

## 240-319 Embedded System Developer Module

### Lab 18

### NodeMCU Wi-Fi Connection ปีการศึกษา 2567

ผู้สอน รองศาสตราจารย์ ดร.ปัญญาศ ไชยกาฬ

#### 1. วัตถุประสงค์

- สามารถเขียนโปรแกรมบนบอร์ดทดลอง NodeMCU สำหรับเชื่อมต่อกับระบบเครือข่ายไร้สาย
- สามารถเขียนโปรแกรมเพื่อให้ NodeMCU อ่านสถานะของสวิตช์และแสดงผลบนแอลอีดี
- สามารถเขียนโปรแกรมบนบอร์ดทดลอง NodeMCU เพื่อรองรับการควบคุมจากโทรศัพท์มือถือที่ใช้ระบบปฏิบัติการ iOS และ Android และสามารถส่งค่าสถานะของฮาร์ดแวร์ไปแสดงผลบนโทรศัพท์มือถือได้
- สามารถเขียนโปรแกรมควบคุม NodeMCU และตรวจสอบข้อบกพร่องผ่านโปรแกรม Serial Monitor ได้

#### 2. อุปกรณ์ที่ใช้ในการทดลอง

- บอร์ด NodeMCU พร้อม Shield จำนวน 1 บอร์ด
- สายไมโครยูเอสบี จำนวน 1 เส้น
- โทรศัพท์มือถือระบบปฏิบัติการ Android หรือ iOS 1 เครื่อง (นักศึกษานำมาเองในวันทดลอง)

#### 3. การส่งงานและวิธีการทำแลบ

การทดลองนี้ประกอบไปด้วย 4 Checkpoint โดยแบ่งคะแนนออกเป็นดังนี้

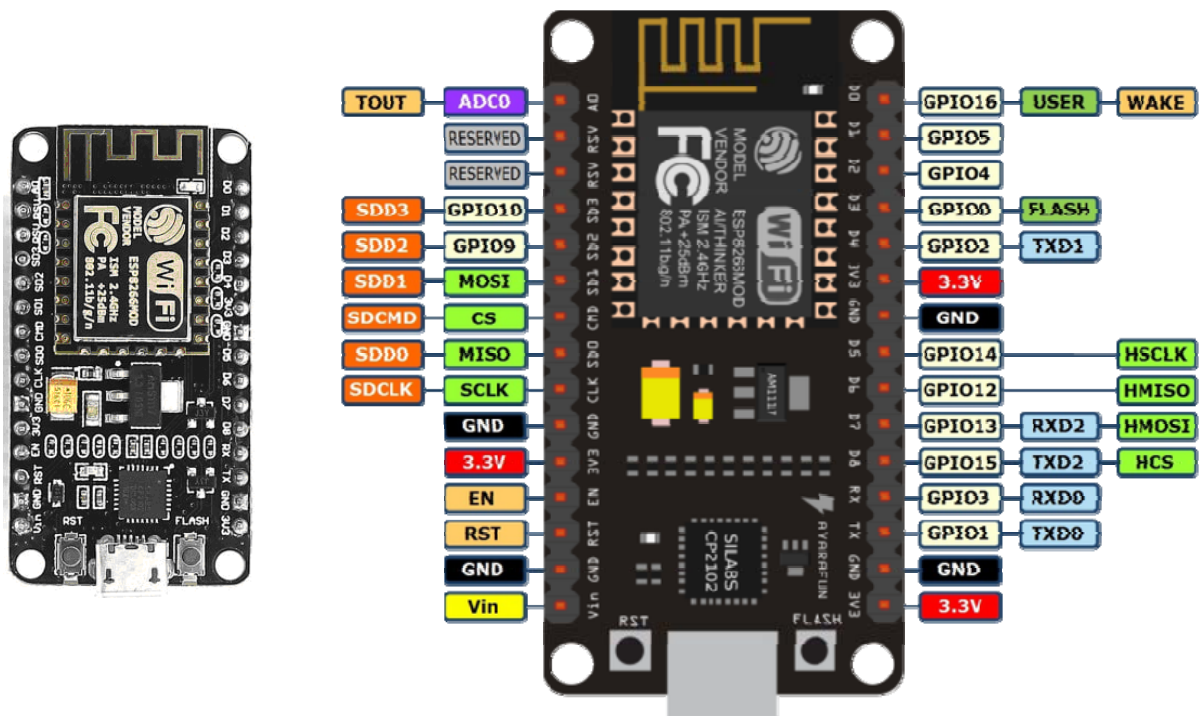
- |               |    |   |
|---------------|----|---|
| - checkpoint1 | 30 | % |
| - checkpoint2 | 20 | % |
| - checkpoint3 | 30 | % |
| - checkpoint4 | 20 | % |

การตรวจ checkpoint ให้เรียก TA หรือเจ้าหน้าที่หรืออาจารย์ตรวจในคาบเวลาปฏิบัติการเท่านั้น

#### 4. แนะนำบอร์ดประมวลผล NodeMCU

NodeMCU เป็นแพลตฟอร์ม IoT แบบโอเพนซอร์ส ซึ่งใช้ตัวประมวลผล ESP8266 SoC ประกอบด้วย ส่วนของ ไมโครคอนโทรลเลอร์ขนาด 32 บิต และส่วนของ Firmware ซึ่งมี TCP/IP Stack ชนิดเต็มรูปแบบในตัว คุณสมบัติของระบบมีดังนี้

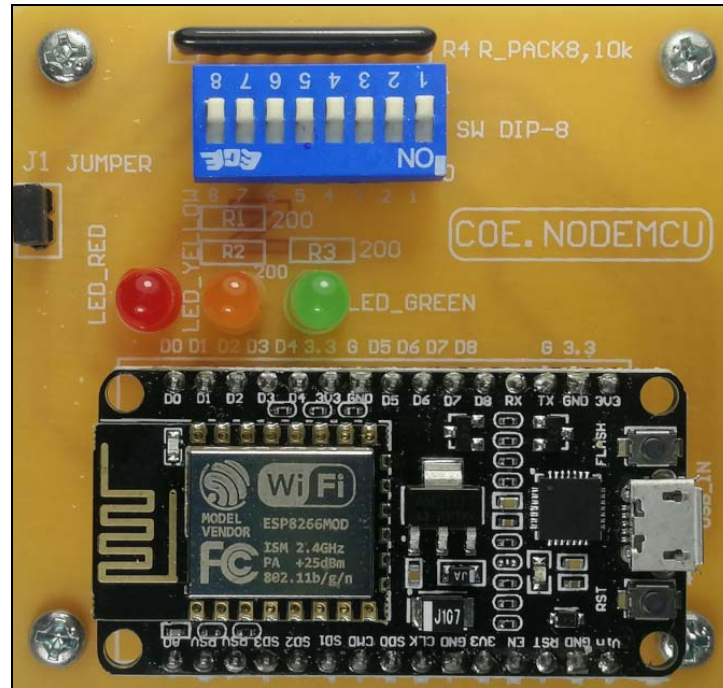
- L106 32-bit RISC (80 MHz)
- 32 KiB instruction RAM
- 80 KiB user-data RAM
- 16 KiB ETS system-data RAM
- IEEE 802.11 b/g/n Wi-Fi  
WEP or WPA/WPA2 authentication, or open networks
- 13 GPIO pins
- UART on dedicated pins
- 10 bit ADC (successive approximation)



รูปที่ 1 บอร์ด NodeMCU และขาใช้งาน

NodeMCU มีขาอินพุตเอาต์พุตเหลือให้ผู้พัฒนาโปรแกรมใช้งานทั้งสิ้น 13 ขา ได้แก่ GPIO0-GPIO5, GPIO9-GPIO10 และ GPIO12-GPIO16 ดังแสดงในรูปที่ 1 (ขวามือ) ผู้ใช้สามารถเขียนโปรแกรมควบคุมในสภาพแวดล้อม Arduino IDE ได้ มีคลังโปรแกรม (Library) ให้ผู้ใช้งานสามารถดาวน์โหลดมาใช้งานจำนวนมาก ช่วยประหยัดเวลาในการพัฒนาโปรแกรม และสามารถติดต่อกับเครือข่ายอินเทอร์เน็ตผ่านการเชื่อมต่อ Wi-Fi ส่งผลให้ผู้ใช้งานสามารถควบคุมฮาร์ดแวร์ตลอดจนการรับค่าสถานะจากฮาร์ดแวร์ไปแสดงผลบนอุปกรณ์ใด ๆ ก็ได้ที่มีการเชื่อมต่ออินเทอร์เน็ต ยกตัวอย่างเช่น โทรศัพท์มือถือ หรือเครื่องคอมพิวเตอร์ เป็นต้น

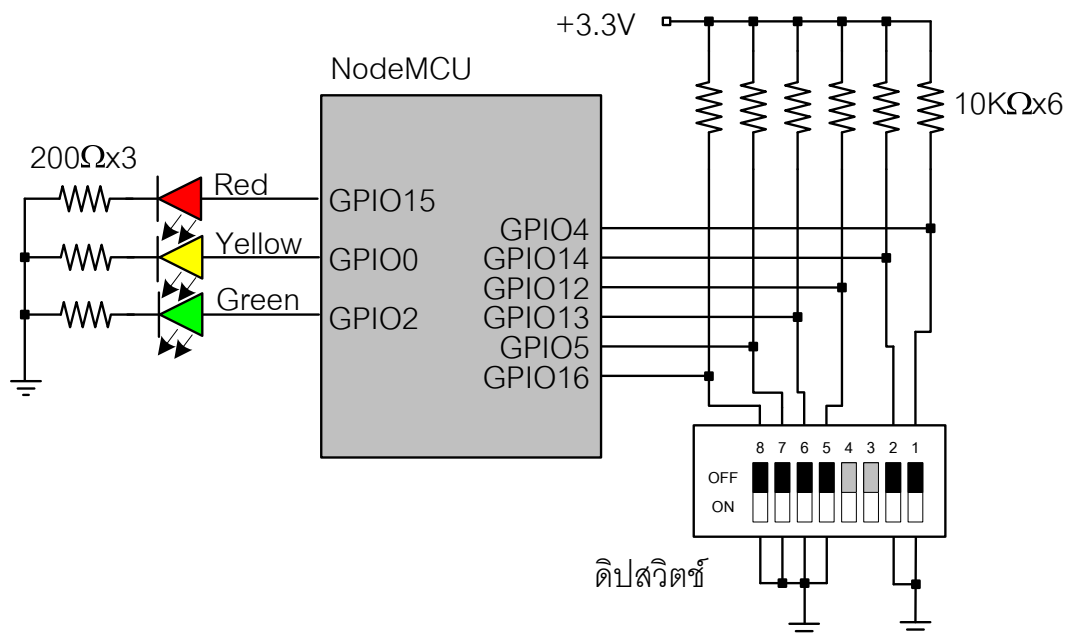
เนื่องจากบอร์ดประมวลผล NodeMCU ไม่มีอุปกรณ์ต่อพ่วงภายนอกให้ใช้ ดังนั้นการใช้งานจริงจึงต้องมีการต่ออุปกรณ์อื่นข้างนอกซึ่งก่อให้เกิดความยุ่งยากและเสียเวลา เพื่อลดปัญหาดังกล่าว ทางสาขาวิชาวิศวกรรมคอมพิวเตอร์ได้พัฒนาบอร์ดเสริมสำหรับเชื่อมต่อกับบอร์ดประมวลผล NodeMCU ดังรูปที่ 2 บอร์ดเสริมนี้เรียกว่า NodeMCU Shield ประกอบไปด้วยองค์ประกอบ 3 ส่วน คือ (1) ดิปลิวซ์ขนาด 8 บิตจำนวน 1 ตัว (2) แอลอีดีจำนวน 3 ตัว และ (3) บอร์ดประมวลผล NodeMCU โดยมีรายละเอียดการเชื่อมต่อระหว่างแอลอีดีและดิปลิวซ์กับบอร์ดประมวลผลดังวงจรในรูปที่ 3



รูปที่ 2 บอร์ด NodeMCU Shield ที่ใช้ในการทดลอง

รูปที่ 3 แสดงรายละเอียดของวงจรของบอร์ด NodeMCU Shield จะสังเกตเห็นว่าแม่ติปสวิตช์จะมีจำนวน 8 บิต แต่มีการต่อใช้งานจริงเพียง 6 บิตเท่านั้น ซึ่งก็คือ บิตที่ 1, 2, 5, 6, 7 และ 8 ส่วนบิตที่ 3 และ 4 ของติปสวิตช์มิได้มีการต่อใช้งานแต่อย่างใด

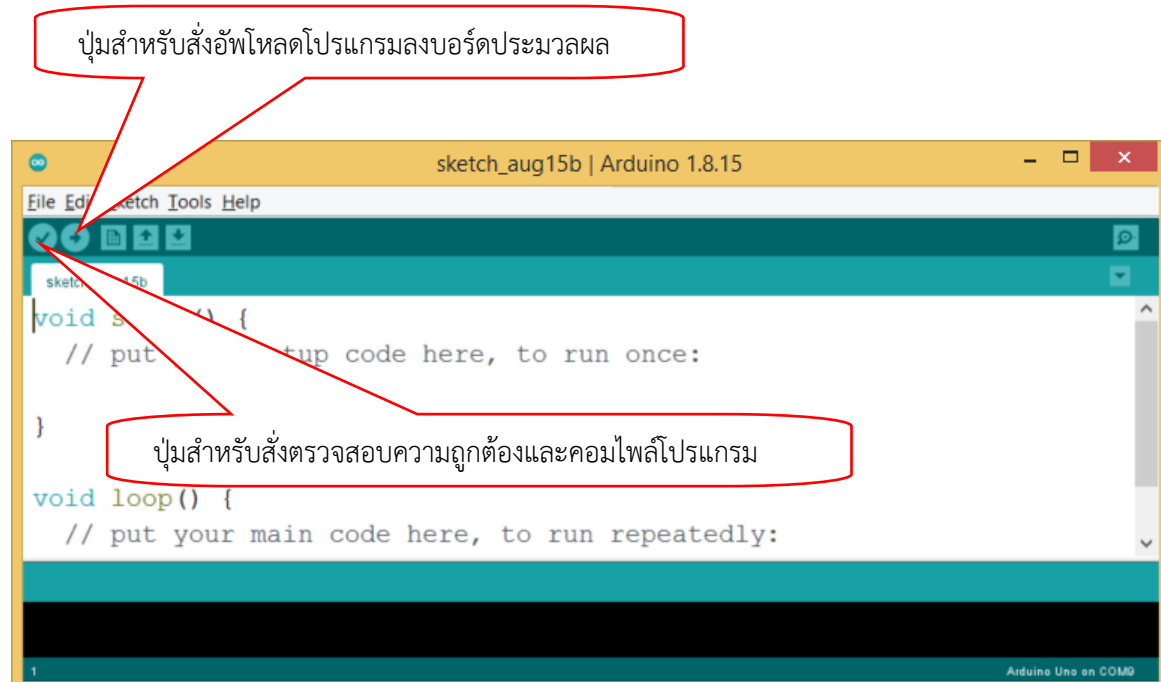
จะเห็นว่าบิตที่ 1 ของติปสวิตช์ถูกเชื่อมต่อเข้ากับ GPIO4 เมื่อเลื่อนบิตของติปสวิตช์ลงมาข้างล่าง จะส่งผลให้ GPIO4 ได้ค่าตรรกะต่ำ (ลอจิก Low) แต่ถ้าหากเลื่อนบิตของติปสวิตช์ขึ้นข้างบน จะส่งผลให้ GPIO4 อ่านค่าได้ตรรกะสูง (ลอจิก High) การเลื่อนตำแหน่งของติปสวิตช์บิตอื่น ๆ ก็ส่งผลให้เกิดการเปลี่ยนแปลงค่าตรรกะที่ขา GPIO ของตัวประมวลผลในลักษณะเช่นเดียวกัน



