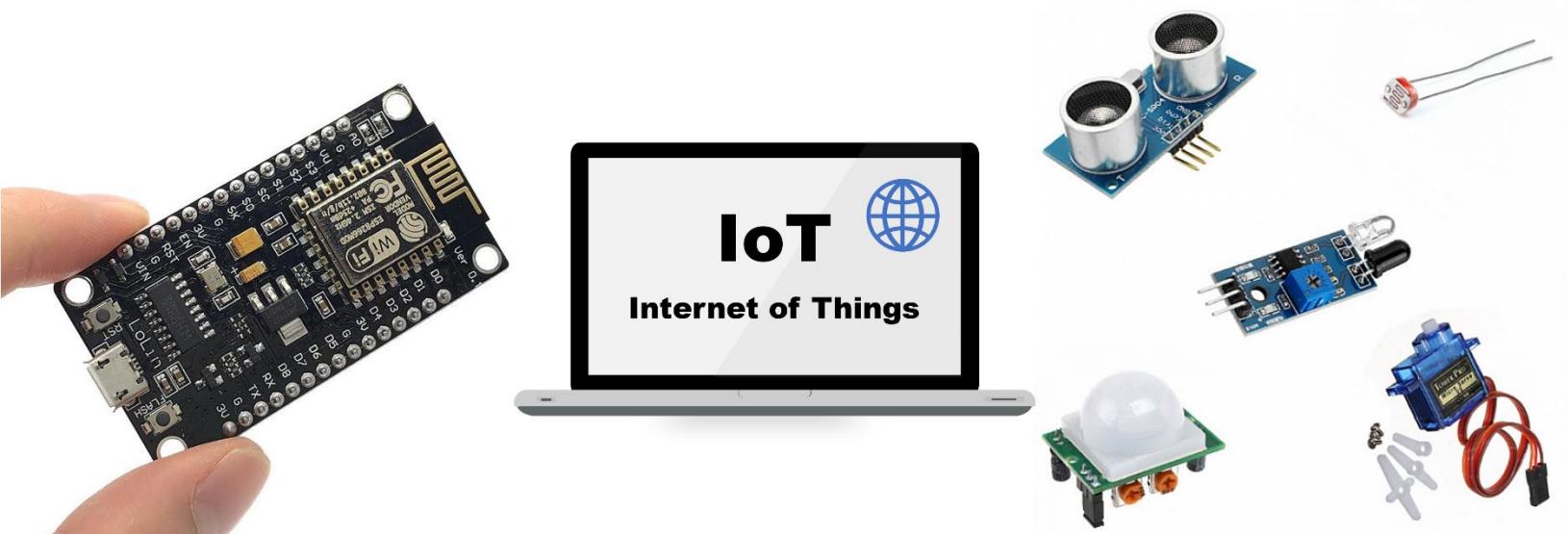


Laboratory Worksheet

ITD62-331 Internet of Things



By Lect.Nannaphat Siribunyaphat. Ph.D.

School of Informatics, Walailak University

Student ID:

Name

คำนำ

เอกสารฉบับนี้ เป็นเอกสารประกอบการเรียนการสอนในวิชา ITD62-332 อินเทอร์เน็ตของสรรพสิ่ง (Internet of Things) โดยมีวัตถุประสงค์เพื่อให้นักศึกษาได้เรียนรู้การพัฒนาระบบอินเทอร์เน็ตของสรรพสิ่งทั้งในด้านอุปกรณ์硬件พื้นฐาน การต่อวงจรอิเล็กทรอนิกส์ การเขียนโปรแกรมสำหรับระบบฝังตัว การเขียนโปรแกรมประยุกต์สำหรับระบบอินเทอร์เน็ตของสรรพสิ่ง และการเชื่อมต่อข้อมูลกับฐานข้อมูล โดยมีเนื้อหาในการเรียนรู้ที่สอดคล้องกับบทเรียนในวิชานี้ เพื่อให้นักศึกษามีความเข้าใจในการพัฒนาระบบอินเทอร์เน็ตของสรรพสิ่ง และสามารถนำไปต่อยอดหรือประยุกต์กับงานต่าง ๆ ได้ในอนาคต

อาจารย์ ดร.นันท์นภัส ศิริบุญญพัฒน์

สารบัญ

คำนำ	ก
สารบัญ	ข
บทที่ 1 บทนำ	1
1. NodeMCU (ESP8266) Microcontroller	1
1.1. ส่วนประกอบของบอร์ด NodeMCU	1
1.2. NodeMCU Specification.....	3
2. Arduino IDE	4
2.1. การติดตั้งโปรแกรม Arduino IDE.....	4
2.2. การติดตั้ง Libraries.....	5
2.3. การติดตั้ง Driver และตรวจสอบ Port.....	8
3. โครงสร้างภาษาซีสำหรับการเขียน NodeMCU ผ่าน Arduino IDE.....	10
4. การทำงานของอุปกรณ์แบบ Active Low และ Active High	10
4.1. Active Low หรือ วงจร Pull-Up.....	10
4.2. Active High หรือ วงจร Pull-Down	11
Lab 1.1: ทดสอบการทำงานของบอร์ด.....	12
บทที่ 2 การใช้งาน Serial Monitor	13
Lab 2.1: Basic Serial Monitor	13
Lab 2.2: การส่งข้อมูลจากแป้นพิมพ์ผ่าน Serial Monitor ไปยังบอร์ด	15
Lab 2.3: การใช้งาน Serial Plotter.....	16
บทที่ 3 พื้นฐานการใช้ขา Input/Output (GPIO)	18

Lab 3.1: วงจรไฟกะพริบโดยใช้ LED (Digital Output).....	18
Lab 3.2: เปิด-ปิดหลอด LED โดยใช้ switch.....	21
บทที่ 4 การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิทัล (Analog to Digital Convertor)	24
Lab 4.1: การวัดสัญญาณอนาล็อก	24
Lab 4.2: กำหนดช่วงของเอาต์พุตด้วยฟังก์ชัน map.....	26
บทที่ 5 การเชื่อมต่อข้อมูลระหว่างอุปกรณ์	30
Lab 5.1: การสื่อสารแบบ UART	31
Lab 5.2: การสื่อสารแบบ SPI	32
5.3 การสื่อสารแบบ I2C	35
บทที่ 6 การขัดการทำงาน (Interrupt)	39
Lab 6.1: Interrupt CHANGE	40
Lab 6.2: Interrupt FALLING.....	42
Lab 6.3: Interrupt RISING.....	43
บทที่ 7 การแสดงผลผ่านโมดูลจอ	44
Lab 7.1: การค้นหา Address ของโมดูลจอ LCD	45
Lab 7.2: การแสดงผลบนจอ LCD.....	47
Lab 7.3: การแสดงผลบนโมดูลจอ OLED.....	50
Lab 7.4: การแสดงภาพกราฟิกบนโมดูลจอ OLED	55
บทที่ 8 การใช้งานเซ็นเซอร์	58
Lab 8.1: การวัดอุณหภูมิและความชื้นในอากาศด้วยโมดูล DHT	58
Lab 8.2.: การเปิด-ปิดหลอดไฟด้วยโมดูล LDR.....	61
Lab 8.3: การวัดระยะทางและแจ้งเตือนด้วยโมดูล Ultrasonic Sensor	63
Lab 8.4: ตรวจจับการเคลื่อนไหวด้วยโมดูล PIR Motion Sensor	65

Lab 8.5: การอ่านค่าด้วย RFID Tag	69
บทที่ 9 การพัฒนาแอปพลิเคชันเพื่อเชื่อมต่อและควบคุมอุปกรณ์ผ่านอินเทอร์เน็ต	72
การเริ่มต้นการใช้งาน Blynk	73
Lab 9.1: ระบบเปิด/ปิดหลอดไฟผ่าน Blynk.....	77
Lab 9.2: การสร้าง Dashboard ในการรายงานอุณหภูมิและความชื้นบนแอปพลิเคชัน Blynk.....	82
บทที่ 10 การเชื่อมต่อระบบ IoT กับ Firebase.....	93
Lab 10.1: การเชื่อมต่อ NodeMCU กับ Firebase.....	93
Lab 10.2: การบันทึกค่าอุณหภูมิและความชื้นลงใน Realtime Database ของ Firebase.....	104
Lab 10.3: การบันทึกสถานะของสวิตช์ลงใน Realtime Database ของ Firebase.....	108
Lab 10.4: การบันทึกสถานะของสวิตช์ลงใน Realtime Database ของ Firebase ร่วมกับการสั่งการบน Blynk	111
Reference	115

