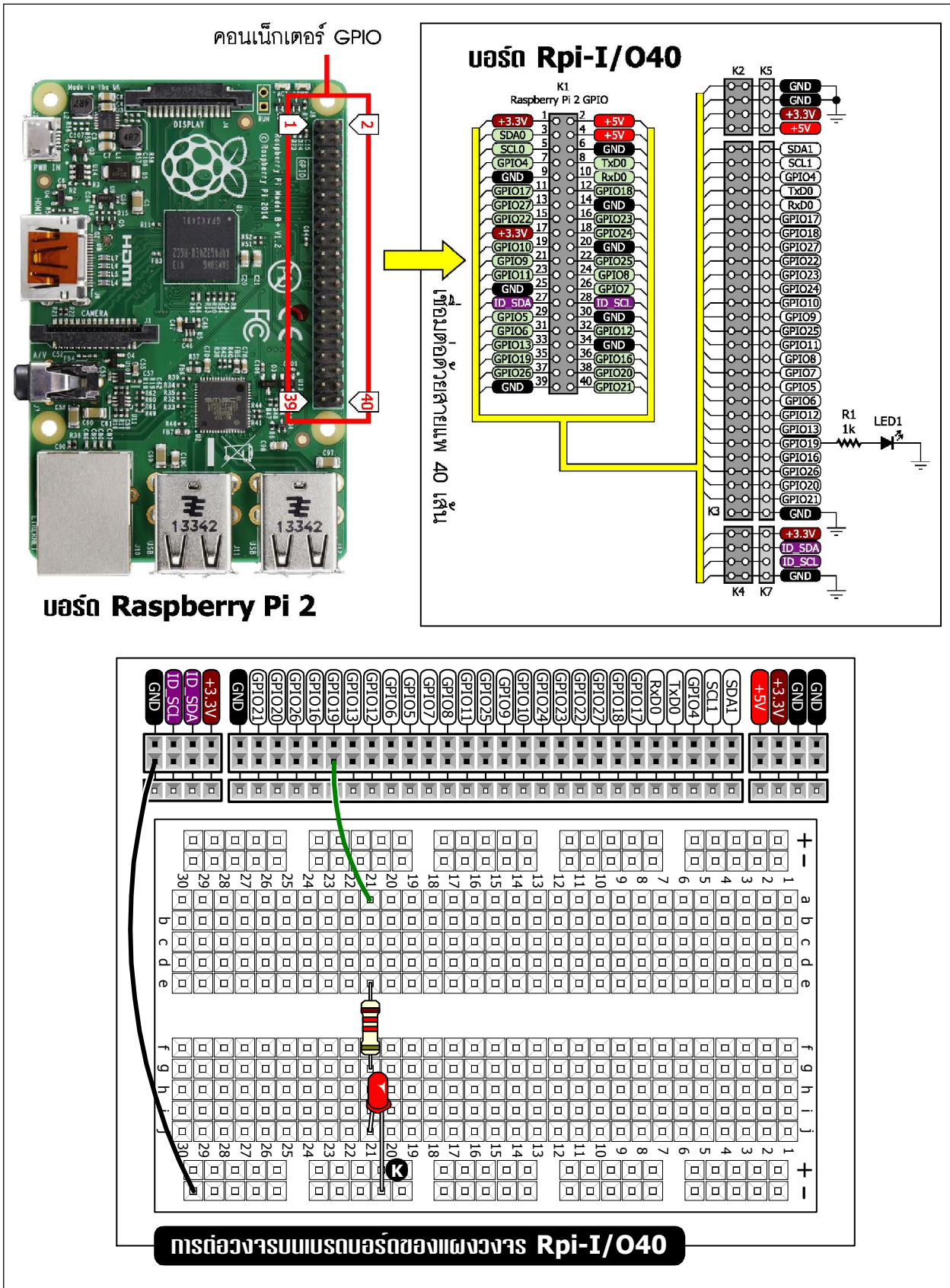


รูปที่ 13-58 หน้าออกแบบของแดชบอร์ด IoT CLOUD ที่มีวิดเจ็ตช่องป้อนข้อมูลเพิ่มเข้ามา