รูปที่ 3 วงจรของบอร์ด NodeMCU Shield ที่ใช้ในการทดลอง

## 5. Arduino IDE

โปรแกรม Arduino IDE เป็นแอปพลิเคชันสำหรับพัฒนาโปรแกรมสำหรับบอร์ดประมวลผล ซึ่งทำงานได้กับหลายแพลตฟอร์ม เช่น Arduino NodeMCU และ ESP32 เป็นต้น โดยโปรแกรมมีเอดิเตอร์ในตัว ช่วยให้สามารถเขียนโปรแกรมควบคุมไมโครโพรเซสเซอร์หรือไมโครคอนโทรลเลอร์ได้ง่าย นอกจากนี้ยังมีคอมไพเลอร์และแอสเซมบลอร์ในตัว อีกทั้งยังมีความสามารถในการอัปโหลดโปรแกรมควบคุมลงสู่ไมโครคอนโทรลเลอร์ซึ่งอยู่บนแผงวงจรได้โดยไม่ต้องหาโปรแกรมอื่นมาใช้งานเพิ่มเติม ซึ่งช่วยอำนวยความสะดวกให้กับผู้พัฒนาระบบเป็นอย่างมาก รูปที่ 4 แสดงตัวอย่างหน้าจอของโปรแกรม Arduino IDE

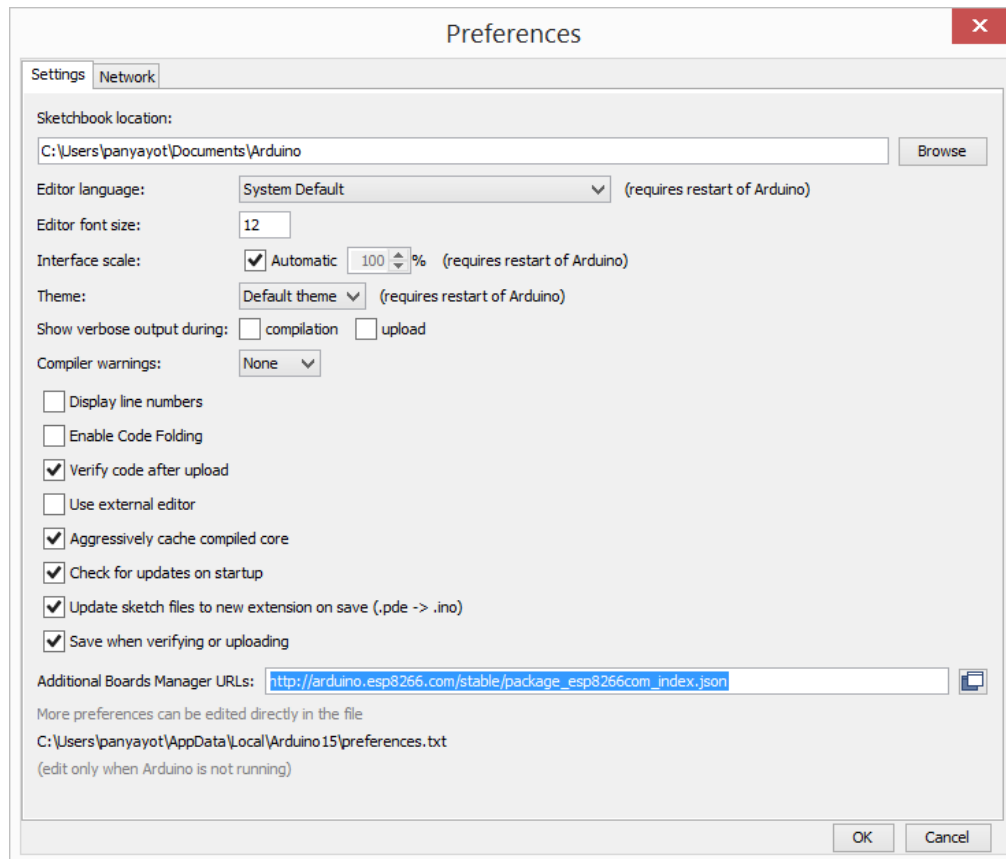


รูปที่ 4 ตัวอย่างหน้าจอของโปรแกรม Arduino IDE

ให้นักศึกษาดาวน์โหลดโปรแกรม Arduino IDE จากเว็บไซต์ <https://www.arduino.cc/en/software> หรือจาก <https://lms2.psu.ac.th/mod/resource/view.php?id=62540> จากนั้นให้ติดตั้งโปรแกรมลงในเครื่องคอมพิวเตอร์ของตนเองให้เรียบร้อยก่อนวันลงแล็บ

หลังจากติดตั้งโปรแกรม Arduino IDE บนเครื่องคอมพิวเตอร์ของตนเองเรียบร้อยแล้ว ให้นักศึกษาทำการติดตั้งไลบรารีเพิ่มเติม เพื่อให้โปรแกรม Arduino IDE สนับสนุนบอร์ดประมวลผล NodeMCU โดยมีขั้นตอนในการดำเนินการดังนี้

1. นำสาย micro USB มาเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์และบอร์ด NodeMCU
2. เรียกโปรแกรม Arduino IDE เลือก File>Preferences จะขึ้นไดอะล็อกบ็อกซ์ดังรูปที่ 5 กรอกข้อความ [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) ลงในช่อง Additional Board Manager URLs



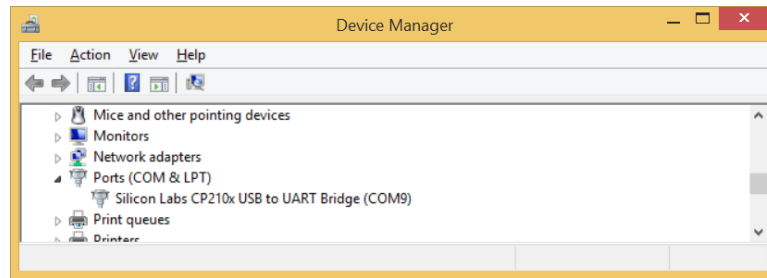
รูปที่ 5 ไดอะล็อกบ็อกซ์ Preference สำหรับเพิ่ม URL ของ Board Manager

3. ในโปรแกรม Arduino IDE เลือกเมนู Tools>Board>Board Manager จะขึ้นไดอะล็อกบ็อกซ์ Boards Manager ดังรูปที่ 6 สั่งให้เพิ่มไลบรารี ESP8266 แล้วกดปุ่ม Install

4. ตรวจสอบว่าเครื่องคอมพิวเตอร์ที่ใช้งานอยู่สามารถใช้งานร่วมกับบอร์ด NodeMCU ได้หรือไม่ ด้วยการเรียก Device Manager ใน Control Panel หากปรากฏรายการ Silicon Labs CP210x USB to UART Bridge ดังรูปที่ 7 แสดงว่าเครื่องคอมพิวเตอร์ที่ใช้งานอยู่มีการลงโปรแกรมขับอุปกรณ์ NodeMCU เสร็จสิ้นเรียบร้อยแล้ว โดยให้สังเกตหมายเลขพอร์ตอนุกรมเสมือนของบอร์ด NodeMCU ว่าได้รับการติดตั้งเป็นพอร์ตหมายเลขใด ซึ่งในเครื่องคอมพิวเตอร์ต่างเครื่องกันอาจมีการกำหนดหมายเลขนี้ให้กับบอร์ด NodeMCU ไม่เหมือนกันก็ได้ ตัวอย่างในรูปที่ 7 จะเห็นว่ามีการตั้งค่าให้บอร์ด NodeMCU เข้ากับพอร์ตอนุกรมเสมือนหมายเลข 9 (COM9)

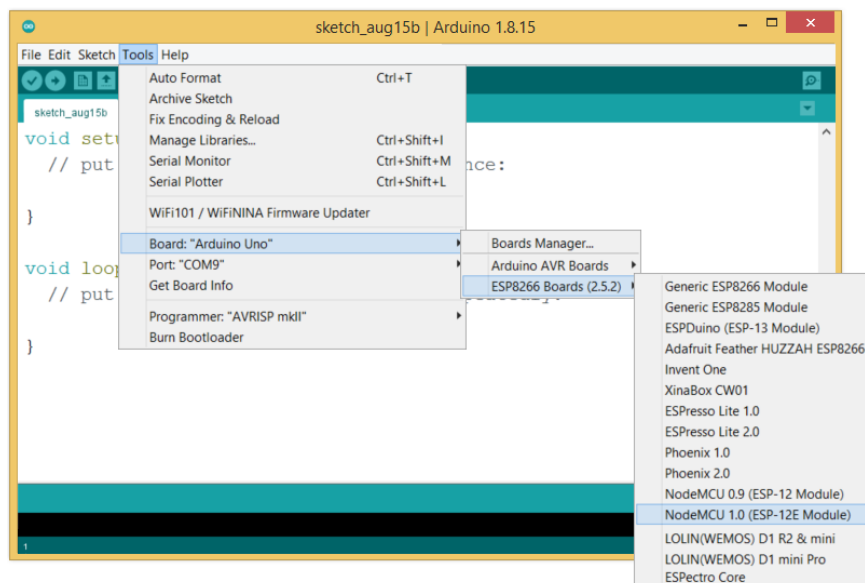


รูปที่ 6 ไดอะล็อกบ็อกซ์ Board Manager สำหรับเพิ่มโปรแกรมขับอุปกรณ์ NodeMCU ESP8266



รูปที่ 7 หน้าต่าง Device Manager

5. เลือกบอร์ด NodeMCU 1.0 ดังรูปที่ 8 ก็เป็นอันเสร็จขั้นตอนการเตรียมพร้อมให้โปรแกรม Arduino IDE สามารถใช้งานกับบอร์ดประมวลผล NodeMCU ได้



รูปที่ 8 การเลือกบอร์ด NodeMCU ในโปรแกรม Arduino IDE

## 6. แนะนำการเขียนโปรแกรมควบคุม NodeMCU

การเขียนโปรแกรมควบคุม NodeMCU ทำได้โดยใช้โปรแกรม Arduino IDE โดยโปรแกรมที่เขียนจะมีโครงสร้างคล้ายกับภาษาซี ตามข้อกำหนดของโปรแกรมที่เขียนขึ้นในสภาพแวดล้อม Arduino จะต้องประกอบไปด้วย ฟังก์ชันหลักจำนวน 2 ตัว คือ ฟังก์ชัน setup และฟังก์ชัน loop

1	#define Y_LED 0	//หลอดแอลอีดีสีเหลือง ต่อกับ GPIO0
2	#define R_LED 15	//หลอดแอลอีดีสีแดง ต่อกับ GPIO15
3	#define SW1 4	//คิปปสวิตช์บิตที่ 1 ต่อกับ GPIO4
4	void setup()	//ฟังก์ชันสำหรับการตั้งค่าเริ่มต้นใช้งานบอร์ด NodeMCU
5	{	
6	pinMode(Y_LED, OUTPUT);	//ตั้งให้ GPIO16 ซึ่งต่อกับแอลอีดีสีเหลืองทำหน้าที่เอาต์พุต
7	pinMode(R_LED, OUTPUT);	//ตั้งให้ GPIO5 ซึ่งต่อกับแอลอีดีสีแดงทำหน้าที่เอาต์พุต
8	pinMode(SW1, INPUT);	//ตั้งให้ GPIO4 ซึ่งต่อกับคิปปสวิตช์บิตที่ 1 ทำหน้าที่เป็นอินพุต
9	}	
10	void loop()	//ฟังก์ชันซึ่งสั่งให้ตัวประมวลผลวนซ้ำไม่รู้จบ
11	{	
12	bool s1 = digitalRead(SW1);	//รับค่าจากคิปปสวิตช์บิตที่ 1



13	if (s1)	//หากค่าจากสวิตช์เป็นตรรกะสูง
14	digitalWrite(Y_LED, HIGH);	//สั่งให้หลอดแอลอีดีสีเหลืองติด
15	else	//แต่ถ้าค่าจากดิปสวิตช์เป็นตรรกะต่ำ
16	digitalWrite(Y_LED, LOW);	//สั่งให้หลอดแอลอีดีสีเหลืองดับ
17		
18	digitalWrite(R_LED, HIGH);	//สั่งให้หลอดแอลอีดีสีแดงติด
19	delay(250);	//หน่วงเวลา 250 มิลลิวินาที
20	digitalWrite(R_LED, LOW);	//สั่งให้หลอดแอลอีดีสีแดงดับ
21	delay(250);	//หน่วงเวลา 250 มิลลิวินาที
22	}	