บทที่ 1

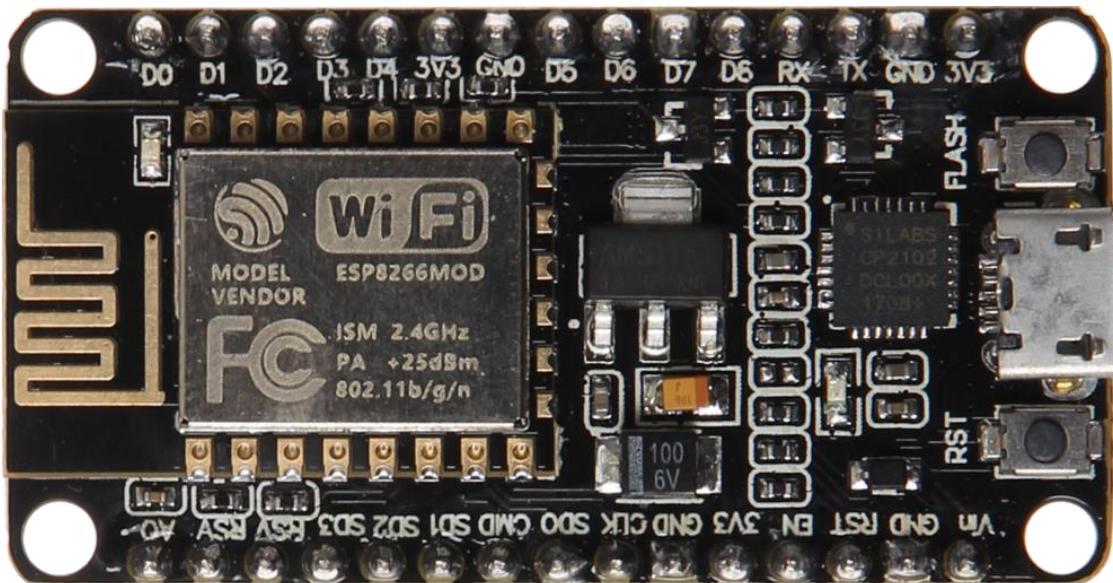
บทนำ

1. NodeMCU (ESP8266) Microcontroller

NodeMCU เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ใช้เฟิร์มแวร์โอเพนซอร์สของ Lua สำหรับชิป ESP32 และ ESP8266 ซึ่งเป็นชิปที่มีระบบการเชื่อมต่อ WiFi ฝังบนบอร์ด (System-on-Chip: SoC) ทำให้สามารถเชื่อมต่อกับเครือข่ายได้ง่าย งานได้หลากหลาย มีขนาดเล็ก และมีราคาถูก สามารถพัฒนาโปรแกรมได้โดยใช้ภาษาซี (C language) ผ่าน Arduino IDE [1] โดยในการเรียนวิชานี้จะใช้ชิป ESP8266 ในการเรียนการสอน

1.1. ส่วนประกอบของบอร์ด NodeMCU

บอร์ด NodeMCU ESP8266 จะมีองค์ประกอบดังที่แสดงในรูปต่อไปนี้ [2]



รูปที่ 1.1 NodeMCU ESP8266

ที่มา: <https://joy-it.net/en/products/SBC-NodeMCU>

- Digital Pin: เป็นขาติดิจิทัล มีทั้งหมด 9 ขา ตั้งแต่ D0 – D8 โดยทำหน้าที่เป็น GPIO คือ สามารถกำหนดให้เป็นขาอินพุตหรือเอาต์พุตก็ได้ นอกจากนี้ยังสามารถทำงานเฉพาะได้สำหรับบางขา คือ
 - D0: Wake pin ใช้สำหรับรับกระแสตุ้นให้ชิปทำงานจากสถานะ deep-sleep

- D1: SCL pin ใช้ในการสื่อสารแบบ I2C ทั้งหมด Master และ Slave สามารถทำงานสูงสุดที่ความถี่ 100 kHz
- D2: SDA pin ใช้ในการสื่อสารแบบ I2C ทั้งหมด Master และ Slave สามารถทำงานสูงสุดที่ความถี่ 100 kHz
- D5-D8: สามารถใช้ในการเชื่อมต่อแบบ SPI และ HSPI ทั้งหมด Master และ Slave รองรับการทำงาน timing ทั้ง 4 โหมดของ SPI format transfer ที่ความถี่สูงสุด 80 MHz และการทำงานแบบ FIFO สูงสุดที่ 64 bytes
- Analog Pin: เป็นขาแอนalog มีเพียง 1 ขา คือ A0 เป็นอินพุตที่สามารถอ่านค่าจากเซ็นเซอร์และทำหน้าที่เป็น Analog to Digital Converter (ADC) แบบ 10-bit เพื่อแปลงสัญญาณ成และนำล็อกจากเซ็นเซอร์มาเป็นสัญญาณดิจิทัลเพื่อใช้ในการประมวลผลในคอนโทรลเลอร์
- Power Pin: บนบอร์ดจะประกอบไปด้วยขา Power ทั้งหมด 4 ขา ดังนี้
 - V_{in}: ใช้ในการรับไฟจากแหล่งพลังงานภายนอกเพื่อจ่ายไฟให้กับ NodeMCU และอุปกรณ์ต่อพ่วงได้โดยตรง โดยสามารถรับได้ไม่เกิน 5 V
 - 3.3V: เป็นเอาต์พุตของตัวควบคุมแรงดันไฟฟ้า onboard และสามารถใช้เพื่อจ่ายไฟให้กับส่วนประกอบภายนอกได้
- GND pin: เป็นขาระหว่างกัน มีทั้งหมด 5 ขา
- RX/TX pin: เป็นขา GPIO และขาที่ใช้ในการรับส่งข้อมูลแบบอนุกรม (Serial Communication) กับอุปกรณ์อื่น
- VU pin: เป็นขาแรงดันไฟ 5 V จากพอร์ต USB
- S2/S3 pin: เป็นขา GPIO สำหรับเชื่อมต่อกับอุปกรณ์ SD card ผ่านอินเทอร์เฟส Secure Digital Input/Output (SDIO)
- S0/S1/SC/SK pin: เป็นขา GPIO สำหรับเชื่อมต่อกับอุปกรณ์ SD card ผ่านอินเทอร์เฟส Secure Digital Input/Output (SDIO) และติดต่อสื่อสารกับโมดูลอื่น ๆ ผ่านอินเทอร์เฟส SPI ที่เร็วกว่า I2C
- EN pin: ขาควบคุมให้โมดูลทำงาน เมื่อมีการจ่ายไฟหรือกำหนดสถานะให้เป็น Active High
- RST pin: ขาเรียกตัวโมดูล เมื่อต้องกราวน์หรือกำหนดสถานะให้เป็น Active Low
- 2.4 GHz Antenna: สายอากาศ มีความถี่ 2.4 GHz

- หลอด LED: ใช้ในการแสดงสถานะการอัปโหลดโปรแกรมลงบนบอร์ด รวมถึงแสดงสถานะการทำงานของโปรแกรมหรือการส่งข้อมูล
- Chip ESP8266: หน่วยประมวลผลหลัก 32-bit มีความถี่ 80-160 MHz หน่วยความจำ 128 KB internal RAM + 4 MB external Flash ตัวรับส่งสัญญาณ WiFi มาตรฐาน 802.11 b/g/n ทำงานที่แรงดันไฟฟ้า 3.0-3.6 V และกระแสไฟฟ้าโดยเฉลี่ยที่ 80 mA
- Chip 3.3V LDO Voltage Regulator: ชิปที่ใช้ในการควบคุมแรงดันไฟฟ้าให้คงที่ที่ 3.3 V
- Chip USB to TTL Convertor: เป็นตัวกลางในการสื่อสารระหว่างอุปกรณ์ที่ใช้พอร์ต USB ไปยังบอร์ดไมโครคอนโทรลเลอร์ เช่น การอัปโหลดโปรแกรม
- Flash button: ใช้สำหรับการอัปโหลดโปรแกรม
- Micro USB Connector: ใช้เสียบสาย USB เพื่อจ่ายไฟ 5 V ให้กับบอร์ด และอัปโหลดข้อมูล
- RST button: ใช้ในการรีเซ็ตบอร์ดเพื่อเริ่มการทำงานใหม่

1.2. NodeMCU Specification

ตารางที่ 1.1: ข้อมูลจำเพาะของบอร์ด NodeMCU ESP8266

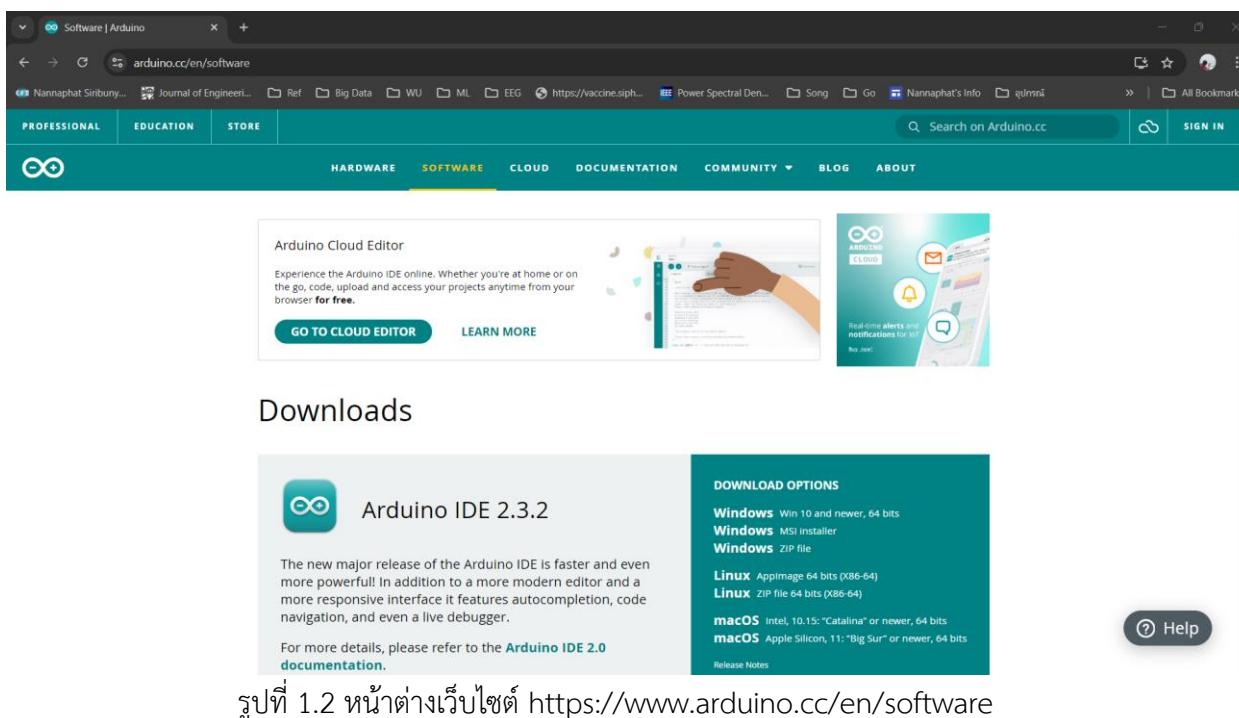
Topic	Specification
Microcontroller	ESP-8266 32-bit
Pin Spacing	0.9" (22.86mm)
Clock Speed	80 MHz
USB Connector	Micro USB
Operating Voltage	3.3V
Input Voltage	4.5V-10V
Flash Memory/SRAM	4 MB / 64 KB
Digital I/O Pins	11
Analog In Pins	1
ADC Range	0-3.3V
UART/SPI/I2C	1 / 1 / 1
WiFi Built-In	802.11 b/g/n