(13.11.2.5) แดชบอร์ด IoT CLOUD ที่มีช่องป้อนค่า **Set duty cycle** แสดงขึ้นมาดังรูปที่ 13-59



รูปที่ 13-59 แดชบอร์ด IoT CLOUD ที่พร้อมสำหรับส่งค่าไปยังบอร์ด Raspberry Pi 2 เพื่อขับ LED ด้วยสัญญาณ PWM



รูปที่ 13-60 วงจรทดลองปรับความสว่าง LED ผ่านคลาวด์เซอร์ฟเวอร์ Beebotte ของบอร์ด Raspberry Pi 2

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-
import RPi.GPIO as GPIO
import time
from beebotte import *

LED_pin=19
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(LED_pin, GPIO.OUT)
blink = GPIO.PWM(LED_pin,500)
blink.start(0)

API_KEY = 'd7f67dec2607f6ca0e904f458969f0e3'
SECRET_KEY= '574dc23358a92207f83cc908a95343037a71763e471151bb1aeedd8f3a2cfe02'
### Replace API_KEY and SECRET_KEY with those of your account
bbt = BBT(API_KEY,SECRET_KEY)

while True:
    try:
        # Read from Beebotte return json
        records = bbt.read('RaspberryPi','valDuty')
        print (records)

        # Get data from json
        valpwm=records[0]['data']
        print (valpwm)
        blink.ChangeDutyCycle(valpwm)

    except Exception:
        print ("Error while writing to Beebotte")
```

คำอธิบายโปรแกรม

จุดสำคัญของโปรแกรมนี้คือการใช้คำสั่ง `records = bbt.read('RaspberryPi','valDuty')` เพื่ออ่านค่าจากส่วนประกอบหรือ Resource ที่ชื่อว่า `valDuty` ผลลัพธ์ที่ส่งกลับมาอยู่ในรูปแบบ json ตัวแปร `records` จะเก็บผลลัพธ์ไว้ เช่น `[{'wts': 1452764880455, "data": 90, 'ts': 1452764880455}]`

จากตัวอย่าง ข้อมูลที่ต้องการคือ `"data": 90` ดังนั้นคำสั่ง `valpwm=records[0]['data']` หมายถึง การดึงค่าที่ตัวแปร `"data"` ระบุไว้มาเก็บไว้ที่ `valpwm` เพื่อนำไปกำหนดความกว้างของสัญญาณ PWM ด้วยคำสั่ง `blink.ChangeDutyCycle(valpwm)`

โปรแกรมที่ 13-3 ไฟล์ LEDPWM.py โปรแกรมภาษา Python สำหรับบอร์ด Raspberry Pi 2 เพื่อรับการควบคุมความสว่างของ LED ที่ต่อกับพอร์ต GPIO จากคลาวด์เซิร์ฟเวอร์ Beebotte

การนำบอร์ด Raspberry Pi 2 มาพัฒนาเป็นอุปกรณ์ IoT อาจมีขั้นตอนพอสมควร เนื่องจากมีส่วนประกอบที่ต้องทำงานร่วมด้วยมากพอสมควร ทั้งการตั้งค่าที่คลาวด์เซิร์ฟเวอร์ การออกแบบและสร้างแดชบอร์ดเพื่อแสดงผล การเขียน โปรแกรมเพื่อสร้างไฟล์สคริปต์ด้วยภาษา Python หรือภาษาอื่นๆ โดยใช้ไลบรารีของการเชื่อมต่อกับคลาวด์เซิร์ฟเวอร์ จากนั้นนำทั้งหมดมาทำงานร่วมกัน ซึ่งหากมีการฝึกฝนอย่างต่อเนื่อง และพยายามทำความเข้าใจ จะพบการพัฒนาไม่ได้ยากจนเกินไป ประโยชน์ที่ได้รับจากการเรียนรู้มีค่าและสามารถนำไปต่อยอดได้อย่างมากมาย การให้บริการของคลาวด์เซิร์ฟเวอร์อาจแตกต่างกันไปในแต่ละผู้ให้บริการ ขอเพียงเข้าใจในแนวคิดและกระบวนการ ผู้พัฒนา ก็สามารถปรับเปลี่ยนหรือคัดแปลงเพื่อให้ทำงานได้ไม่ยาก