รูปที่ 9 ตัวอย่างโปรแกรมสำหรับควบคุม NodeMCU Shield อย่างง่าย

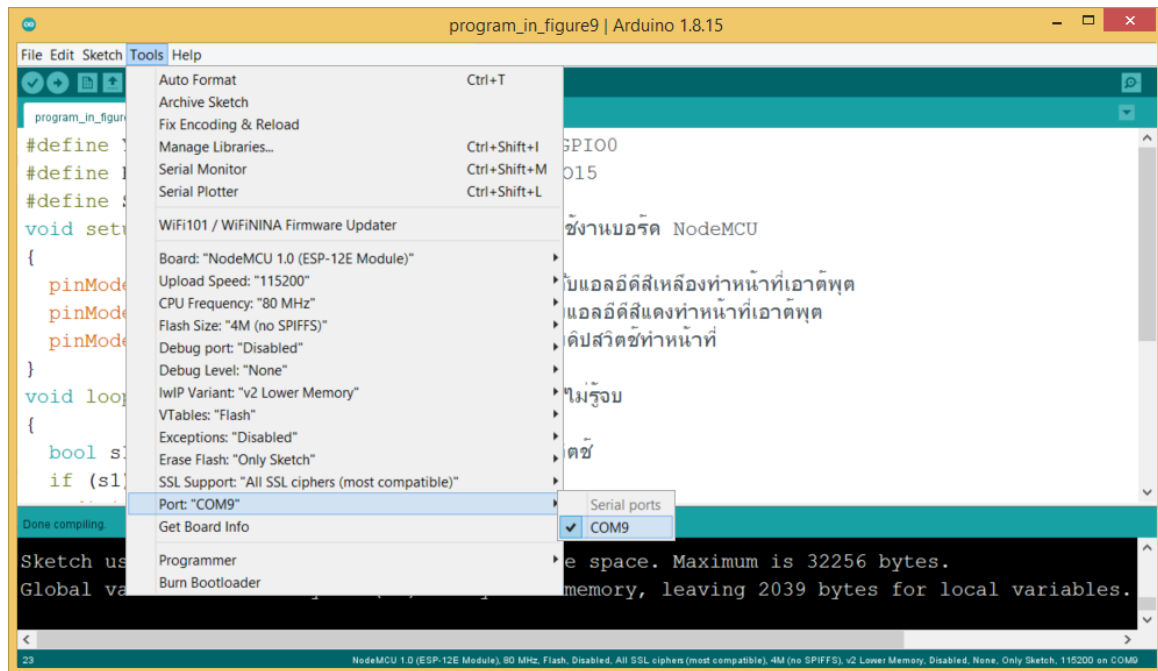
ฟังก์ชัน setup จะถูกเรียกขึ้นมาใช้งานเพียงครั้งเดียวเมื่อบอร์ดประมวลผลเริ่มทำงาน ส่วนฟังก์ชัน loop จะถูกเรียกใช้งานแบบไม่รู้จบตรงไปที่ตัวประมวลผลยังคงได้รับไฟเลี้ยงอยู่ ในการเขียนโปรแกรม ผู้เขียนจะต้องมีการกำหนดหมายเลข GPIO สำหรับแต่ละอุปกรณ์ให้ถูกต้อง ดังตัวอย่างโปรแกรมในรูปที่ 9 โดยจะเห็นว่าโค้ดในบรรทัดที่ 1-3 เป็นการตั้งค่าหมายเลข GPIO ให้กับสัญลักษณ์ในโปรแกรมซึ่งกำหนดขึ้นมาสำหรับใช้แทนแอลอีดีและสวิตช์ ในส่วนของฟังก์ชัน setup จะต้องมีการตั้งค่าทิศทางของ GPIO แต่ละขาเสียก่อน โดยจะเห็นว่า GPIO ขาที่ต่อกับแอลอีดีจะต้องตั้งให้เป็นเอาต์พุต ส่วน GPIO ขาที่ต่อกับดิปสวิตช์จะต้องตั้งให้เป็นอินพุต โค้ดบรรทัดที่ 12 เป็นการใช้ฟังก์ชัน digitalWrite ในการอ่านค่าจากดิปสวิตช์และนำค่าที่ได้มาตรวจสอบ หากพบว่าเป็นตรรกะสูงให้สั่งการให้แอลอีดีสีเหลืองติด แต่หากเป็นตรรกะต่ำให้สั่งให้แอลอีดีสีเหลืองดับ จากนั้นทำการสั่งให้แอลอีดีสีแดงติดเป็นเวลา 250 มิลลิวินาที ตามด้วยสั่งให้แอลอีดีสีแดงดับเป็นเวลา 250 มิลลิวินาที

โปรแกรมที่แสดงในรูปที่ 9 มีการทำงาน คือ ตัวประมวลผลจะรับค่าจากดิปสวิตช์บิตที่ 1 จากนั้นตรวจสอบว่ามีค่าตรรกะสูงหรือไม่ หากเป็นตรรกะสูงจะสั่งให้แอลอีดีสีเหลืองติด แต่หากเป็นตรรกะต่ำจะสั่งให้แอลอีดีสีเหลืองดับ และนอกจากนี้ยังมีการสั่งงานให้แอลอีดีสีแดงสลับกันติดและดับทุก ๆ 250 มิลลิวินาที

## 7. การทดลองสำหรับ Checkpoint1

การทดลองนี้มีวัตถุประสงค์เพื่อให้นักศึกษาหัดการเขียนโปรแกรมบอร์ด NodeMCU อย่างง่าย โดยมีขั้นตอนในการดำเนินการดังนี้

1. เปิดโปรแกรม Arduino IDE ทำการติดตั้งไลบรารี NodeMCU ตามขั้นตอนที่ระบุในหัวข้อที่ 5
2. คัดลอกโปรแกรมควบคุมบอร์ดในรูปที่ 9 มาใส่ในหน้าจอดีเตอร์ของโปรแกรม Arduino IDE จากนั้นกดปุ่มส่งคอมไพล์
3. เชื่อมต่อบอร์ด NodeMCU กับเครื่องคอมพิวเตอร์ผ่านพอร์ต USB จากนั้นทำการตั้งค่าหมายเลขพอร์ตอนุกรมเสมือนสำหรับอัปโหลดโปรแกรมใน Arduino IDE ดังแสดงตัวอย่างในรูปที่ 10 โดยให้เลือกหมายเลข COM พอร์ตให้ตรงกับหมายเลขที่เครื่องของนักศึกษาตรวจพบในขั้นตอนที่ 5 ของหัวข้อที่ 2 (ตัวอย่างในรูปที่ 10 เป็นการตั้งค่าไว้ให้เท่ากับ COM9)
4. ทำการอัปโหลดโปรแกรมลงสู่บอร์ด NodeMCU Shield
5. สังเกตการทำงานของบอร์ด NodeMCU จากนั้นเลื่อนดิปสวิตช์บิตที่ 1 ขึ้นและลง ทำการสังเกตผลลัพธ์ที่แอลอีดีสีเหลืองและแอลอีดีสีแดง บันทึกผลการทดลอง
6. เขียนโปรแกรมควบคุมตัวใหม่ โดยกำหนดให้ความต้องการของโปรแกรม คือ ตัวประมวลผลตรวจสอบสถานะของดิปสวิตช์จำนวน 4 บิต และสั่งควบคุมการติดดับของแอลอีดีทั้งสามดวง ดังตารางที่ 1



รูปที่ 10 การเลือกหมายเลข COM port ของบอร์ด NodeMCU ในโปรแกรม Arduino IDE

7. คอมไพล์โปรแกรม และอัปโหลดโปรแกรมลงสู่บอร์ด NodeMCU บันทึกผลการทดลอง
8. ในการ checkpoint กับผู้ตรวจหรือ TA ให้อธิบายการทำงานของโค้ดในตัวรายงานและอธิบายผลการทดลองแต่ละขั้นตอนอย่างละเอียด

ตารางที่ 1 รูปแบบการติดต่อบนของแอลอีดีทั้งสามตัวใน checkpoint1 ซึ่งควบคุมโดยการตั้งค่าจากคิปลิสต์

หมายเลข GPIO และค่าสถานะตรรกะที่อ่านได้				หน้าที่การทำงานของโปรแกรม
16	5	13	12	
1	X	X	X	แอลอีดีสีแดงกระพริบติดดับทุก ๆ 1 วินาที
0	0	0	0	แอลอีดีทุกดวงดับ
0	0	0	1	แอลอีดีสีแดงติดพร้อมสีเหลือง ส่วนแอลอีดีสีเขียวดับ
0	0	1	0	แอลอีดีสีแดงติดพร้อมสีเขียว ส่วนแอลอีดีสีเหลืองดับ
0	0	1	1	แอลอีดีสีเหลืองติดพร้อมสีเขียว ส่วนแอลอีดีสีแดงดับ
0	1	0	0	แอลอีดีทุกดวงกระพริบติดดับพร้อมกันทุก ๆ 1 วินาที
หมายเหตุ ค่า X คือ don't care term				

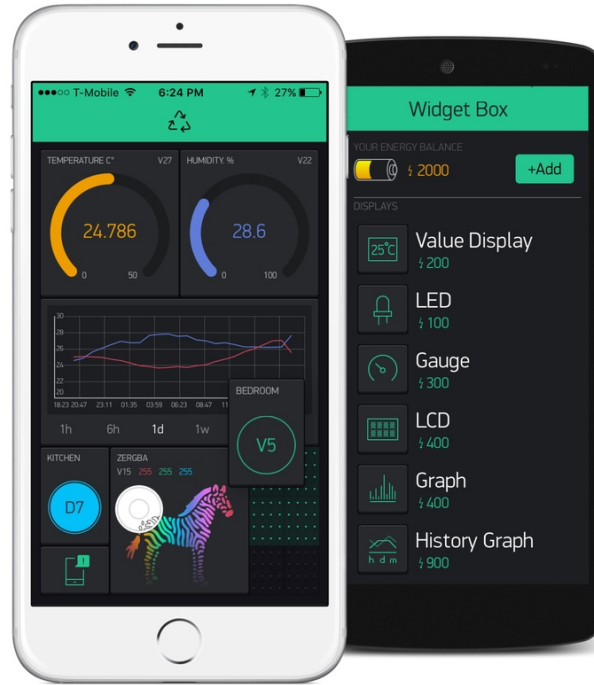
## 8. แนะนำ Blynk

Blynk เป็นแพลตฟอร์ม IoT ซึ่งทำงานบนระบบปฏิบัติการ iOS และ Android ซึ่งสามารถติดต่อกับฮาร์ดแวร์ได้หลายแพลตฟอร์ม ไม่ว่าจะเป็น Arduino, Raspberry Pi หรือแม้แต่ NodeMCU ผู้ใช้สามารถเขียนโปรแกรมเพื่อควบคุมฮาร์ดแวร์ผ่านโทรศัพท์มือถือได้หากวงจรฮาร์ดแวร์ที่ใช้สามารถติดต่อสื่อสารผ่านระบบอินเทอร์เน็ตได้ ในการทดลองนี้จะให้นักศึกษาใช้แพลตฟอร์ม Blynk ควบคุมฮาร์ดแวร์ NodeMCU ผ่านโทรศัพท์มือถือระบบปฏิบัติการ iOS หรือ Android

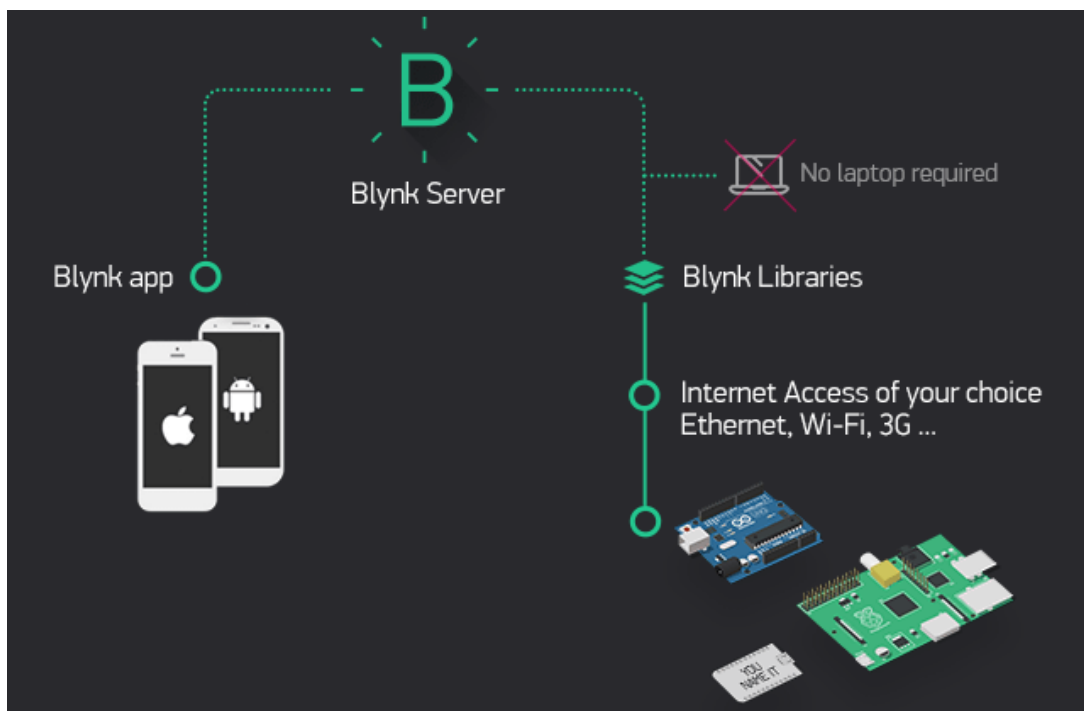
การทำงานของ Blynk ประกอบด้วยองค์ประกอบ 3 ส่วน ได้แก่ Blynk App ซึ่งทำงานอยู่บนโทรศัพท์มือถือ Blynk Server ทำหน้าที่เป็นตัวเชื่อมการสื่อสารระหว่างโทรศัพท์และฮาร์ดแวร์ โดยผู้ใช้สามารถ



เลือกว่าจะใช้ Blynk Cloud หรือสร้าง Blynk Server ขึ้นมาใช้งานเอง (ในแลบนี้จะใช้ Blynk Cloud เป็นหลัก) และองค์ประกอบสุดท้าย คือ Blynk Libraries ซึ่งเป็นคลังโปรแกรมซึ่งสนับสนุนอุปกรณ์ฮาร์ดแวร์ของแต่ละแพลตฟอร์ม ซึ่งในกรณีของการทดลองนี้จะใช้แพลตฟอร์มฮาร์ดแวร์ของ NodeMCU ESP8266 เป็นหลัก



รูปที่ 11 โปรแกรม Blynk ซึ่งทำงานอยู่บนระบบปฏิบัติการ iOS (ซ้าย) และ Android (ขวา)