2. Arduino IDE

ในการพัฒนาโปรแกรมสำหรับ NodeMCU ESP8266 จะใช้ภาษาซีในการพัฒนา โดยทำการเขียนโปรแกรมผ่าน Arduino IDE และทำการฝังโปรแกรมลงบนบอร์ด โดยจะต้องดำเนินการเตรียมโปรแกรมให้พร้อมก่อนที่จะทำการเขียนโปรแกรมลงบนบอร์ด โดยมีขั้นตอนดังนี้ [3]

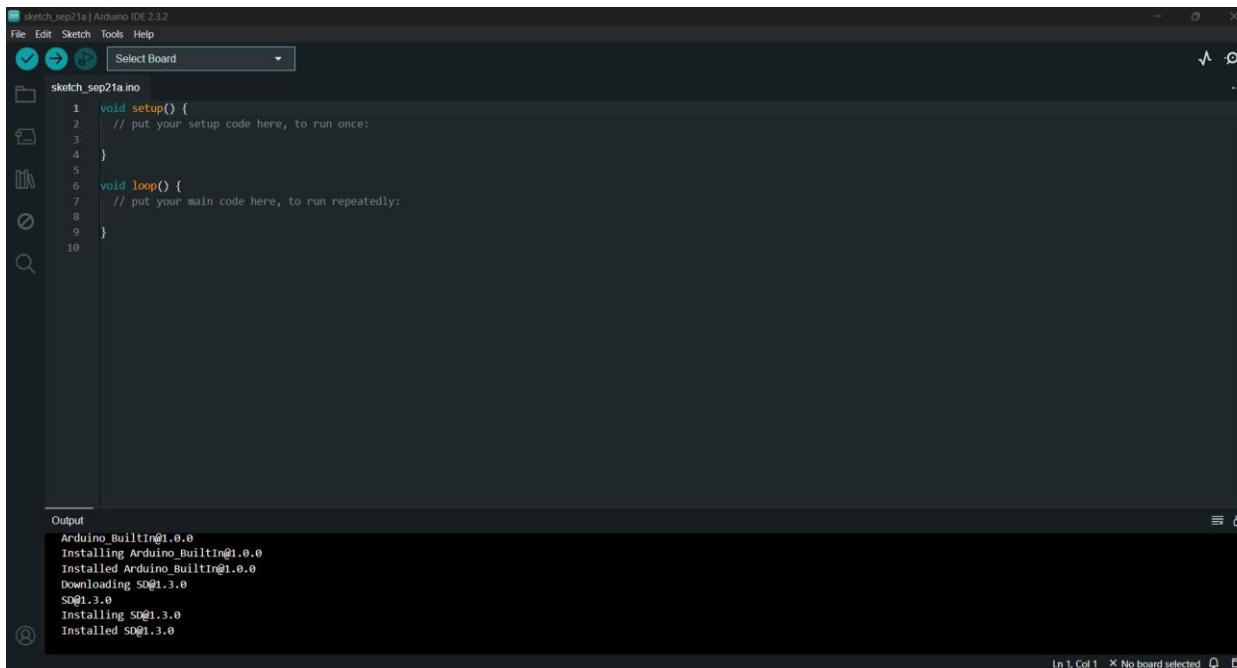
2.1. การติดตั้งโปรแกรม Arduino IDE

- ไปที่เว็บไซต์ <https://www.arduino.cc/en/software> จากนั้นทำการดาวน์โหลดตัวติดตั้งโปรแกรม Arduino IDE ให้ตรงกับระบบปฏิบัติการของคอมพิวเตอร์ที่ใช้



รูปที่ 1.2 หน้าต่างเว็บไซต์ <https://www.arduino.cc/en/software>

- คลิกปุ่ม Just Download จากนั้นเลือกตำแหน่งจัดเก็บไฟล์แล้วกด Save
- ติดตั้งโปรแกรมตามขั้นตอนให้เสร็จ
- เมื่อเปิดโปรแกรม จะแสดงหน้าจอดังรูป



รูปที่ 1.3 หน้าต่างโปรแกรม Arduino IDE

2.2. การติดตั้ง Libraries

- เข้าเว็บไซต์ <https://github.com/esp8266/arduino> จากนั้นเลื่อนลงมาที่ Board Manager Link จากนั้นคัดลอก URL ดังที่แสดงในรูป

<https://github.com/esp8266/arduino>

Installing with Boards Manager

Starting with 1.6.4, Arduino allows installation of third-party platform packages using Boards Manager. We have packages available for Windows, Mac OS, and Linux (32 and 64 bit).

- Download and install Arduino IDE 1.x or 2.x
- Start Arduino and open the Preferences window
- Enter https://arduino.esp8266.com/stable/package_esp8266com_index.json into the File>Preferences>Additional Boards Manager URLs field of the Arduino IDE. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and install esp8266 platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

Latest release [release v1.1.2](#)

Boards manager link: https://arduino.esp8266.com/stable/package_esp8266com_index.json

Documentation: <https://arduino-esp8266.readthedocs.io/en/3.1.2/>

Using git version

Also known as latest git or master branch.

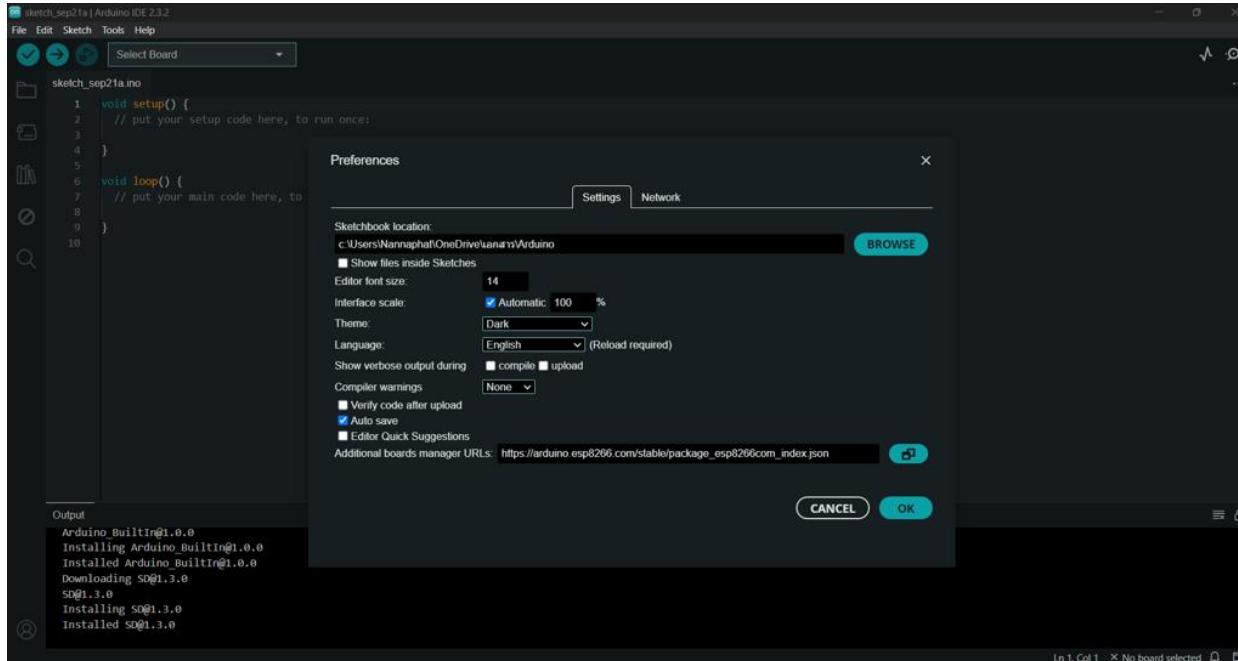
- When using Arduino IDE, follow our instructions here.
- When using PlatformIO, refer to [platformio/espressif8266 platform documentation](#).