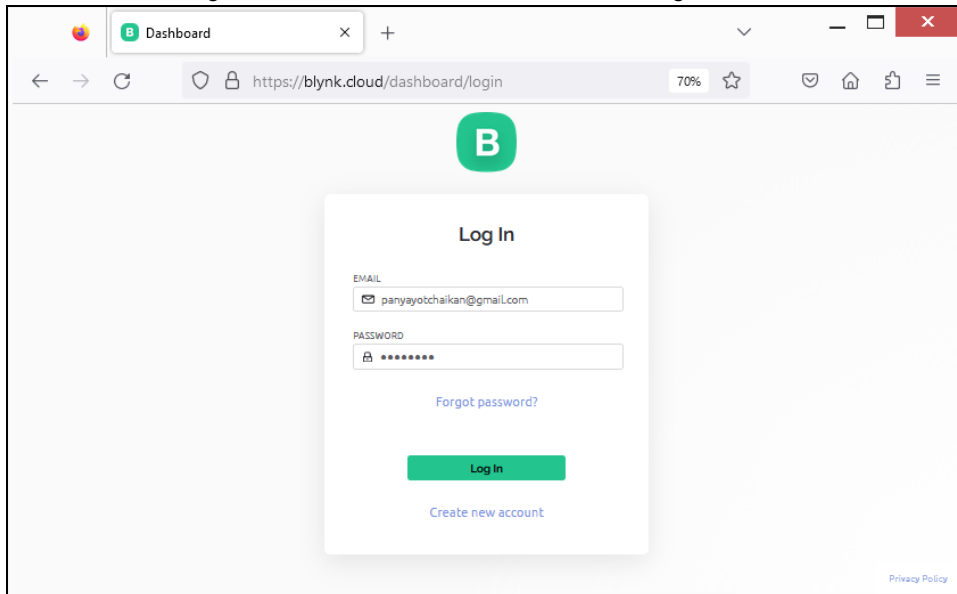


รูปที่ 12 การทำงานของแพลตฟอร์ม Blynk

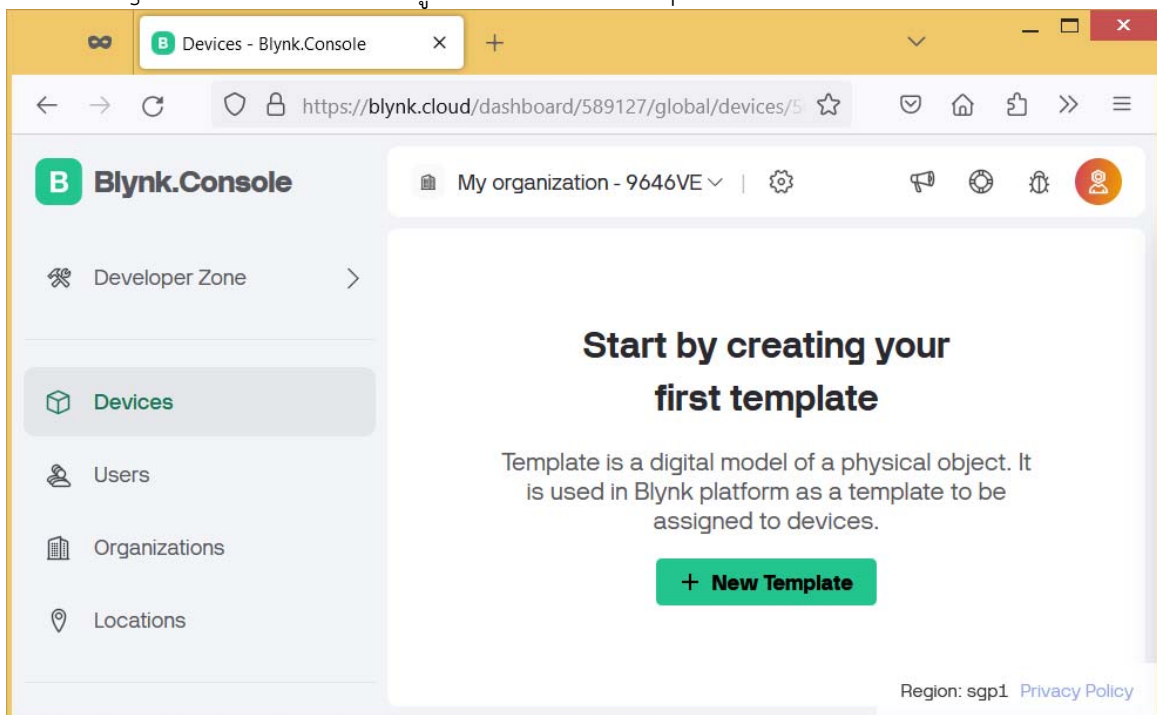
## 9. การทดลองสำหรับ Checkpoint2

ในหัวข้อนี้จะเป็นการยกตัวอย่างการนำ Blynk มาควบคุมการทำงานของวงจรซึ่งเชื่อมต่ออยู่กับ NodeMCU โดยใช้ Widget จำนวน 3 ตัว ได้แก่ Button, Slider และ LED ซึ่งผู้ใช้งาน Blynk สามารถเลือกได้จาก Widget Box โปรแกรมทางฟาก NodeMCU มีการทำงาน คือ อ่านค่าจากดิปสวิตซ์ซึ่งต่อกับ GPIO4 นำค่าที่ได้มาแสดงผลบน LED ซึ่งอยู่บนหน้าจอโทรศัพท์ ตัวแอลอีดีสีแดงถูกควบคุมความสว่างได้โดยใช้ Slider ในขณะที่แอลอีดีสีเขียวถูกควบคุมการเปิดปิดได้จาก Button บนหน้าจอโทรศัพท์มือถือ

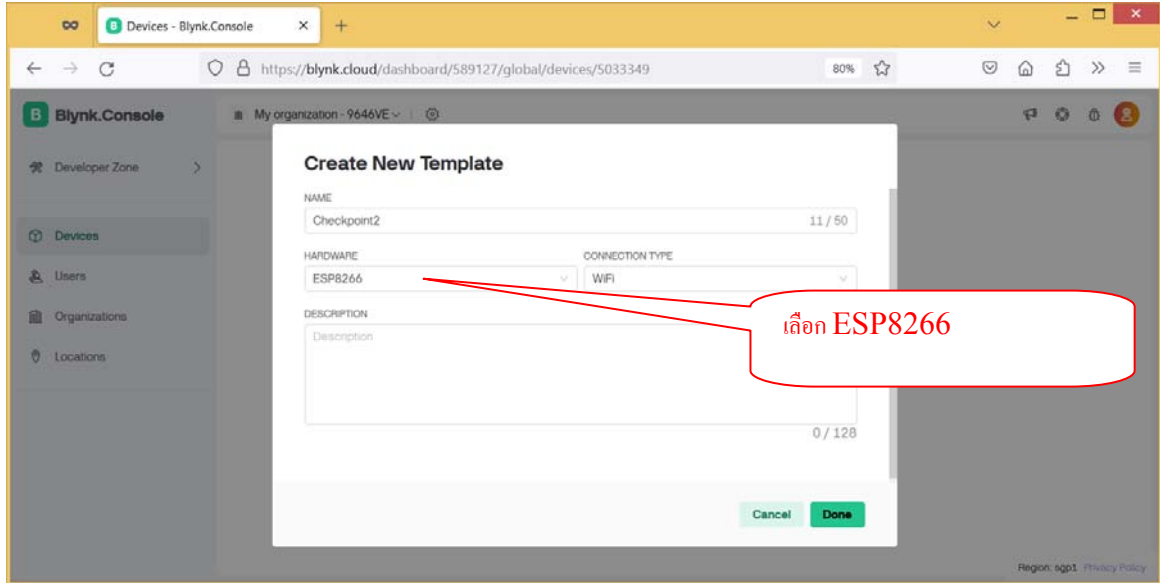
9.1 เปิดเว็บเบราว์เซอร์ จากนั้นไปที่ลิงก์ <https://blynk.cloud/> ที่หน้าจอ Login ป้อนอีเมลและรหัสผ่าน หากนักศึกษายังไม่มี Login ให้สร้างแอคเคาต์ขึ้นมาก่อน แล้วค่อย Login



9.2 เมื่อ Login สำเร็จ จะขึ้นหน้าจอ ดังรูป ให้คลิกที่ New Template



9.3 ป้อนชื่อ Template เป็น Checkpoint 2 และเลือกฮาร์ดแวร์เป็น ESP8266 และประเภทการเชื่อมต่อเป็น WiFi



NAME: Checkpoint2 (11 / 50)

HARDWARE: ESP8266

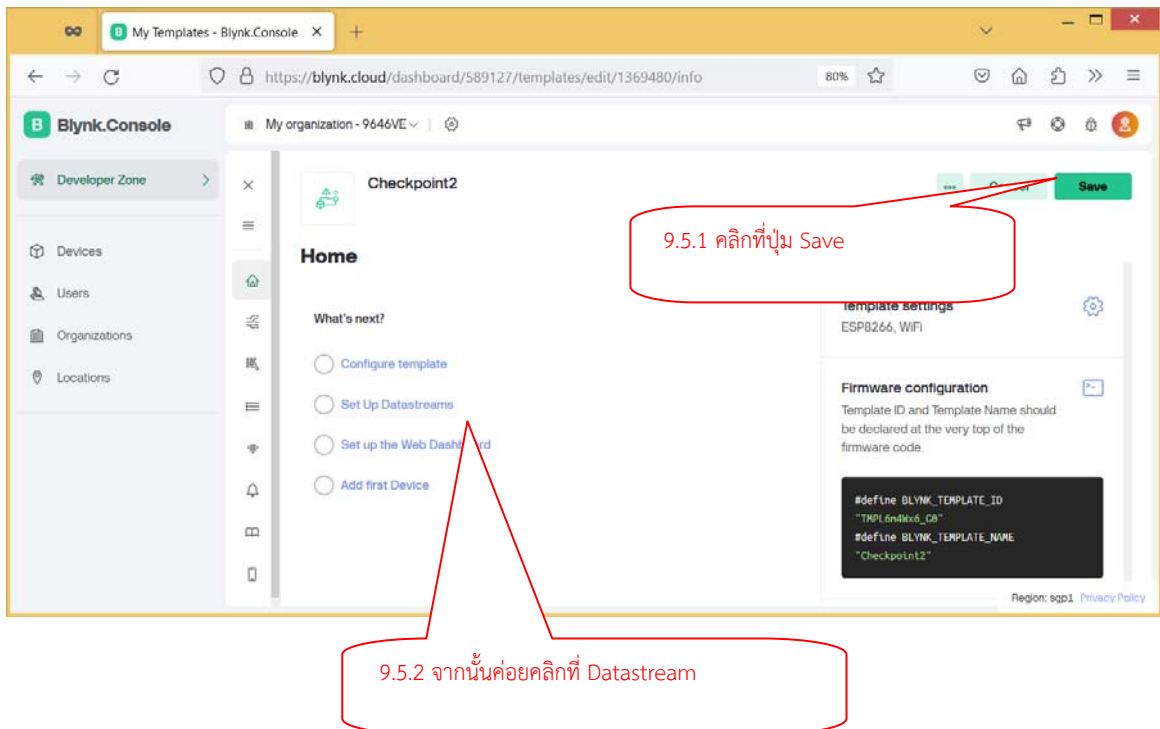
CONNECTION TYPE: WiFi

DESCRIPTION: Description (0 / 128)

Buttons: Cancel, Done

Region: sgp1 Privacy Policy

9.5 กดปุ่ม Save เพื่อจัดเก็บ Template ที่สร้างขึ้น จากนั้นกดที่ Setup Datastreams



Checkpoint2

Home

What's next?

- Configure template
- Set Up Datastreams
- Set up the Web Dashboard
- Add first Device

Buttons: Save

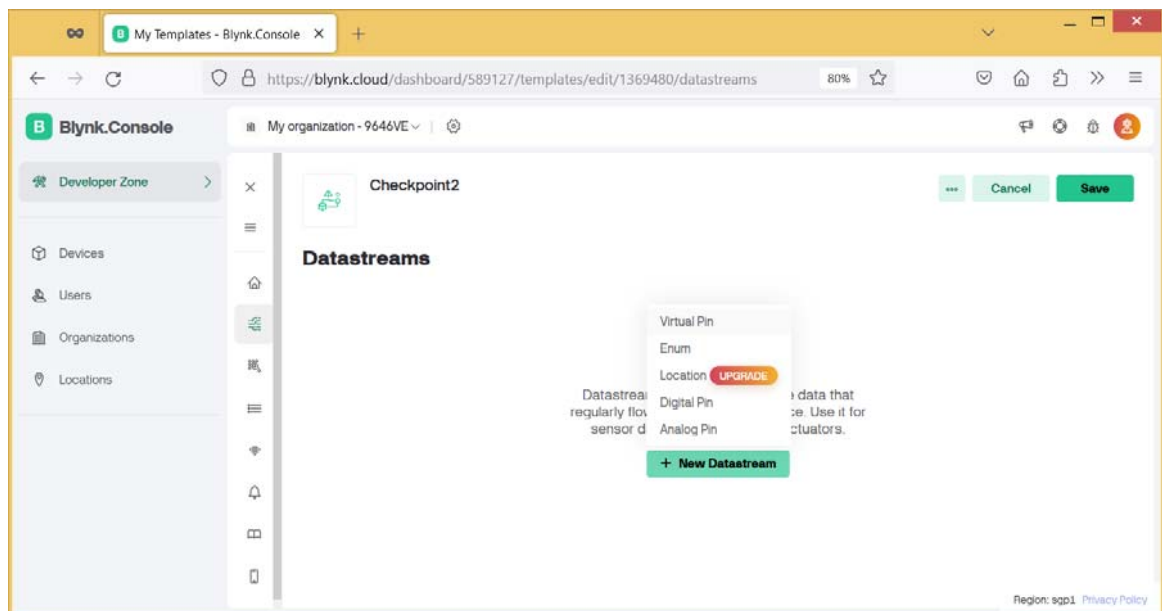
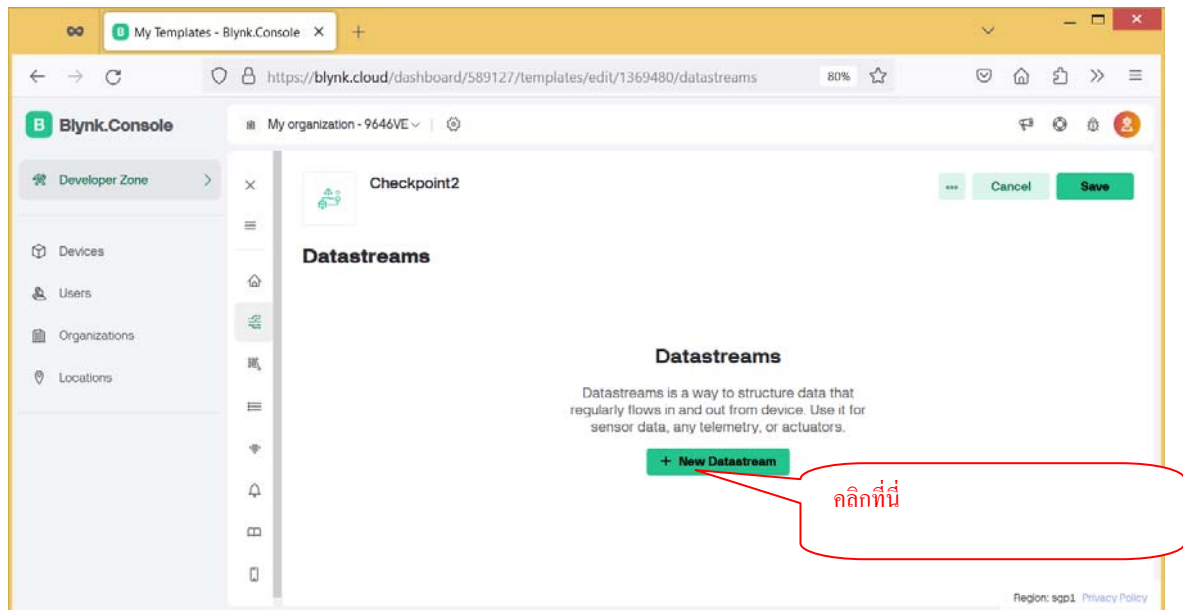
template settings: ESP8266, WiFi

Firmware configuration

Template ID and Template Name should be declared at the very top of the firmware code.

```
#define BLYNK_TEMPLATE_ID  
"TNP16nd4x6_C0"  
#define BLYNK_TEMPLATE_NAME  
"Checkpoint2"
```


Region: sgp1 Privacy Policy



9.7 กดที่ปุ่ม New Datastream เลือกที่ Virtual Pin จะขึ้นไดอะล็อกบ็อกซ์ดังรูป จากนั้นเปลี่ยนชื่อเป็น Red LED V0

### Virtual Pin Datastream

**General**   Expose to Automations

NAME:    ALIAS:  