Using PlatformIO

PlatformIO is an open source ecosystem for IoT development with a cross-platform build system, a library manager,

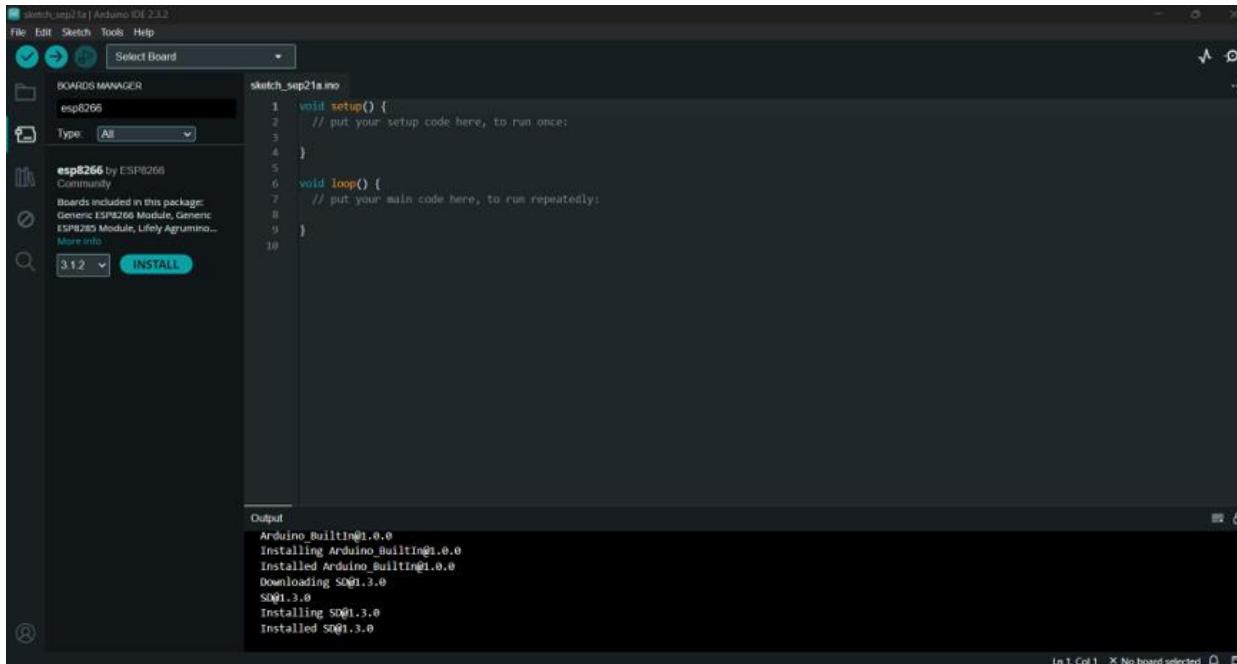
รูปที่ 1.4 ลิงก์สำหรับติดตั้งในการจัดการบอร์ด ESP8266

- เปิดโปรแกรม Arduino IDE คลิกเมนู File -> Preferences.. ที่แท็บ Settings ให้นำ URL ที่คัดลอกไว้มาวางในช่อง Additional Boards Manager URLs จากนั้นคลิกปุ่ม OK



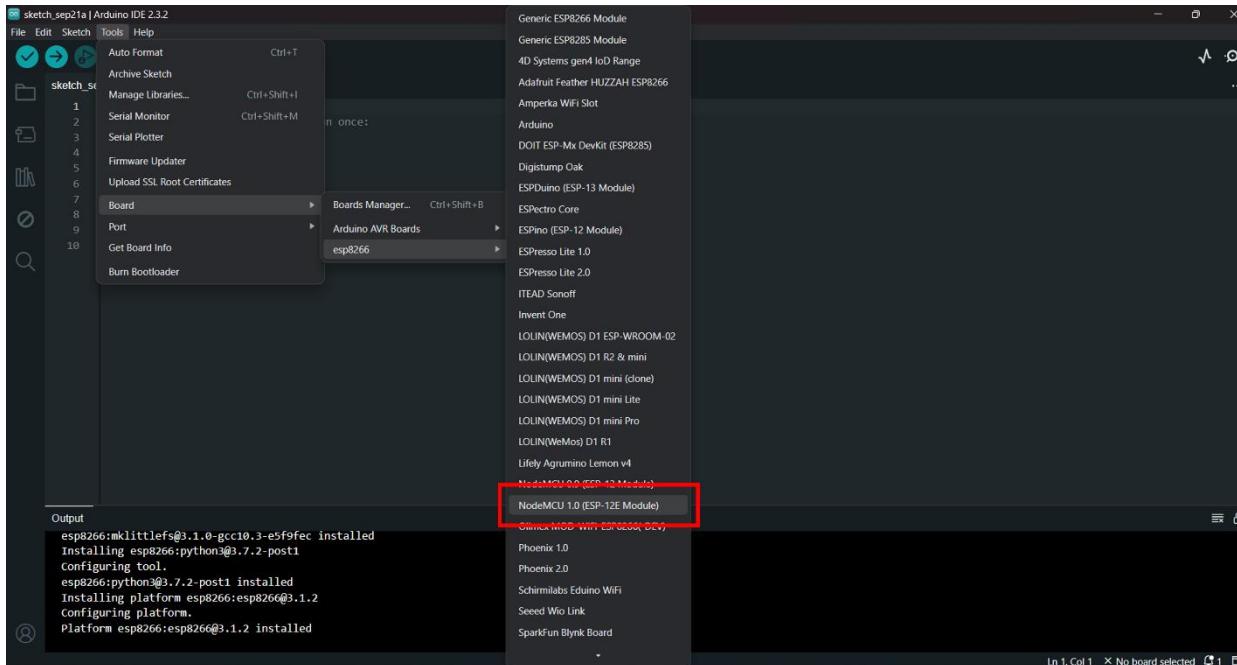
รูปที่ 1.5 Additional Boards Manager URLs

- คลิกที่เมนู Tools -> Board -> Board Manager...
- ในช่องคนหาพิมพ์คำว่า esp8266 จะปรากฏ Libraries ที่รองรับบอร์ด กดแม่ Install เพื่อดาวน์โหลดและติดตั้ง



รูปที่ 1.6 ติดตั้ง Library สำหรับ esp8266

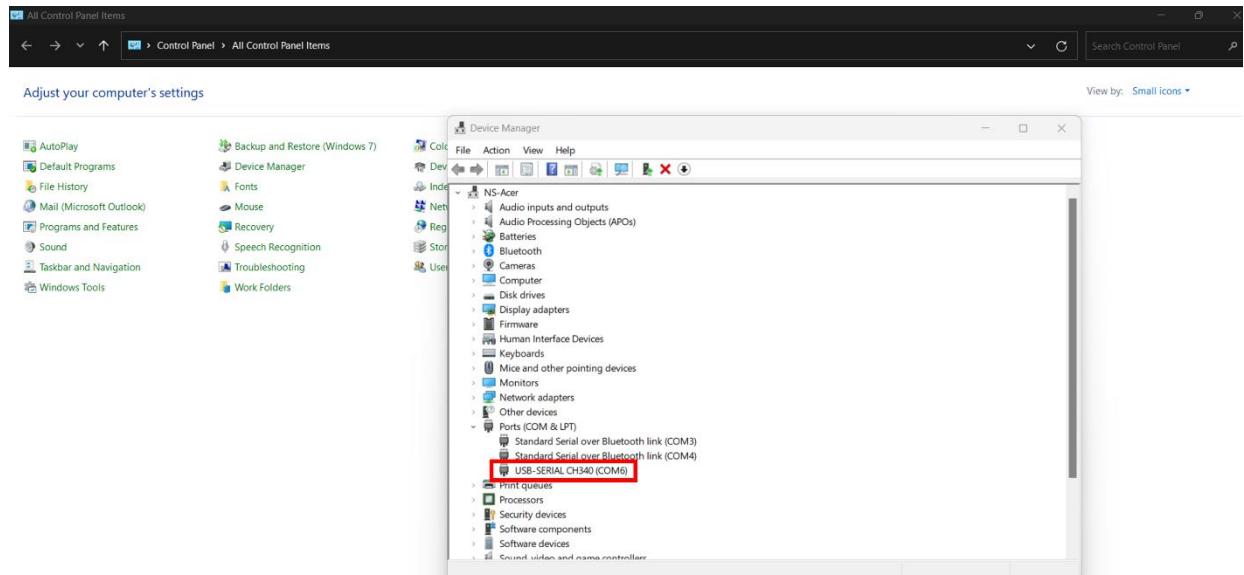
- คลิกเมนู Tools -> Board -> esp8266 เลือก NodeMCU 1.0 (ESP-12E Module)



รูปที่ 1.7 การเลือกบอร์ดสำหรับการเขียนโปรแกรม

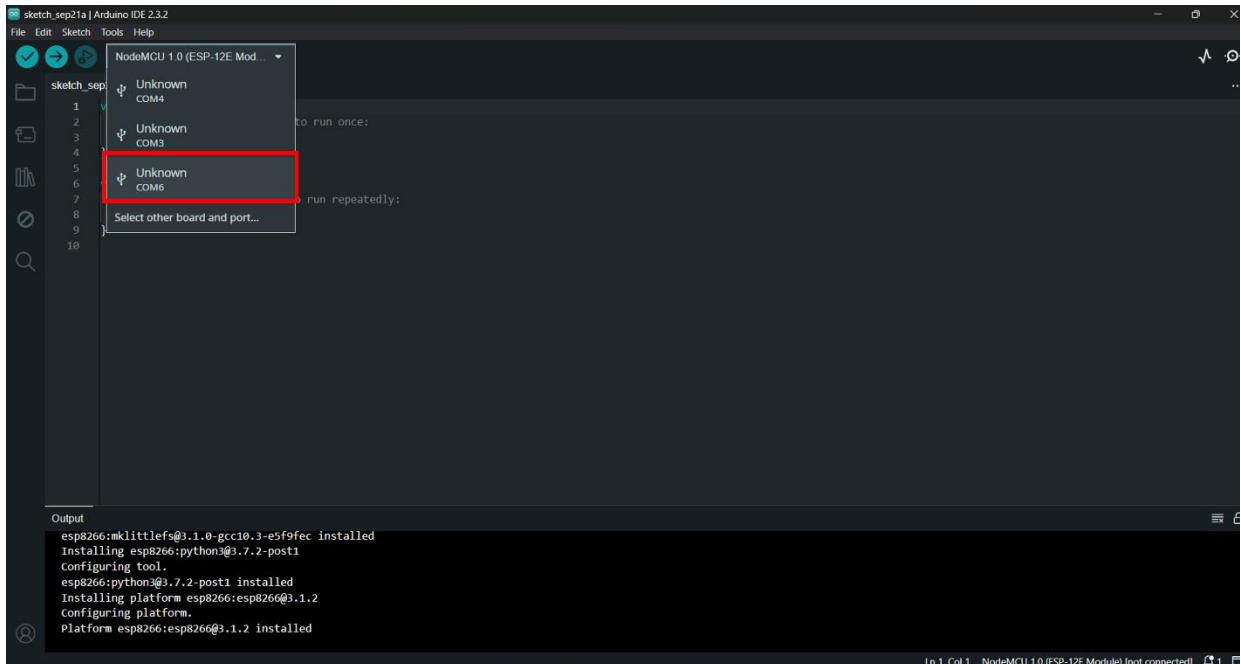
2.3. การติดตั้ง Driver และตรวจสอบ Port

- ต่อบอร์ด NodeMCU กับคอมพิวเตอร์
- ไปที่ Control Panel -> Device Manager จากนั้นไปที่ Ports (COM & PLT) ทำการตรวจสอบพอร์ตที่เชื่อมต่อบอร์ด