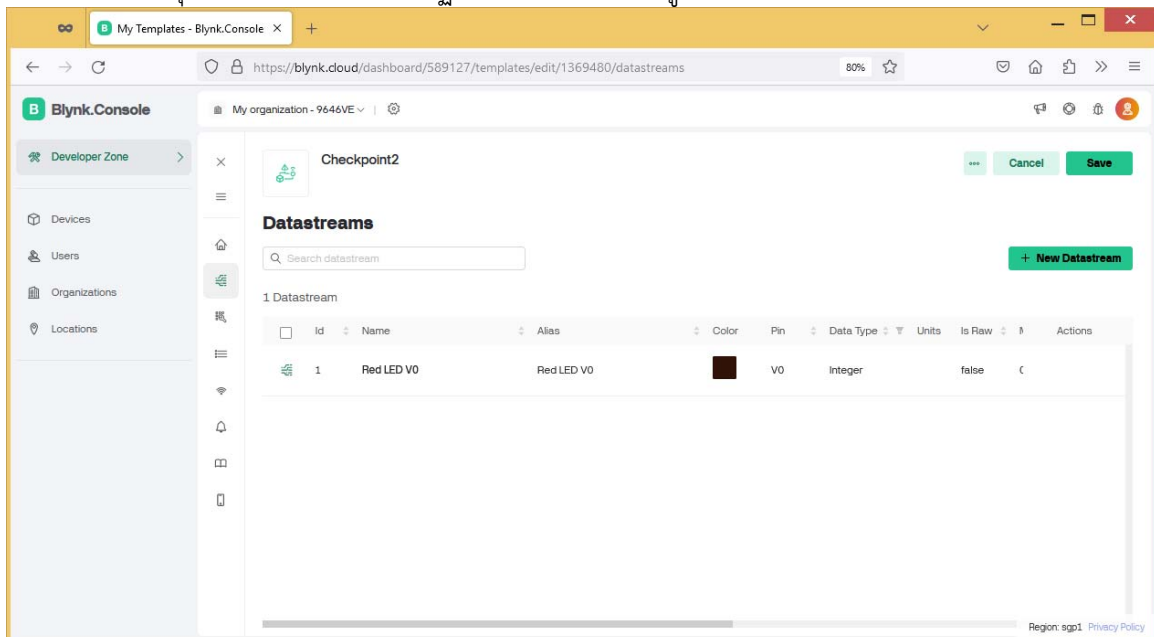
PIN:    DATA TYPE:

UNITS:


MIN:    MAX:    DEFAULT VALUE:

☐ Enable history data

9.8 หลังจากกดปุ่ม Create แล้ว จะปรากฏ Datastream ขึ้นดังรูป



The screenshot shows the Blynk Console interface. On the left is a sidebar with navigation options: Developer Zone, Devices, Users, Organizations, and Locations. The main area displays the 'Datastreams' section for a device named 'Checkpoint2'. At the top right of this section are 'Cancel' and 'Save' buttons. Below is a search bar and a '+ New Datastream' button. A table lists the existing datastreams:

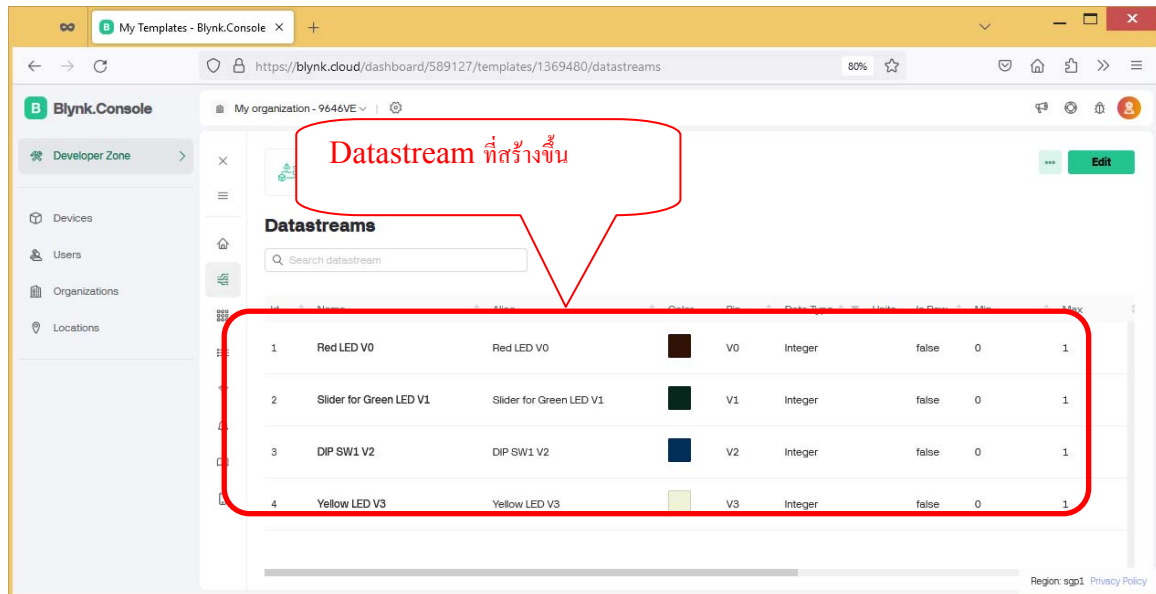
Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Actions
1	Red LED V0	Red LED V0		V0	Integer		false	(edit icon)

At the bottom right, it shows 'Region: sgp1' and a 'Privacy Policy' link.

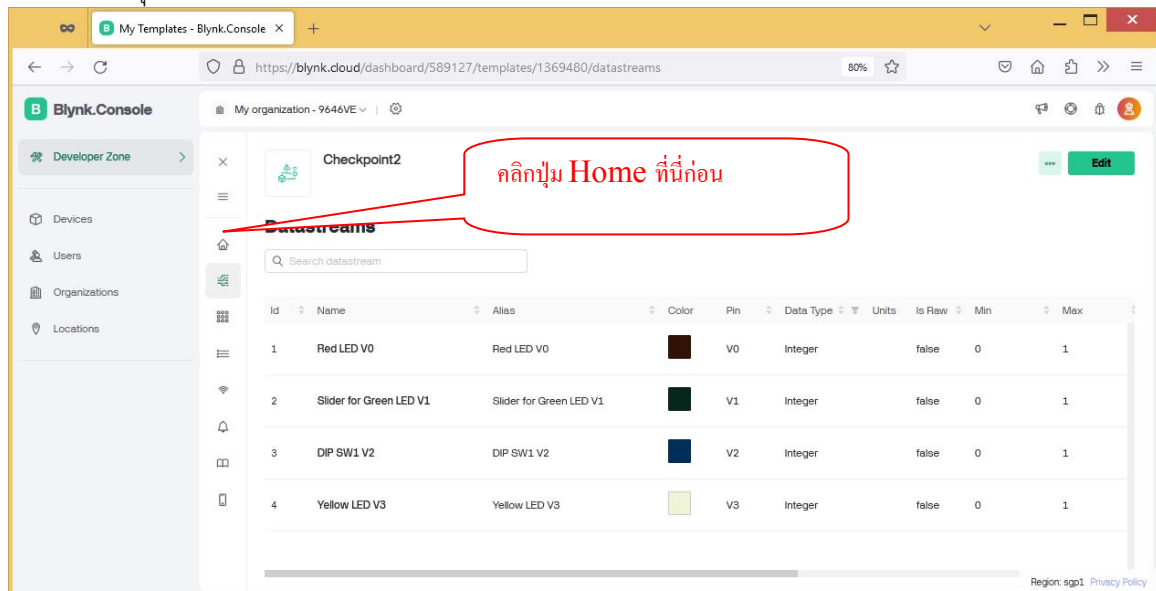
### 9.9 ป้อน Datastream ขึ้นอีก 3 รายการ ดังนี้

- Slider for Green LED V1
- DIP SW1 V2
- Yellow LED V3

ดังรูป จากนั้นกดปุ่ม Save

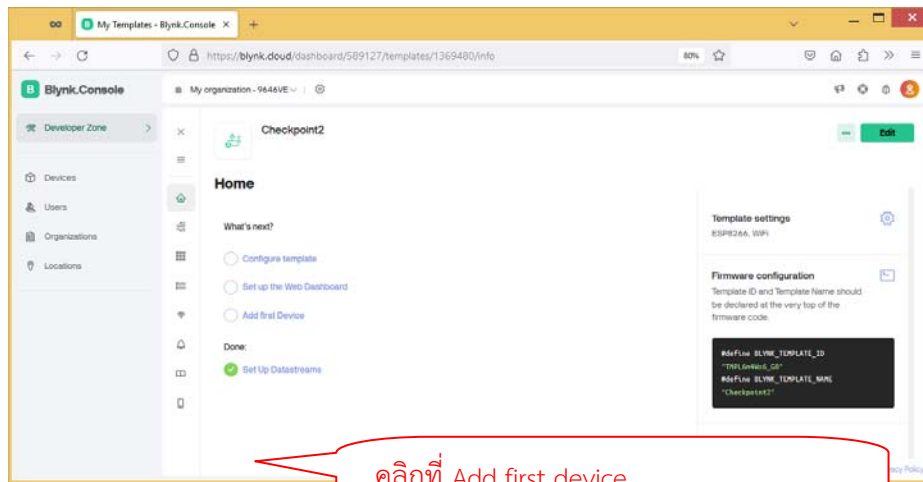


### 9.10 คลิกที่ปุ่ม Home จากนั้นคลิกที่ Add first Device



### 9.11 คลิกที่ Add first Device





9.12 ที่ใดจะคลิกบ็อกซ์ New Device ให้ใส่ชื่อ ในที่นี้ใช้ชื่อว่า lab18\_ckpt2 กดที่ปุ่ม Create

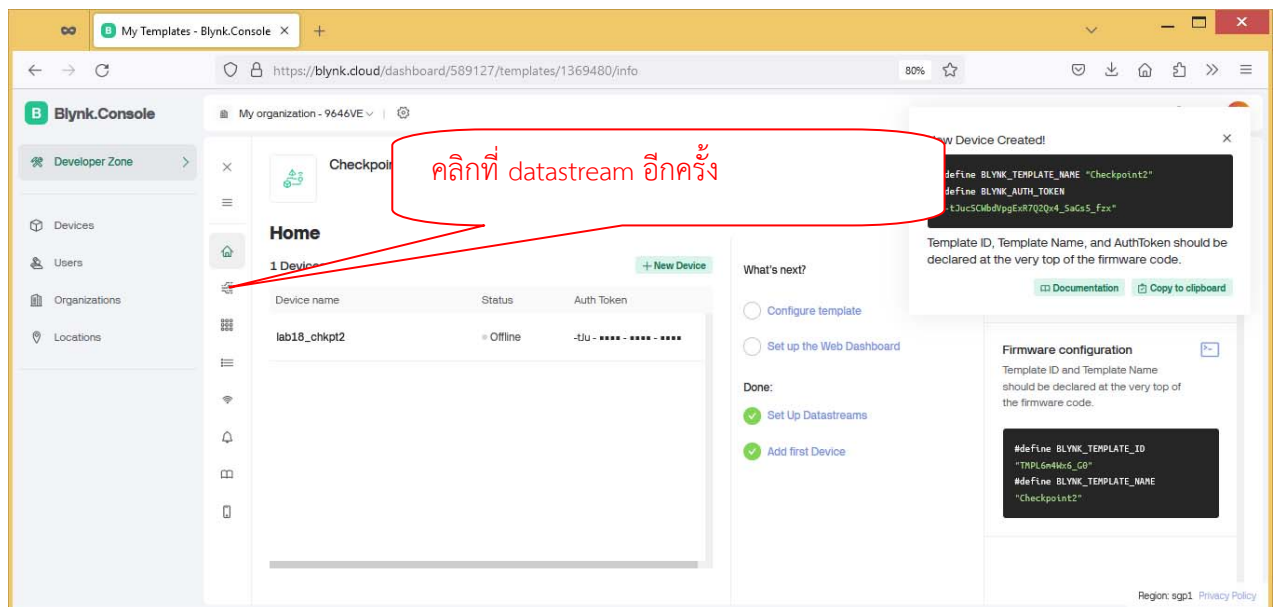
### New Device

DEVICE NAME

lab18\_ckpt2
12 / 50

Cancel
Create

9.13 จะขึ้น หน้าจอดังรูป ซึ่งจะเห็นว่าการระบุตัวตนนี้มี 1 device ใน Template ที่สร้างขึ้น



9.14 ไปที่ ปุ่ม datastream อีกครั้ง จากนั้นคลิก

My Templates - Blynk.Console

https://blynk.cloud/dashboard/589127/templates/edit/1369480/datastreams

My organization - 9646VE

**Datastreams**

4 Datastreams

Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Max	Actions
1	Red LED V0	Red LED V0	Red	V0	Integer		false	0		
2	Slider for Green LED V1	Slider for Green LED V1	Green	V1	Integer		false	0		
3	DIP SW1 V2	DIP SW1 V2	Blue	V2	Integer		false	0		
4	Yellow LED V3	Yellow LED V3	Yellow	V3	Integer		false	0		

Region: sgp1 Privacy Policy

9.15 แก้ไข Datastream ที่ชื่อ Slider for Green LED V1 ใหม่ ป้อนค่า MAX เป็น 1023 แล้วกด save

**Virtual Pin Datastream**

General Expose to Autom

NAME: Slider for Green LED V1 ALIAS: Slider for Green LED V1

PIN: V1 DATA TYPE: Integer

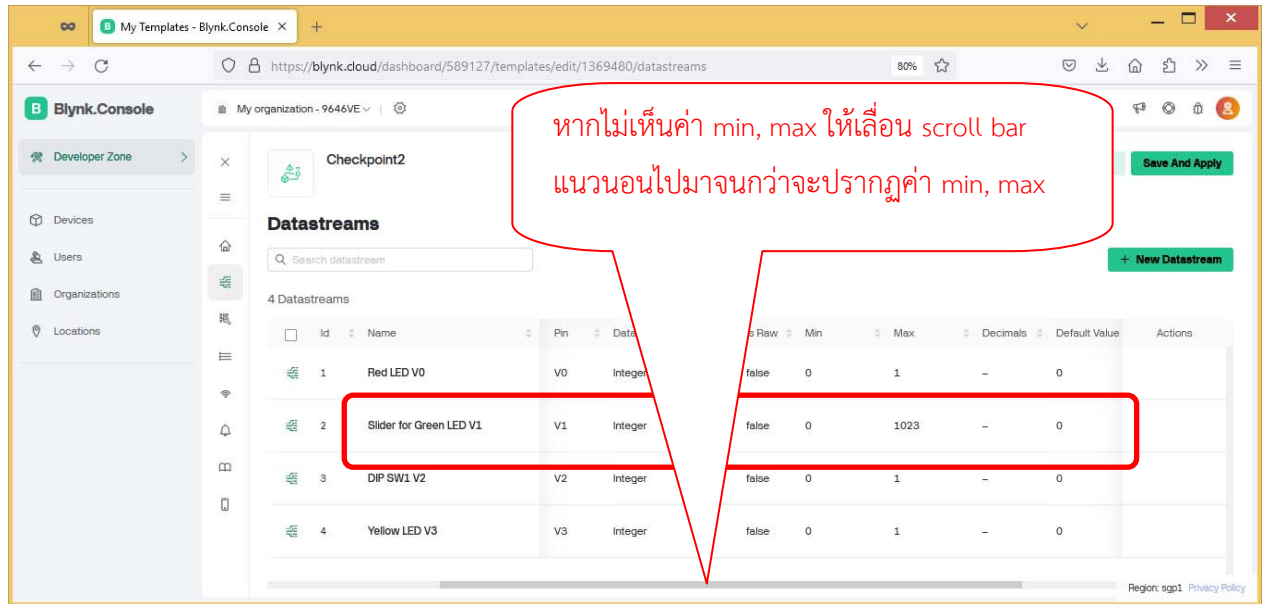
UNITS: None

MIN: 0 MAX: 1023 DEFAULT VALUE: 0

Cancel Save

หากไม่เห็นค่า min, max ให้เลื่อน scroll bar ลงมา

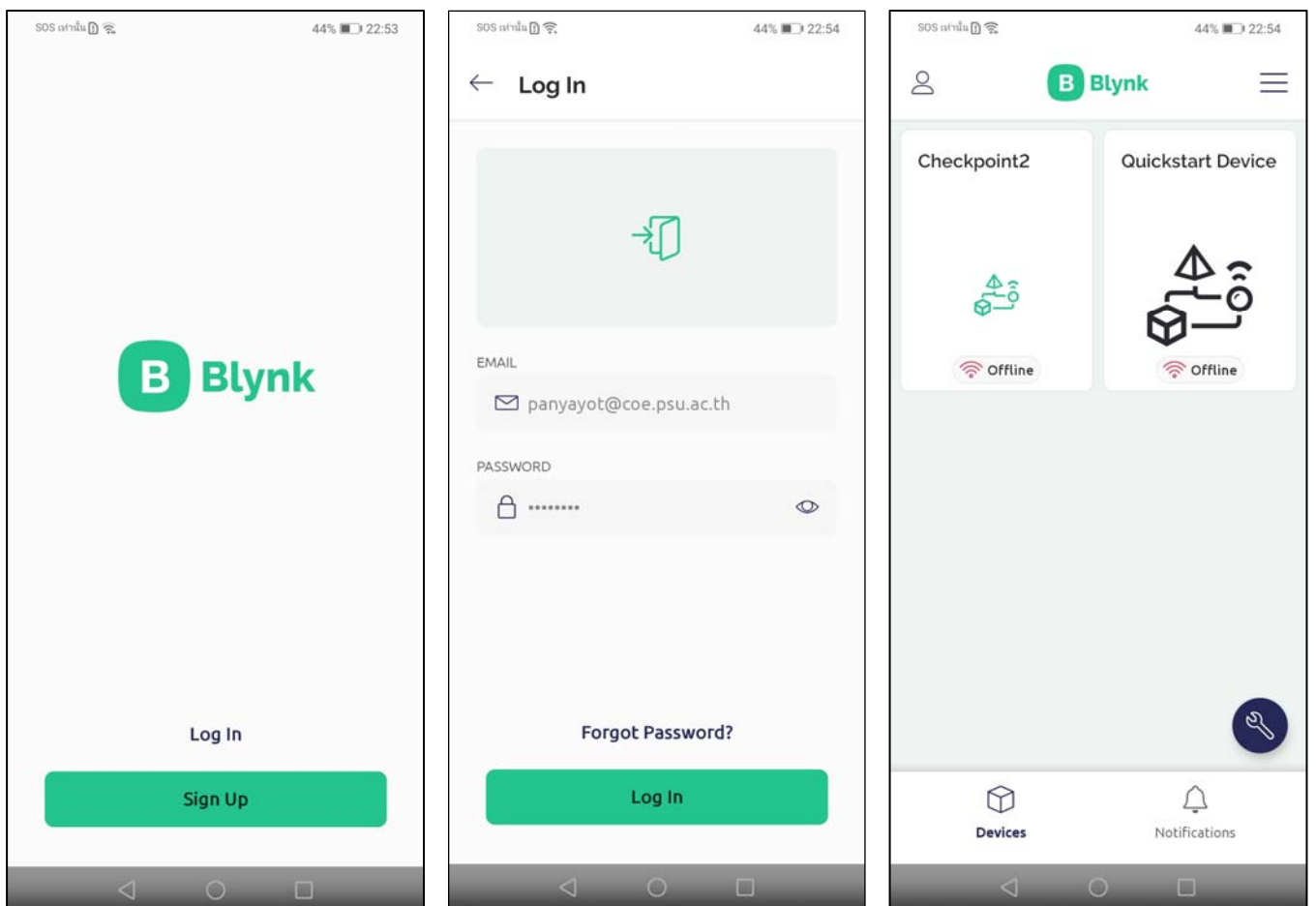
9.16 หลังจากแก้ไข Slider for Green LED V1 เสร็จแล้ว สังเกตค่า MAX ที่เปลี่ยนไป



หากไม่เห็นค่า min, max ให้เลื่อน scroll bar  
แนวนอนไปมาจนกว่าจะปรากฏค่า min, max

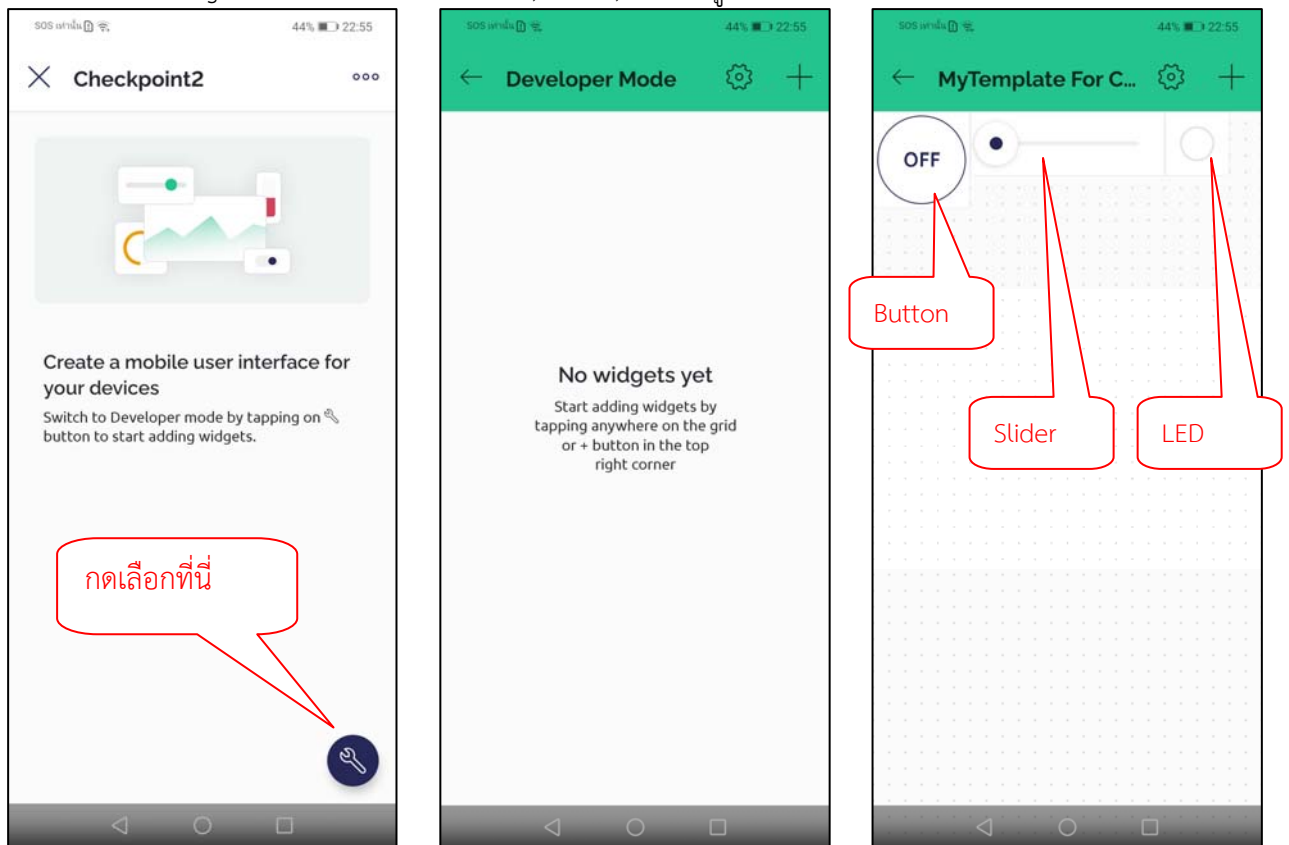
	Id	Name	Pin	Data	Raw	Min	Max	Decimals	Default Value	Actions
1		Red LED V0	V0	Integer	false	0	1	-	0	
2		Slider for Green LED V1	V1	Integer	false	0	1023	-	0	
3		DIP SW1 V2	V2	Integer	false	0	1	-	0	
4		Yellow LED V3	V3	Integer	false	0	1	-	0	

9.17 ติดตั้ง Blynk IoT บนโทรศัพท์มือถือ จากนั้น Login จะเห็น Template ที่สร้างไว้ใน Web browser

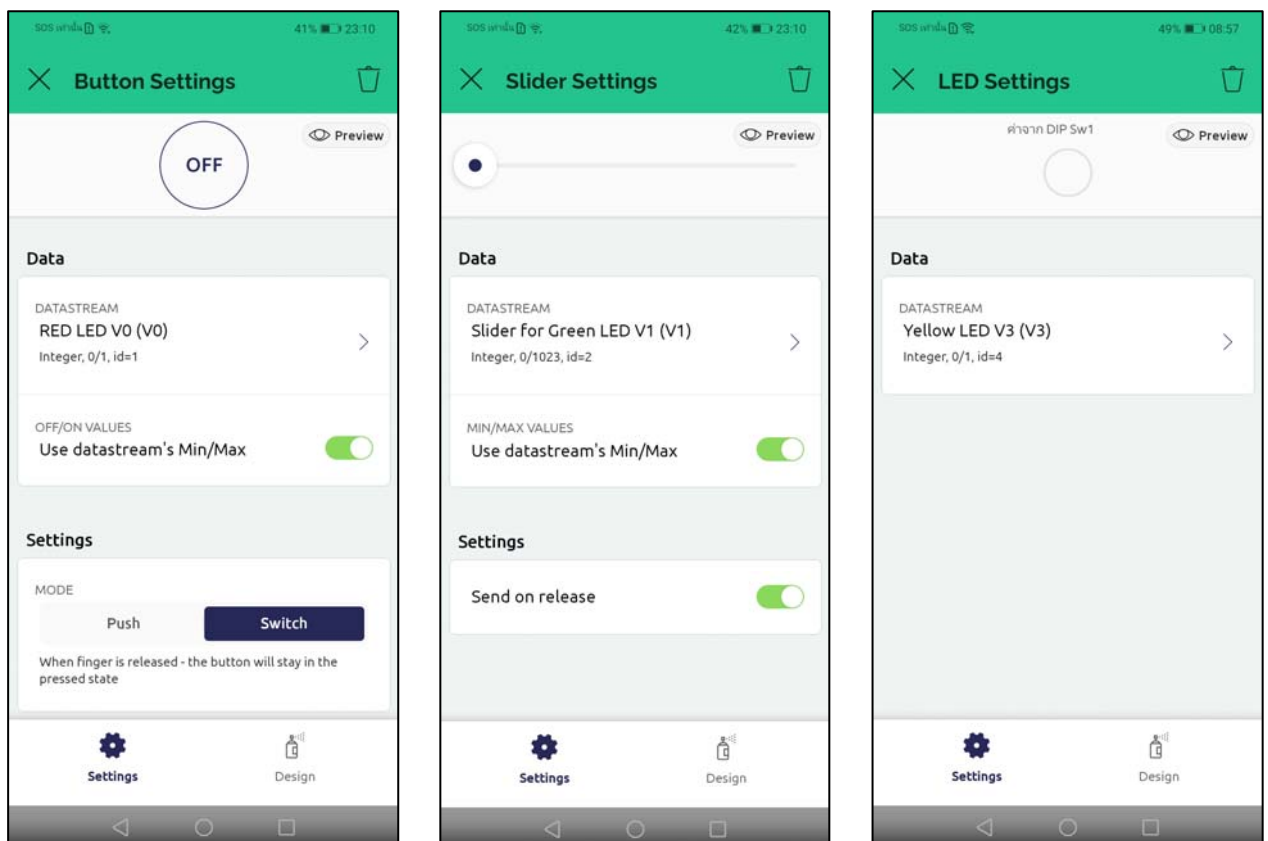


9.18 เลือก Device ที่ได้สร้างไว้แล้วใน Web browser ในที่นี้คือ Checkpoint 2

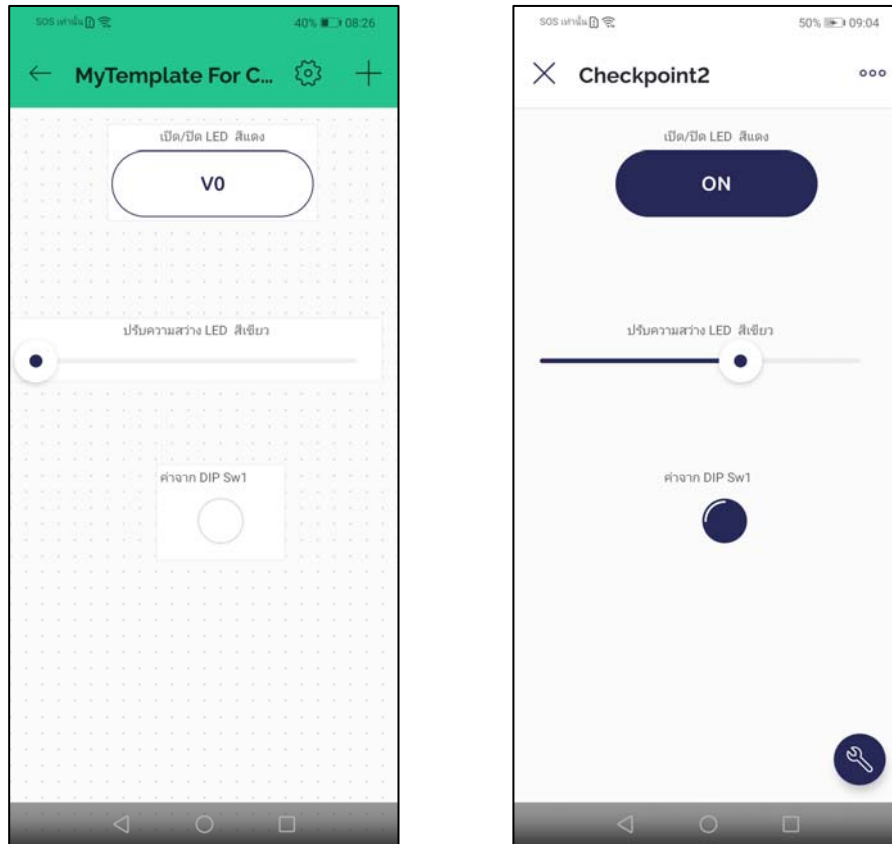
จากนั้นเพิ่ม Widget สามรายการ ได้แก่ Button, Slider, LED ดังรูป



9.19 ตั้งค่า Setting ของ Button, Slider และ LED ดังรูป



9.20 เมื่อตั้งค่าของ Widget ทั้งสามเสร็จ ก็ให้ปรับแต่งหน้าตาของโปรแกรมพร้อมกับแก้ไขคำอธิบายแต่ละปุ่มเป็นดังรูป



9.21 จากนั้นให้เปิดโปรแกรม Arduino IDE จากนั้นสร้างโปรเจกต์ใหม่ และคัดลอกโปรแกรมต่อไปนี้ลงไป หน้าต่าง Editor ของ Arduino IDE

```

1  #define Y_LED 0
2  #define R_LED 15
3  #define G_LED 2
4  #define SW1 4
5
6  /*****
7   This is a simple demo of sending and receiving some data.
8   Be sure to check out other examples!
9   *****/
10
11  #define BLYNK_TEMPLATE_ID "TPdL6fVql04m0"
12  #define BLYNK_TEMPLATE_NAME "MyTemplate for Checkpoint 2"
13  #define BLYNK_AUTH_TOKEN "kRms_Td2Xrzade7bC2C2-xkW71Cy3UEEI"
14
15  /* Comment this out to disable prints and save space */
16  #define BLYNK_PRINT Serial
17  #include <ESP8266WiFi.h>
18  #include <BlynkSimpleEsp8266.h>
19
20  // Your WiFi credentials.
21  // Set password to "" for open networks.