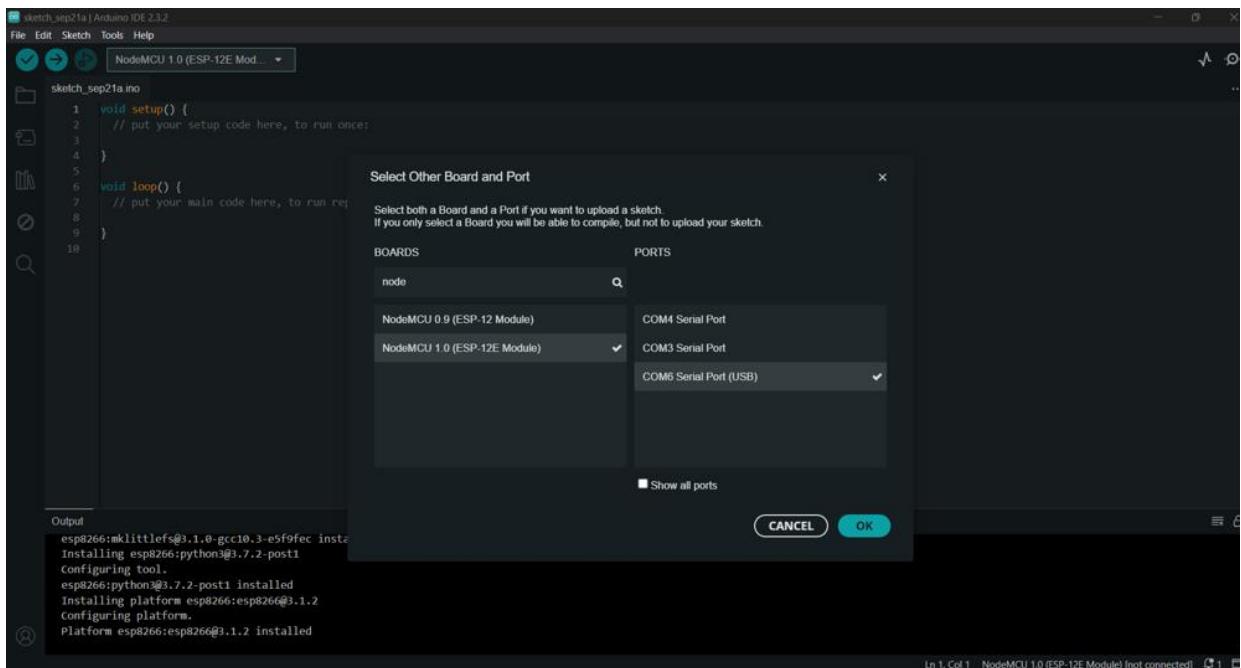
รูปที่ 1.8 พор์ตที่เชื่อมต่อกับบอร์ด NodeMCU

- กลับมาที่ Arduino IDE ที่ແບນໜຸ້ງ ຄົກທີ dropdown ທີ່ມີຊື່ NodeMCU ຈາກນັ້ນກັດເລືອກພອر์ຕີໃຫຍ່ ທີ່ ຕຽບກັບໃນ Device Manager



รูปที่ 1.9 เลือกพอร์ตที่ตรงกับที่คอมพิวเตอร์กำหนด

- ตรวจสอบบอร์ดและพอร์ตให้ถูกต้อง จากนั้นคลิก OK



รูปที่ 1.10 ตรวจสอบบอร์ดและพอร์ตให้ถูกต้อง

3. โครงสร้างภาษาซีสำหรับการเขียน NodeMCU ผ่าน Arduino IDE

การเขียนโปรแกรมภาษาซีสำหรับการใช้งานบนบอร์ด NodeMCU จะมีองค์ประกอบที่สำคัญ 3 ส่วน คือ

1. Header

Header เป็นส่วนที่รวมรวมคำสั่ง พังก์ชัน หรือตัวแปรจาก Libraries ซึ่งสามารถ import เข้ามา เพื่อช่วยในการเขียนโปรแกรมได้ โดยปกติแล้วจะมีหรือไม่มีก็ได้ ขึ้นอยู่กับการเรียกใช้งานในการเขียน โปรแกรมนั้น ๆ

2. void setup()

เป็นพังก์ชันบังคับที่ต้องมีในทุก ๆ โปรแกรม โดยเป็นพังก์ชันที่จะทำการตั้งค่าต่าง ๆ เพื่อให้ โปรแกรมทำงานเพียงรอบเดียวในช่วงที่เริ่มต้นการทำงาน เช่น การตั้งค่าอุปกรณ์เชื่อมต่อ การตั้งค่า Baud Rate เป็นต้น

3. void loop()

เป็นพังก์ชันบังคับที่ต้องมีในทุก ๆ โปรแกรม โดยจะเป็นส่วนที่ใช้ในการทำงานของบอร์ด โดยจะมี การทำงานต่อเนื่องไปเรื่อย ๆ จนกระทั่งมีคำสั่งในอกจาก loop หรือจนกว่าบอร์ดจะหยุดการทำงาน

4. การทำงานของอุปกรณ์แบบ Active Low และ Active High

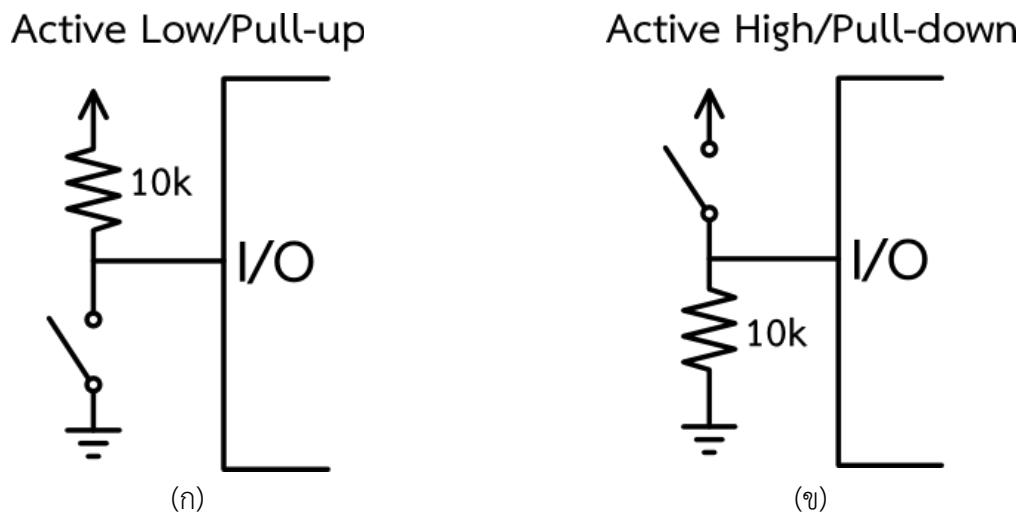
Active Low และ Active High เป็นแนวคิดพื้นฐานสำคัญสำหรับการเขียนโปรแกรมไมโครคอนโทรลเลอร์ โดยจะใช้ในการอธิบายสถานะของสัญญาณว่าเมื่ออูปกรณ์ในสถานะใดจึงถือว่ามีการทำงาน (active) โดยจะมี 2 สถานะ คือ Active Low และ Active High

4.1. Active Low หรือ วงจร Pull-Up

Active Low หรือ วงจร Pull-Up คือ การที่อุปกรณ์จะทำงาน (active) ในกรณีที่สัญญาณมีค่าต่ำ (0) หรือต่อลด GND ตัวอย่างเช่น สวิตช์แบบ Active Low จะมีการเริ่มต้นการต่อวงจรโดยนำตัว ต้านทานมาต่อระหว่างขาอุปกรณ์ I/O และ V_{cc} (การต่อแบบ Pull-Up) จากนั้นต่อสวิตช์ระหว่างขา I/O กับ GND ในกรณีที่ไม่กดสวิตช์ ไฟเลี้ยงจาก V_{cc} จะไปเลี้ยงที่ขา I/O ตลอดเวลา หรือมีค่าสัญญาณเป็น 1 หรือ HIGH และเมื่อทำการกดสวิตช์ ขาของ I/O จะถูกต่อลง GND จึงทำให้สัญญาณที่ได้รับมีค่าเป็น 0 หรือ LOW

4.2. Active High หรือ วงจร Pull-Down

Active High หรือ วงจร Pull-Down จะทำงานในด้านตรงกันข้ามกับวงจร Pull-Up คือ การที่อุปกรณ์จะทำงาน (active) ในกรณีที่สัญญาณมีค่าสูง (1) หรือต่อกายสัญญาณเข้าที่ V_{cc} ตัวอย่างเช่น สวิตช์แบบ Active High จะมีการเริ่มต้นการต่อวงจรโดยนำตัวต้านทานมาต่อระหว่างขาอุปกรณ์ I/O และขา GND (การต่อแบบ Pull-Down) จากนั้นต่อสวิตช์ระหว่างขา I/O กับ V_{cc} ในกรณีที่ไม่กดสวิตช์ ขา I/O จะต่อลง GND ตลอดเวลา หรือมีค่าสัญญาณเป็น 0 หรือ LOW และเมื่อทำการกดสวิตช์ ขาของ I/O จะถูกต่อเข้ากับ V_{cc} จึงทำให้สัญญาณที่ได้รับมีค่าเป็น 1 หรือ HIGH



รูปที่ 1.11 การต่อวงจรในการทำงานบนบอร์ดไมโครคอนโทรลเลอร์ (ก) การต่อวงจรแบบ Active Low (ข) การต่อวงจรแบบ Active High ที่มา <http://www.atrobt.com/2015/10/active-low.html>

โดยทั่วไปแล้ว วงจรอิเล็กทรอนิกส์แบบดิจิทัลจะมีการทำงานแบบ Active Low เนื่องจาก เมื่อระบบมีการทำงาน (active) สัญญาณที่ได้รับจะมีความเสถียรกว่า เพราะว่าการกดสวิตช์ของวงจรแบบ kd จะทำให้สัญญาณต่อลง GND ซึ่งได้ค่า 0 ที่แน่นอน แต่ในกรณีที่ต้องการต่อวงจรแบบ Active High เมื่อกดสวิตช์ วงจรจะต่อสัญญาณกับ V_{cc} ซึ่ง V_{cc} จะมีค่าแปรผันตามแหล่งจ่ายไฟ ทำให้สัญญาณอาจเกิดข้อผิดพลาดได้

Lab 1.1: ทดสอบการทำงานของบอร์ด

- พิมพ์โค้ดต่อไปนี้

```
void setup() {  
    pinMode(D4, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(D4, LOW);  
    delay(1000);  
    digitalWrite(D4, HIGH);  
    delay(2000);  
}
```

- กดปุ่ม Verify เพื่อตรวจสอบโค้ด รอจนกระทั่งตรวจสอบเสร็จ
- กดปุ่ม Upload เพื่ออัปโหลดโค้ดไปยังบอร์ด รอจนกระทั่งอัปโหลดโค้ดเสร็จ

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

บทที่ 2

การใช้งาน Serial Monitor

Serial Monitor เป็นเครื่องมือที่สำคัญในการสร้างโปรเจกต์ด้วย Arduino สามารถใช้เป็นเครื่องมือในการดีบัก ทดสอบแนวคิด หรือทำการสื่อสารกับบอร์ดโดยตรงได้ โดยที่การรับส่งข้อมูลระหว่างบอร์ดกับคอมพิวเตอร์จะเป็นการส่งข้อมูลแบบอนุกรม (Serial Communication) โดยจะต้องทำการตั้งค่า Baud Rate ให้ตรงกัน โดยค่าความเร็วมาตรฐานที่ใช้คือ 9600 baud และค่าสูงสุดที่ใช้ได้ 115200 baud เนื่องจากถ้าค่าสูงกว่านี้จะส่งผลต่อความผิดพลาดได้ร้าย โดยทำการกำหนดค่า Baud Rate ได้จากคำสั่ง Serial.begin() ในโค้ด และตั้งค่า Baud Rate ในหน้าต่าง Serial Monitor ให้ตรงกัน

Lab 2.1: Basic Serial Monitor

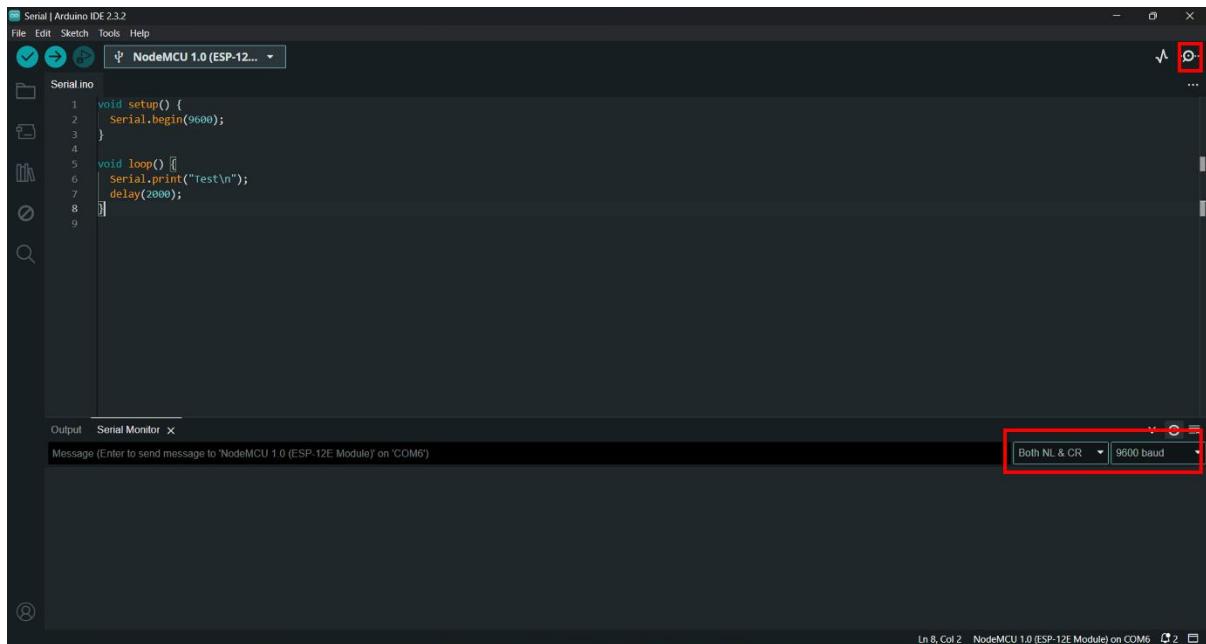
ขั้นตอนในการทำ

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.print("Test\n");  
    delay(2000);  
}
```

- เปิดหน้า Serial Monitor บริเวณมุมขวาของແບບເຄືອງມືອ ແລະ ທ່ານກຳນົດຕັ້ງຄ່າ Baud Rate ໃຫ້ເທົ່າກັບ 9600 ແລະ ປັບໂທມດໃຫ້ເປັນ Both NL & CR

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด



รูปที่ 2.1 การเปิดหน้า Serial Monitor และตั้งค่า Baud Rate

Lab 2.2: การส่งข้อมูลจากแป้นพิมพ์ผ่าน Serial Monitor ไปยังบอร์ด

ขั้นตอนในการทำ

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
const int ledPin = D4;

void setup() {
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    if(Serial.available() > 0){
        char inRead = Serial.read();
        if(inRead == '1') {
            digitalWrite(ledPin, LOW);
            Serial.println("put 1");
        } else if(inRead == '0') {
            digitalWrite(ledPin, HIGH);
            Serial.println("put 0");
        } else if (inRead > '1') {
            Serial.println("Please enter 0 or 1 only");
        }
        delay(100);
    }
}
```

- เปิดหน้า Serial Monitor บริเวณนูมขวาของแทบเครื่องมือ และทำการปรับค่า Baud Rate ให้เท่ากับ 9600 และปรับโหมดให้เป็น Both NL & CR
- ทดลองพิมพ์เลข 0 1 และ 2 ลงในช่อง Message (Enter to send message to NodeMCU) แล้วสังเกตผลลัพธ์ที่เกิดขึ้น

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

Lab 2.3: การใช้งาน Serial Plotter

ขั้นตอนในการทำ

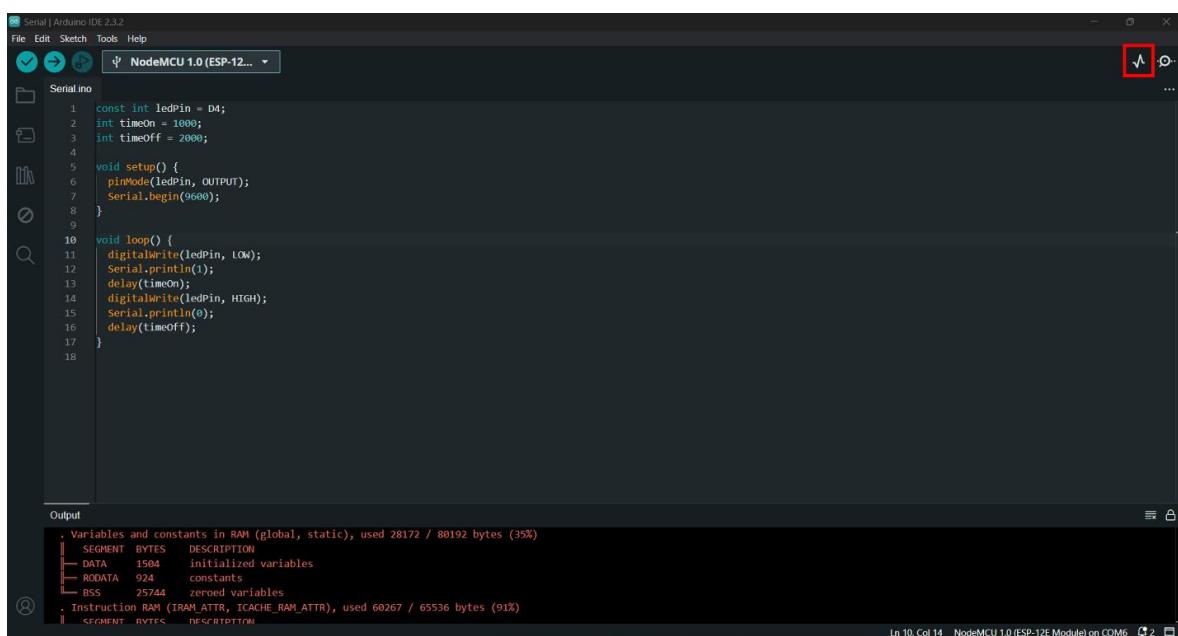
- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
const int ledPin = D4;
int timeOn = 1000;
int timeOff = 2000;