```

```

22 char ssid[] = "CoEIoT";
23 char pass[] = "iot.coe.psu.ac.th";
24 BlynkTimer timer;
25
26 // This function is called every time the Virtual Pin 0 state changes
27 BLYNK_WRITE(V0)
28 {
29     // Set incoming value from pin V0 to a variable
30     int value = param.asInt();
31     if (value == 1)
32         digitalWrite(R_LED, HIGH);
33     else
34         digitalWrite(R_LED, LOW);
35 }
36
37 BLYNK_WRITE(V1)
38 {
39     int value = param.asInt();
40     analogWrite (G_LED, value);
41 }
42
43 // This function is called every time the device is connected to the Blynk.Cloud
44 BLYNK_CONNECTED()
45 {
46     // Change Web Link Button message to "Congratulations!"
47     Blynk.setProperty(V1 0 0 ,"offImageUrl", "https://static-
48 image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
49     Blynk.setProperty(V1 0 0 ,"onImageUrl", "https://static-
50 image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.png");
51     Blynk.setProperty(V100, "url", "https://docs.blynk.io/en/getting-started/what-do-i-need-to-blynk/how-quickstart-
52 device-was-made");
53 }
54
55 // This function sends Arduino's uptime every second to Virtual Pin 2.
56
57 void ReadSW()
58 {
59     uint8_t d = digitalRead(SW1);
60     Blynk.virtualWrite(V3,d);
61 }
62
63 void setup()
64 {
65     pinMode(Y_LED,OUTPUT);
66     pinMode(G_LED,OUTPUT);
67     pinMode(R_LED,OUTPUT);
68     pinMode(SW1,INPUT);
69     Serial.begin(115200);
70     Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
71     // Setup a function to be called every 100 ms
72     timer.setInterval(100L, ReadSW);

```

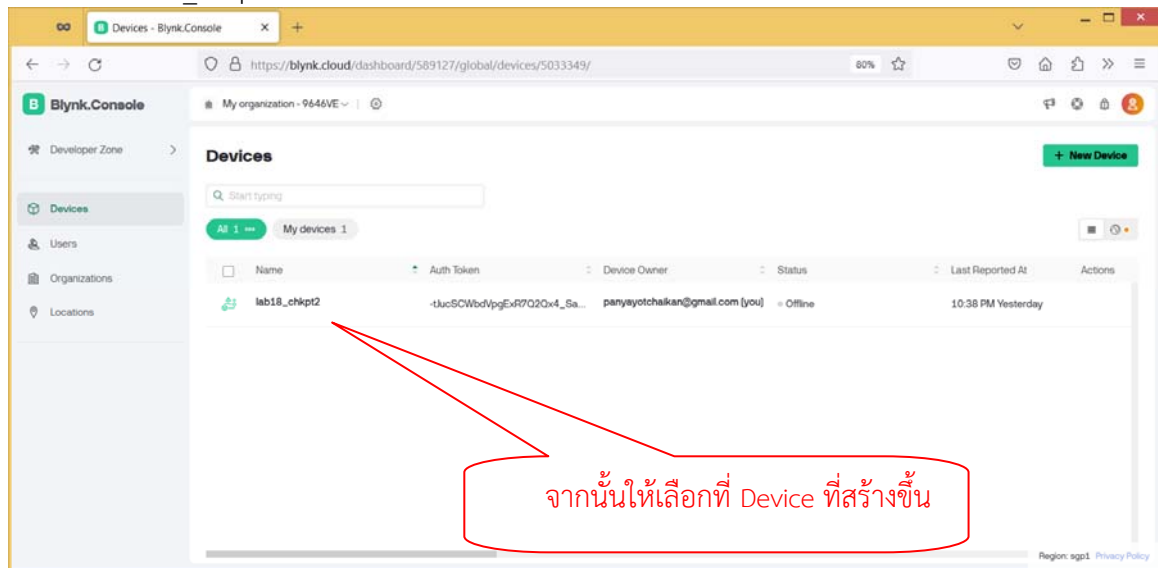


```

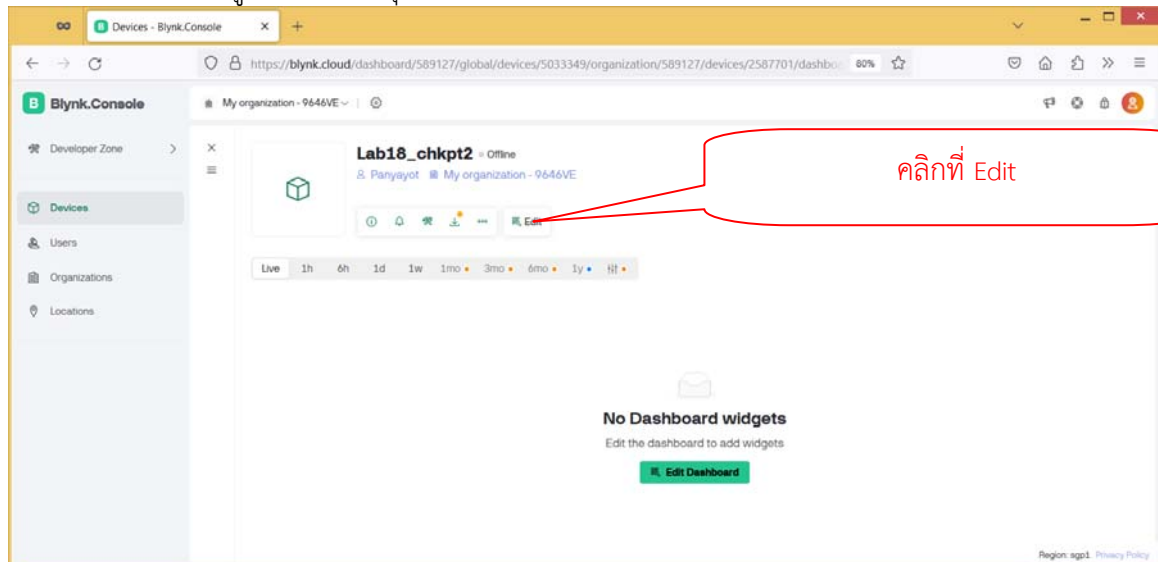
73 }
74
75 void loop()
76 {
77   Blynk.run();
78   timer.run();
79 }

```

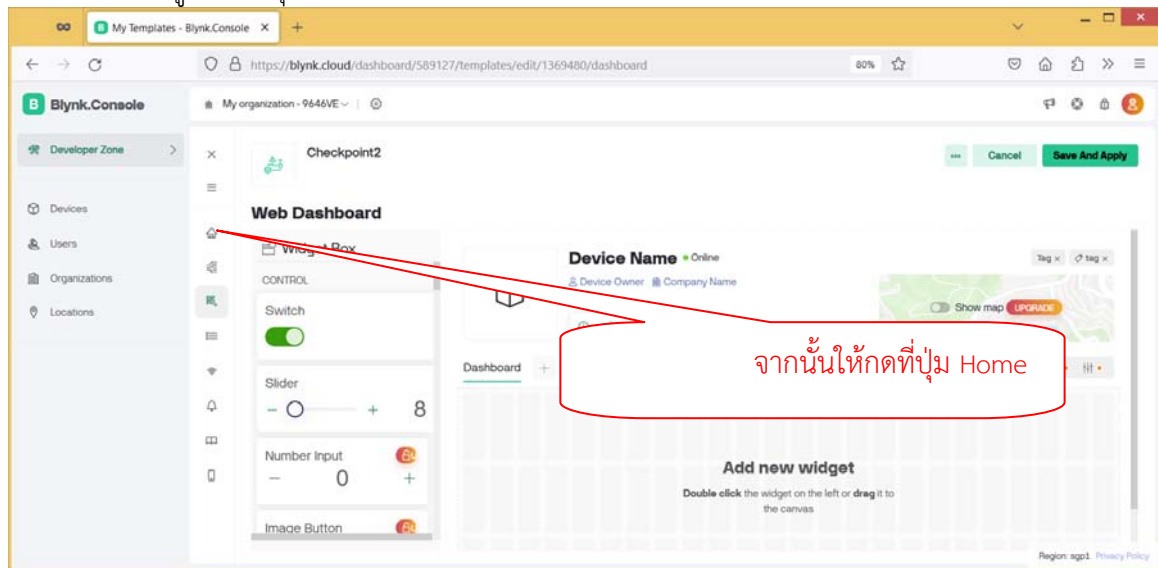
9.22 กลับมาที่เว็บเบราว์เซอร์ที่หน้าจอของ Blynk กดที่ปุ่มค้นหาด้านซ้ายมือ จากนั้นเลือก Device ที่สร้างขึ้น (ในที่นี้คือ lab18\_chkpt2)



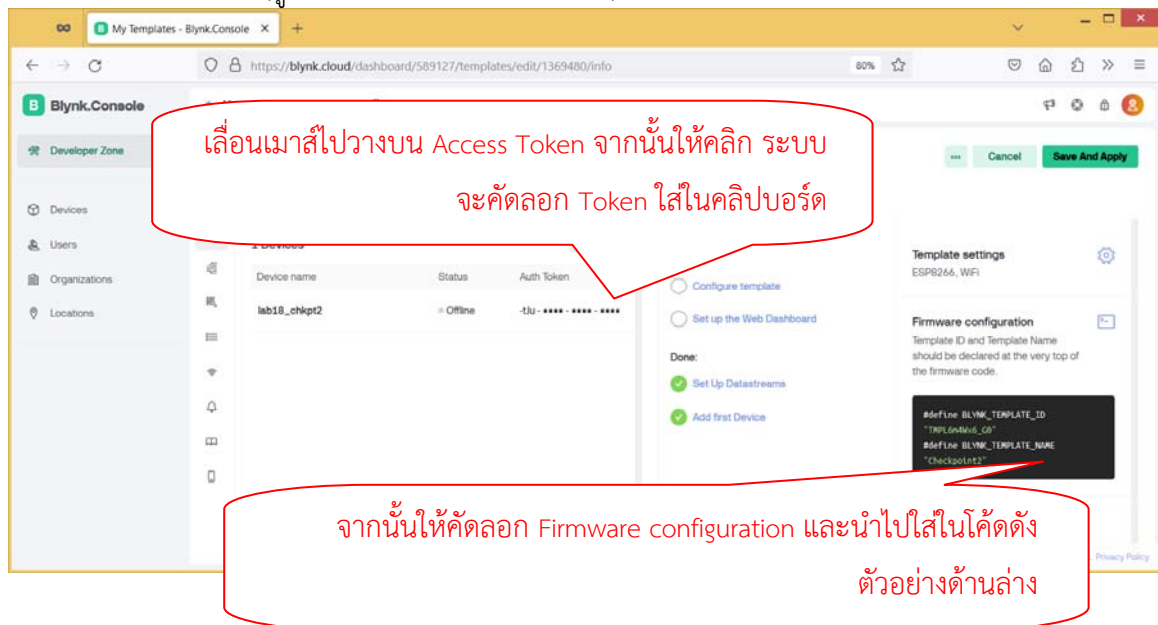
9.23 จะขึ้นหน้าจอดังรูป ให้เลือกกดปุ่ม Edit



จะขึ้นหน้าจอตั้งรูป ให้กดปุ่ม Home



9.24 จากนั้นให้คัดลอก Token แล้วนำไปแทนที่ในส่วนของ Authentication token ในหน้าจอ Editor ของโปรแกรม Arduino IDE (ดูโค้ดบรรทัดที่ ในขั้นตอนที่ 9.21)



ตัวอย่าง Token ที่ต้องนำไปวางใน Arduino IDE แทนที่ Authentication Token เดิม

```
#define BLYNK_TEMPLATE_ID "TMPL6PUo5SK"
#define BLYNK_TEMPLATE_NAME "PanyayotTemplate"
#define BLYNK_AUTH_TOKEN "JYDqSP45kgldaddNvKKZkYEAZz-_jiSoh5w"
```

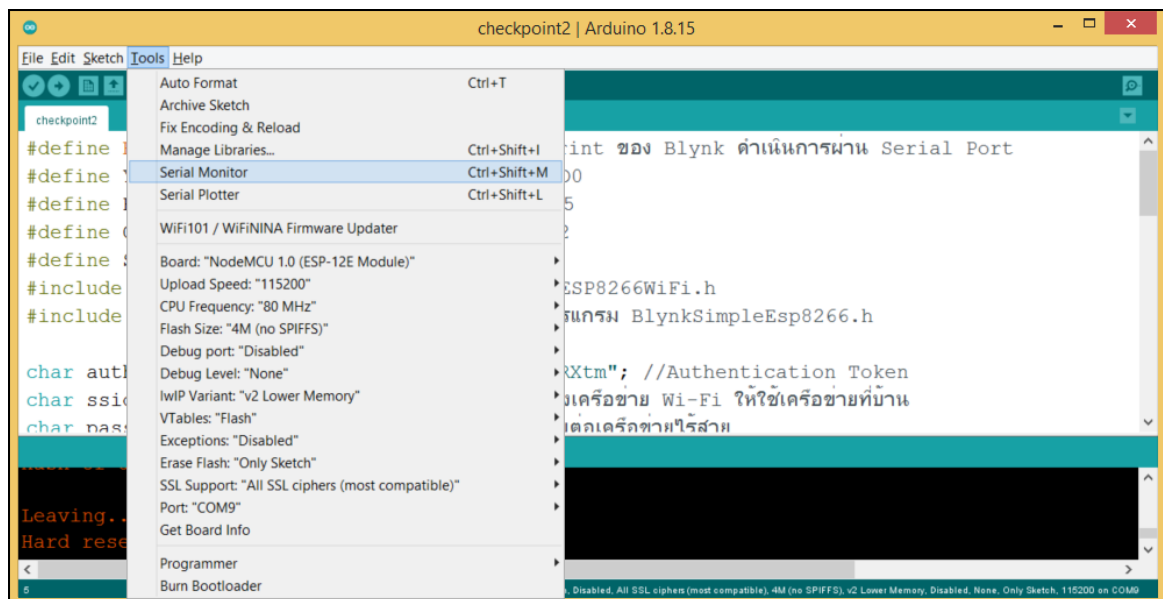
9.25 ที่โปรแกรม Arduino IDE ให้เลือกเมนู Tools>Manage Libraries จะขึ้นไดอะล็อกบ็อกซ์ Library Manager ดังรูปที่ 16 สั่งให้เพิ่มไลบรารี Blynk แล้วกดปุ่ม Install



รูปที่ 16 ไดอะล็อกบ็อกซ์ Library Manager สำหรับเพิ่มโปรแกรมซัพพอร์ต Blynk

9.26 สังเกต Code บรรทัดที่ 22-23 ในหัวข้อ 9.21 ซึ่ง ssid[] นี้เป็นค่า SSID ของเครือข่าย Wi-Fi ที่นักศึกษาใช้ ซึ่งได้ตั้งไว้แล้วให้เป็นเครือข่าย "CoEIoT" แต่ในกรณีที่นักศึกษาทดลองอยู่ ณ ที่พำนักของตนเอง หรือที่อื่น ให้เปลี่ยนตัวแปร pass[] เป็นรหัสผ่านของเครือข่าย Wi-Fi ที่ใช้ กรณีที่นักศึกษาไม่มีเครือข่าย Wi-Fi ที่บ้านแต่ใช้อินเทอร์เน็ตบนโทรศัพท์มือถือ ให้นักศึกษาตั้งค่าโทรศัพท์มือถือของตนเองให้ปล่อยสัญญาณฮอตสปอต Wi-Fi (วิธีการตั้งค่าแต่ละเครื่องแตกต่างกันไป ให้ศึกษาจากอินเทอร์เน็ต) โดยนำค่า SSID และรหัสผ่านที่ตั้งไว้มาใส่ในโค้ดบรรทัดที่ 22 และ 23

9.27 คอมไพล์และอัปโหลดโปรแกรมลงสู่บอร์ด NodeMCU จากนั้นทดลองใช้แอป Blynk ที่สร้างขึ้นบนโทรศัพท์ในการควบคุมแอลอีดีบนบอร์ด และทดลองเลื่อน SW1 และสังเกตการเปลี่ยนแปลงบนโทรศัพท์มือถือ.

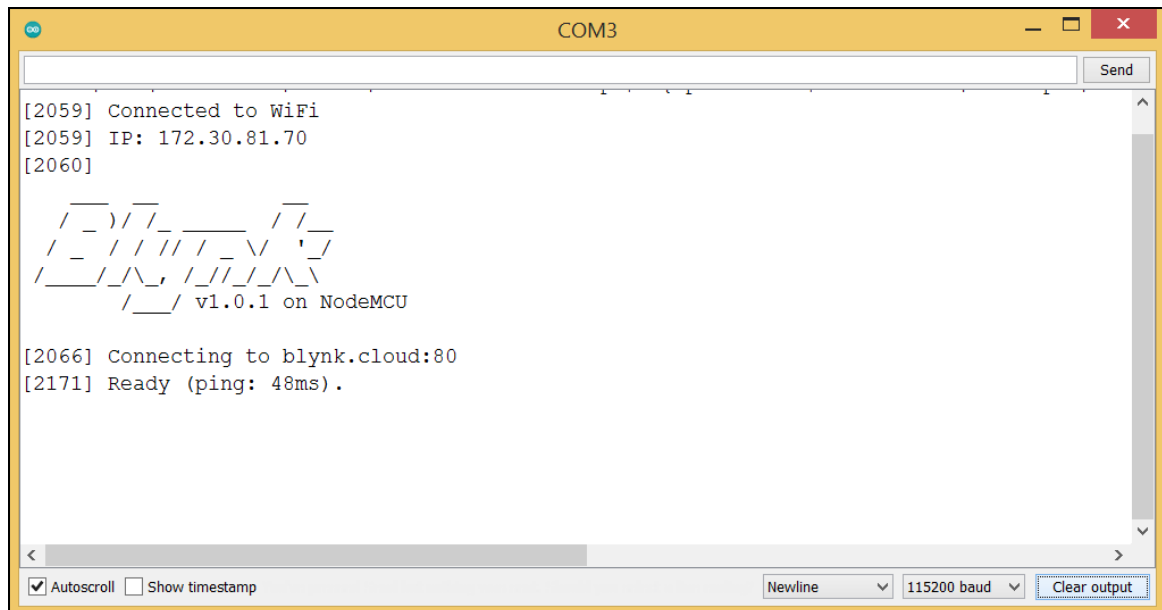


รูปที่ 18 วิธีการเปิดโปรแกรม Serial Monitor ใน Arduino IDE

9.28 เปิด Serial Monitor ใน Arduino IDE จากนั้นกดปุ่ม Reset บนบอร์ด NodeMCU ตรวจสอบการทำงานของบอร์ด NodeMCU หากการเชื่อมต่อระบบเครือข่ายไร้สาย Wi-Fi สำเร็จจะขึ้นสถานะในโปรแกรม Serial Monitor ดังแสดงในรูปที่ 19

9.29 ทดลองกด Button ในโปรแกรม Blink บนหน้าจอโทรศัพท์มือถือและเลื่อน Slider ไปมา และสังเกตการเปลี่ยนแปลงของหลอดแอลอีดีสีแดงและสีเขียวซึ่งเชื่อมต่อกับ NodeMCU

9.30 เลื่อนดิปสวิทช์บิตที่ 1 ไปมา สังเกตการณ์เปลี่ยนแปลงทางตรรกะของค่าที่อ่านได้จาก LED บนหน้าจอมือถือ



รูปที่ 19 สถานะการเชื่อมต่อ Wi-Fi ของ NodeMCU ซึ่งแสดงผลผ่านโปรแกรม Serial Monitor

#### 10. การทดลองสำหรับ Checkpoint3

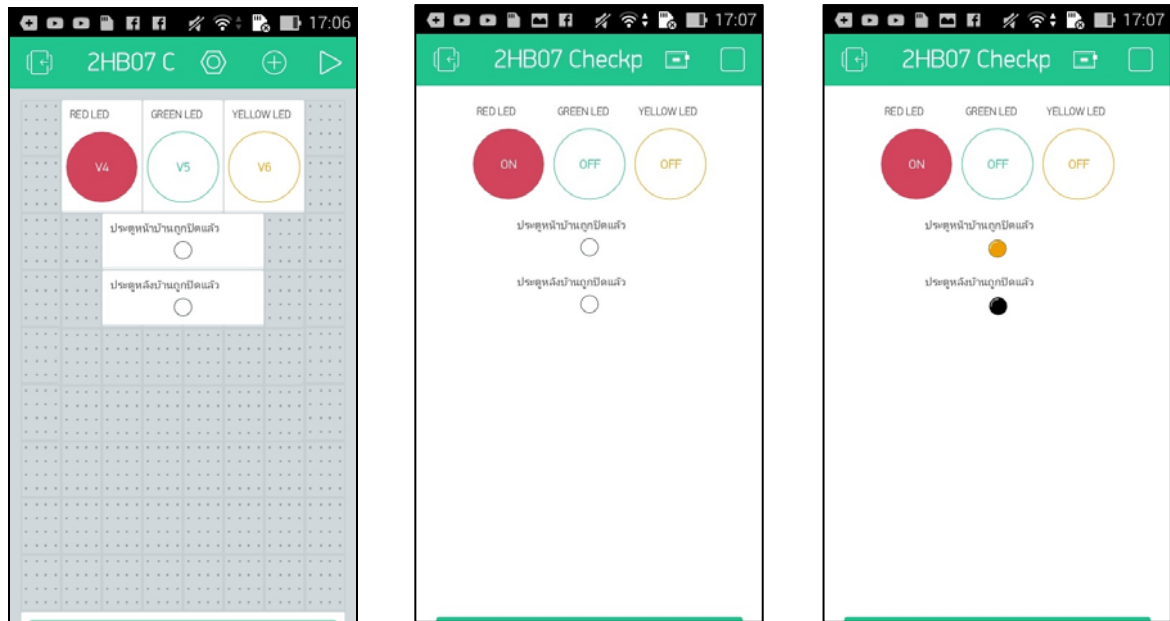
จงเขียนโปรแกรมบน NodeMCU เพื่อให้อ่านค่าจากดิปสวิทช์บิตที่ 5-8 ซึ่งต่ออยู่กับ GPIO12, GPIO13, GPIO5 และ GPIO16 ตามลำดับ) เพื่อควบคุมการติดดับของแอลอีดีทั้งสามดวง โดยมีรูปแบบการควบคุมดังแสดงในตารางที่ 2 หลังจากนั้นให้สร้างโปรเจกต์บน Blynk โดยรับข้อมูลจากดิปสวิทช์สองบิตล่าง ซึ่งถูกอ่านโดย NodeMCU ให้ใช้ LED Widget จำนวน 2 ตัวเพื่อแสดงสถานะทางตรรกะที่อ่านได้จากสวิทช์ กำหนดให้สวิทช์แต่ละตัวจำลองการผ่านค่าจากตัวตรวจรู้ (Sensor) ดังต่อไปนี้

- ดิปสวิทช์บิตที่ 1 จำลองสถานการณ์เปิด/ปิดของประตูหน้าบ้าน
- ดิปสวิทช์บิตที่ 2 จำลองสถานการณ์เปิด/ปิดของประตูหลังบ้าน

ตารางที่ 2 รูปแบบการติดดับของแอลอีดีทั้งสามตัวซึ่งควบคุมโดยการตั้งค่าจากดิปสวิทช์

ค่าที่อ่านได้จากดิปสวิทช์ GPIO (16, 5, 13, 12)	หน้าที่การทำงานของโปรแกรม
1000 <sub>2</sub>	แอลอีดีทุกดวงกระพริบติดดับพร้อมกันทุก ๆ 1 วินาที
1001 <sub>2</sub>	แอลอีดีสีแดงถูกควบคุมการติดดับโดย Button บนโทรศัพท์มือถือ แอลอีดีดวงอื่นดับตลอดเวลา และไม่สามารถถูกควบคุมได้โดยโทรศัพท์
1010 <sub>2</sub>	แอลอีดีสีเขียวถูกควบคุมการติดดับโดย Button บนโทรศัพท์มือถือ แอลอีดีดวงอื่นดับตลอดเวลา และไม่สามารถถูกควบคุมได้โดยโทรศัพท์
1011 <sub>2</sub>	แอลอีดีสีเหลืองถูกควบคุมการติดดับโดย Button บนโทรศัพท์มือถือ แอลอีดีดวงอื่นดับตลอดเวลา และไม่สามารถถูกควบคุมได้โดยโทรศัพท์
0100 <sub>2</sub>	แอลอีดีทั้งสามดวงถูกควบคุมการติดดับโดย Button บนโทรศัพท์มือถือ
0101 <sub>2</sub>	แอลอีดีสีแดงกระพริบติดสลับกับดับทุก ๆ 1 วินาที ด้วยความสว่าง 20 %
0110 <sub>2</sub>	แอลอีดีสีเขียวกระพริบติดสลับกับดับทุก ๆ 1 วินาที ด้วยความสว่าง 50 %
0111 <sub>2</sub>	แอลอีดีสีเหลืองกระพริบติดสลับกับดับทุก ๆ 1 วินาที ด้วยความสว่าง 70 %

บันทึกโปรแกรมลงบอร์ด NodeMCU และทดสอบการเชื่อมต่อระหว่างบอร์ด NodeMCU และโปรแกรม Blynk ที่พัฒนาขึ้นบนโทรศัพท์ และเรียก TA หรือเจ้าหน้าที่ตรวจ Checkpoint



รูปที่ 22 ตัวอย่างหน้าจอของโปรเจกต์บน Blynk ของ Checkpoint 3

#### 11. การทดลองสำหรับ Checkpoint4 (Bonus)

checkpoint4 นี้ใครไม่ทำก็ไม่เป็นไร แต่หากทำส่งก็จะมีคะแนนพิเศษให้ กำหนดให้นักศึกษาเชื่อมต่อบอร์ด ESP8266 เข้ากับจอแอลซีดีและ Serial EEPROM ภายนอก (แนะนำให้ใช้บอร์ด DS-1307 ซึ่งมีไอซี Serial EEPROM เบอร์ 24C32 ในตัว) กำหนดให้เมื่อเปิดเครื่อง ตัวบอร์ด ESP8266 จะนำข้อความที่เก็บใน Serial EEPROM ซึ่งยาวไม่เกิน 8 ตัวอักษรแสดงผลออกแสดงผลทางจอแอลซีดี กำหนดให้ผู้ใช้สามารถอัปเดตหรือเปลี่ยนแปลงข้อความซึ่งเก็บใน Serial EEPROM นี้ได้ผ่านทางแอปพลิเคชันบนโทรศัพท์มือถือ (แนะนำให้ทำผ่าน Blynk App) นักศึกษาสามารถทำการออกแบบการเชื่อมต่อฮาร์ดแวร์ได้เองอย่างอิสระ

ในการส่งงาน ให้ TA เป็นผู้กำหนดข้อความที่จะเขียนลง Serial EEPROM บนโทรศัพท์ และทำการส่งให้ ESP8266 เขียนลงใน Serial EEPROM และทดลองปลดแหล่งจ่ายไฟของ ESP8266 ออกจากนั้นทดสอบดูว่าข้อมูลที่รับเข้ามาใหม่นั้นยังอยู่ใน Serial EEPROM หรือไม่โดยการสังเกตข้อความบนจอแอลซีดี

## ภาคผนวก

### 1. การใช้งานฟังก์ชัน analogWrite

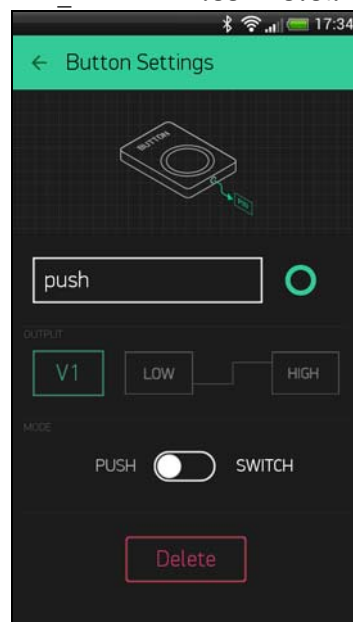
ฟังก์ชัน analogWrite ใช้ในการส่งสัญญาณพัลส์ที่สามารถปรับค่าวัฏจักรหน้าที่ (Duty Cycle) ได้ สัญญาณที่ออกมาเป็นสัญญาณที่เกิดจากการมอดูเลตความกว้างของพัลส์ (Pulse Width Modulation : PWM) รูปแบบการใช้งานของฟังก์ชันนี้ คือ

```
analogWrite(pin, value)
```

เมื่อ pin คือ หมายเลขของขา GPIO ของบอร์ด NodeMCU ที่ใช้งาน และ  
value คือ ค่าสำหรับควบคุมวัฏจักรหน้าที่ของสัญญาณพัลส์ขาออก ค่าที่สามารถใช้ได้กับบอร์ด NodeMCU คือ 0-1023

### 2. การส่งข้อมูลจากโปรแกรมประยุกต์ Blynk ไปยังฮาร์ดแวร์

Widget Controller ต่าง ๆ บน Blynk สามารถส่งข้อมูลไปควบคุมฮาร์ดแวร์ได้ผ่านการตั้งค่าของตัวควบคุมแต่ละตัว อย่างไรก็ตาม หากผู้ใช้ต้องการนำค่า ที่ Blynk ส่งมาไปประมวลผลเพิ่มเติม หรือต้องการนำมาจัดการเอง สามารถทำได้โดยการใช้งานฟังก์ชัน BLYNK\_WRITE ดังตัวอย่างต่อไปนี้



จากรูปด้านบน เป็นการตั้งค่าของ Button ให้ทำงานกับ Virtual PIN หมายเลข 1 (หรือ V1) หากผู้ใช้ต้องการนำค่าที่ V1 ส่งมาไปประมวลผลต่อเพิ่มเติม สามารถทำได้ดังนี้

```
BLYNK_WRITE(V1) //Button Widget is writing to pin V1
{
  int pinData = param.asInt();
  //ใส่โค้ดที่จะนำตัวแปร pinData ไปใช้ตรงนี้
  //โค้ดอื่น ๆ (ถ้ามี)
}
```