void setup() {
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    digitalWrite(ledPin, LOW);
    Serial.println(1);
    delay(timeOn);
    digitalWrite(ledPin, HIGH);
    Serial.println(0);
    delay(timeOff);
}
```

- เปิดหน้า Serial Plotter กดเลือก value 1 สังเกตผลที่มีการเปลี่ยนแปลง



รูปที่ 2.2 การเข้าหน้า Serial Plotter

- เปิด interpolate ให้เป็น on สังเกตผลที่มีการเปลี่ยนแปลง

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

บทที่ 3

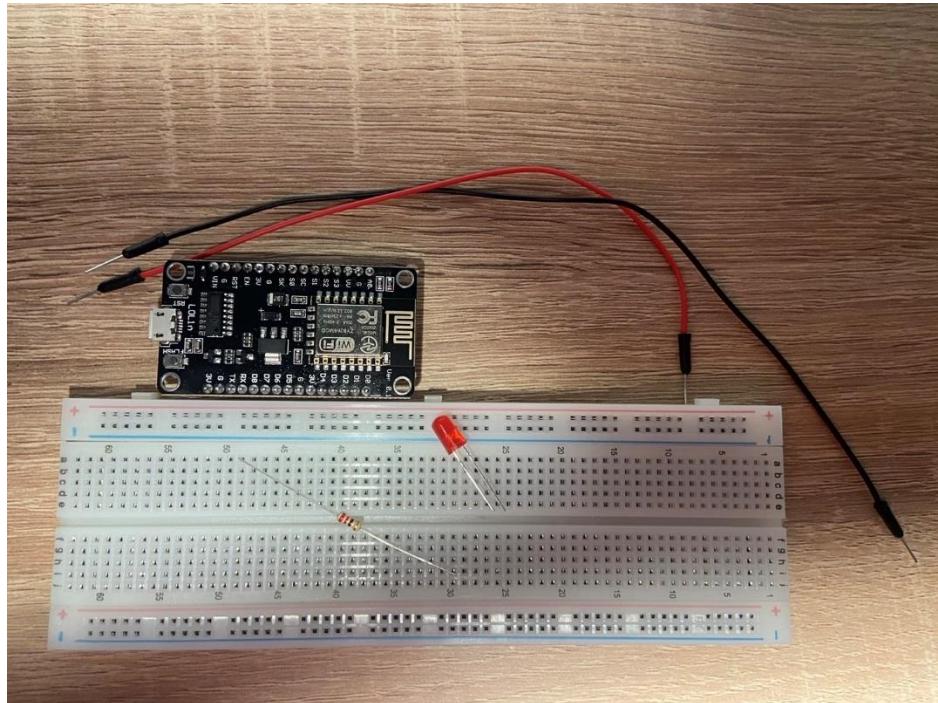
พื้นฐานการใช้ขา Input/Output (GPIO)

General Purpose Input Output (GPIO) เป็นขาของบอร์ด NodeMCU ที่ใช้ในการเชื่อมต่อกับอุปกรณ์ อินพุตและเอาต์พุต โดยสามารถเขียนโค้ดเพื่อออคำสั่งให้แต่ละขาที่ต้องการ เพื่อให้สามารถทำงานได้ตามที่ต้องการ

Lab 3.1: วงจรไฟกระพริบโดยใช้ LED (Digital Output)

อุปกรณ์ที่ต้องใช้

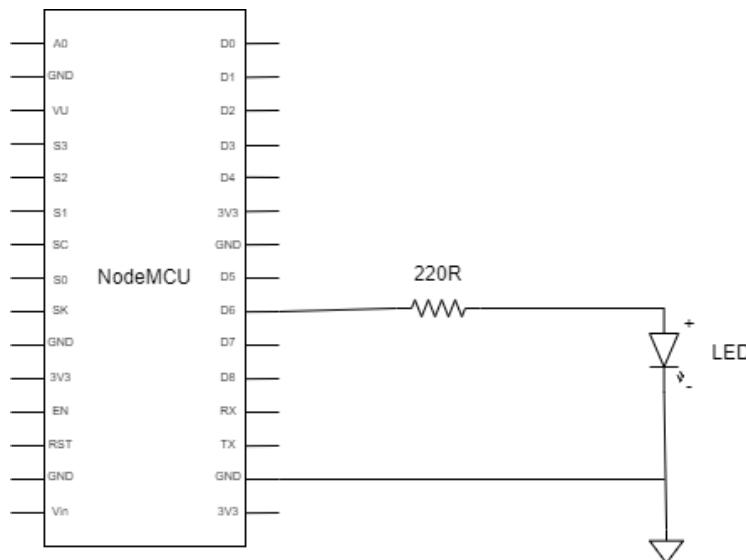
- บอร์ด NodeMCU 1 ตัว
- หลอด LED 1 อัน
- ตัวต้านทานขนาด $220\ \Omega$ 1 อัน
- สาย Jumper
- Breadboard 1 อัน



รูปที่ 3.1 อุปกรณ์ที่ต้องใช้สำหรับแลบ 3.1

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 3.2 วงจรของแลบที่ 3.1

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```

void setup() {
    pinMode(D6, OUTPUT);
}

void loop() {
    digitalWrite(D6, HIGH);
    delay(1000);
    digitalWrite(D6, LOW);
    delay(1000);
}

```

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

- จักราชรหัสมอร์สต่อไปนี้ งบรับปรงโค้ดเพื่อให้วงจรสามารถส่งสัญญาณเป็นคำว่า SOS ได้

ตารางที่ 3.1 ตารางรหัสมอร์ส

ตัวอักษร	รหัส	ตัวอักษร	รหัส	ตัวอักษร	รหัส
A	.-	J	.---	S	...
B	-...	K	-.-	T	-
C	-..	L	.-..	U	..-
D	-..	M	--	V	...-
E	.	N	-.	W	.--
F	.. --.	O	---	X	-..-
G	--.	P	.--.	Y	-.--
H	Q	--.-	Z	--..
I	..	R	.-.		

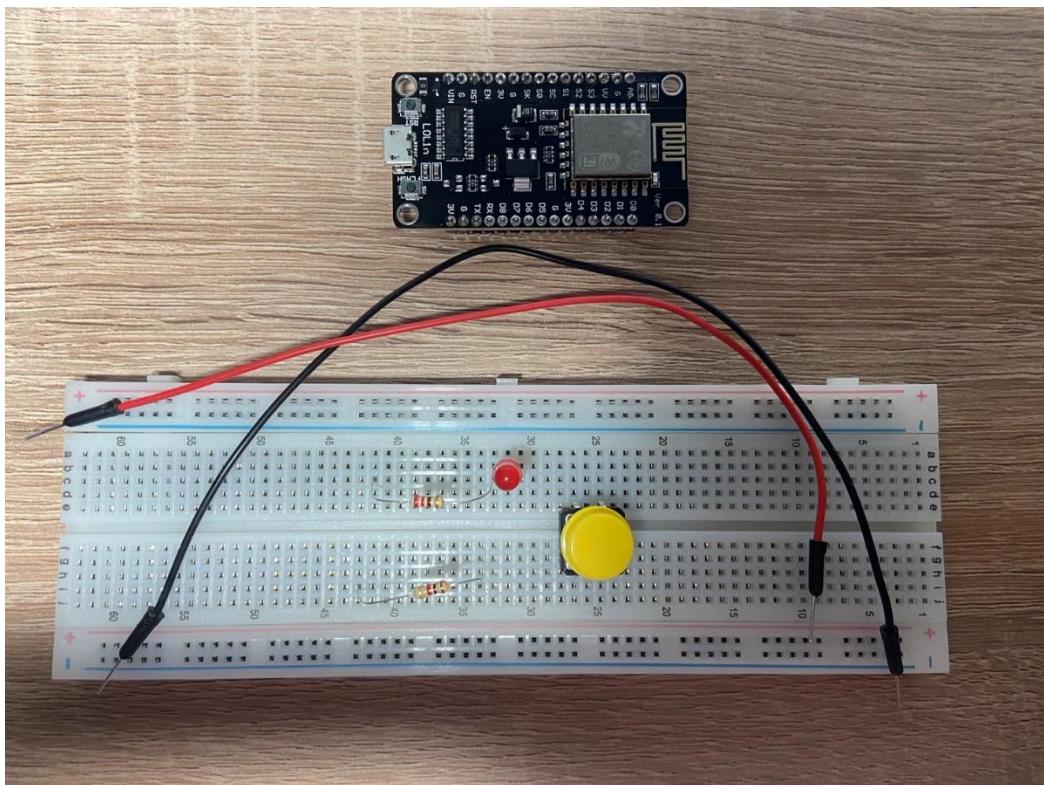
ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

ส่งไฟล์โค้ดที่ทำการแก้ไขแล้วใน Google Classroom ในหัวข้อ Morse Code

Lab 3.2: เปิด-ปิดหลอด LED โดยใช้ switch

อุปกรณ์ที่ต้องใช้

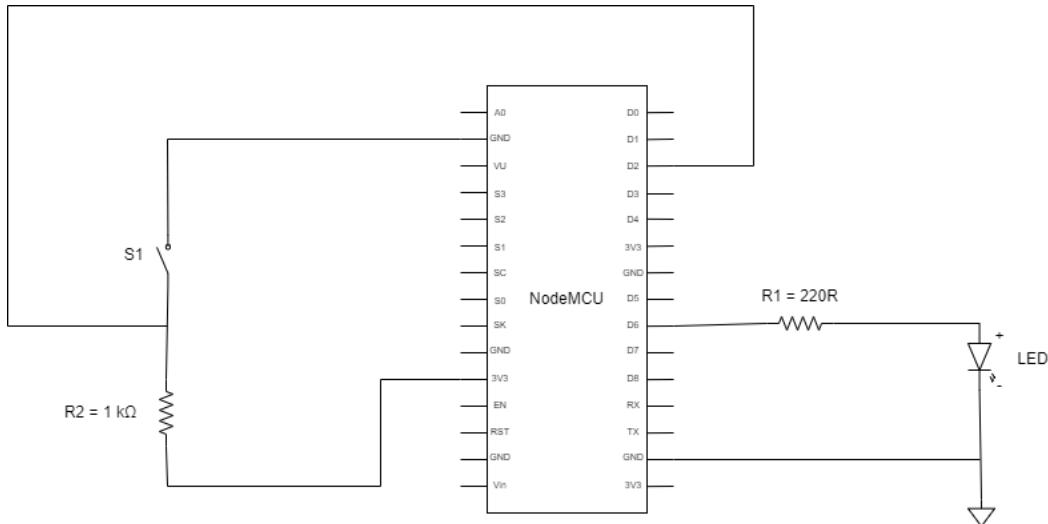
- บอร์ด NodeMCU 1 ตัว
- หลอด LED 1 อัน
- ตัวต้านทานขนาด $220\ \Omega$ 1 อัน
- ตัวต้านทานขนาด $1\ k\Omega$ 1 อัน
- สาย Jumper
- Breadboard 1 อัน
- Tact Switch (สวิตซ์กดติดปล่อยดับ) 1 อัน



รูปที่ 3.3 อุปกรณ์ที่ต้องใช้สำหรับแลบ 3.2

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 3.4 วงจรของแลบที่ 3.2

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```

void setup() {
    pinMode(D2, INPUT);
    pinMode(D6, OUTPUT);
}

void loop() {
    int buttonStatus = digitalRead(D2);
    if (buttonStatus == 0) {
        digitalWrite(D6, HIGH);
    } else {
        digitalWrite(D6, LOW);
    }
    delay(100);
}

```

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

วงจรที่ต่อเป็นวงจรแบบใด เพราะอะไร

บทที่ 4

การแปลงสัญญาณแอนalog เป็นสัญญาณดิจิทัล (Analog to Digital Convertor)

โดยปกติแล้วเซ็นเซอร์จะรับสัญญาณทางกายภาพเข้ามา ซึ่งจะได้เป็นสัญญาณแอนalog การที่บอร์ดไมโครคอนโทรลเลอร์จะสามารถทำงานได้จะต้องทำการแปลงสัญญาณจากแอนalog เป็นดิจิทัล (Analog to Digital Convertor: ADC) สำหรับบอร์ด NodeMCU ESP8266 นั้นจะมีขา A0 เป็นขาอินพุตแบบแอนalog 1 ช่อง โดยมีความละเอียดในการแปลงสัญญาณขนาด 10 bit หรือ 2^{10} หรือ 1024 ระดับ ซึ่งสามารถระบุข้อมูลตัวเลขแบบแอนalog ได้ตั้งแต่ 0 – 1023 ระดับ ซึ่งสามารถเทียบแรงดันบนบอร์ดได้ตั้งแต่ 0 – 3.3 V ซึ่งสามารถคำนวณได้จากการสมการต่อไปนี้

$$\text{Analog Level} = \frac{1023 \times V_{in}}{V_{max}}$$

เมื่อ Analog Level คือ ระดับของสัญญาณแอนalog มีค่าตั้งแต่ 0 – 1023

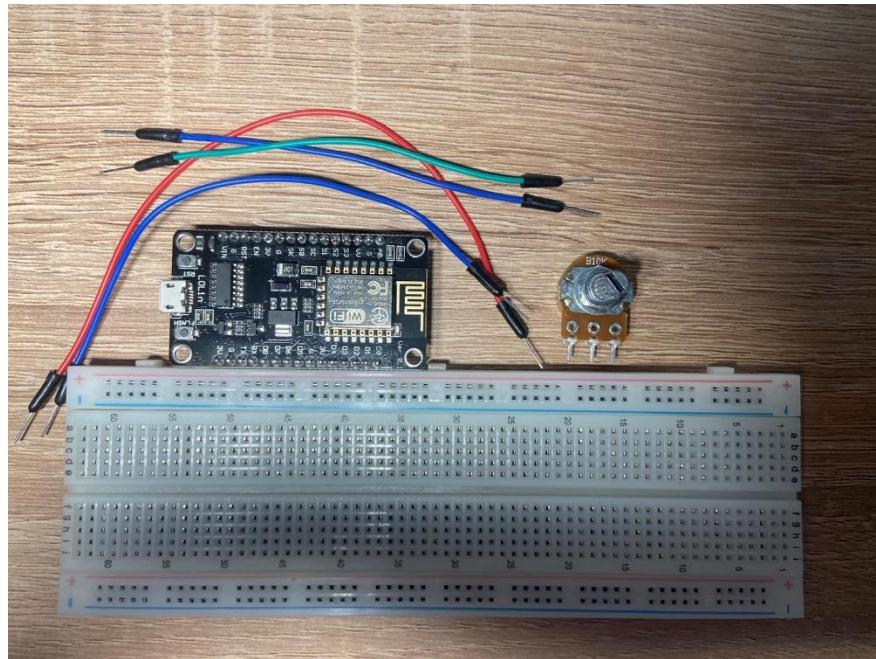
V_{in} คือ ค่าแรงดันขาเข้า (ค่าสัญญาณจากเซ็นเซอร์)

V_{max} คือ ค่าแรงดันสูงสุดของบอร์ด ซึ่งสำหรับบอร์ด NodeMCU ESP8266 มีค่าที่ 3.3 V

Lab 4.1: การวัดสัญญาณแอนalog

อุปกรณ์ที่ต้องใช้

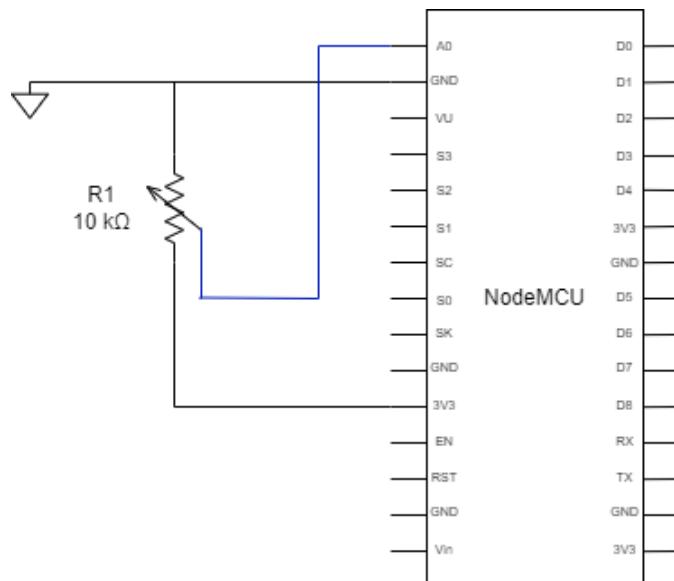
- บอร์ด NodeMCU 1 ตัว
- ตัวต้านทานปรับค่าได้ขนาด 10 kΩ 1 ตัว
- สาย Jumper
- Breadboard 1 อัน



รูปที่ 4.1 อุปกรณ์ที่ต้องใช้สำหรับแลบ 4.1

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 4.2 วงจรของแลบที่ 4.1

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
const int analogPin = A0;
int sensorValue = 0;

void setup() {
    Serial.begin(115200);
}

void loop() {
    sensorValue = analogRead(analogPin);
    Serial.println(sensorValue);
    delay(50);
}
```

- เปิด Serial Monitor พร้อมทำการปรับค่าตัวด้านท่านปรับค่าได้โดยการหมุน
- ปิดหน้าต่าง Serial Monitor และเปิดหน้าต่าง Serial Plotter พร้อมทำการปรับค่าตัวด้านท่านแบบปรับค่าได้โดยการหมุน

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

Lab 4.2: กำหนดช่วงของเอาต์พุตด้วยฟังก์ชัน map

function map -> map(value, fromLow, fromHigh, toLow, toHigh);

เมื่อ value คือ ค่าที่ต้องการแปลง

fromLow คือ ค่าตัวเลขต่ำสุดของช่วงเดิม

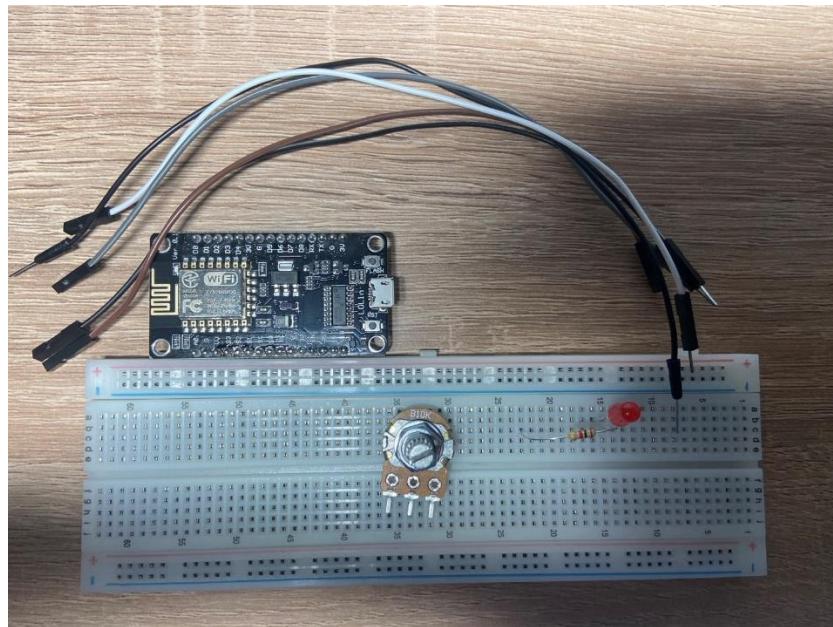
fromHigh คือ ค่าตัวเลขสูงสุดของช่วงเดิม

toLow คือ ค่าตัวเลขต่ำสุดของช่วงใหม่

toHigh คือ ค่าตัวเลขสูงสุดของช่วงใหม่

อุปกรณ์ที่ต้องใช้

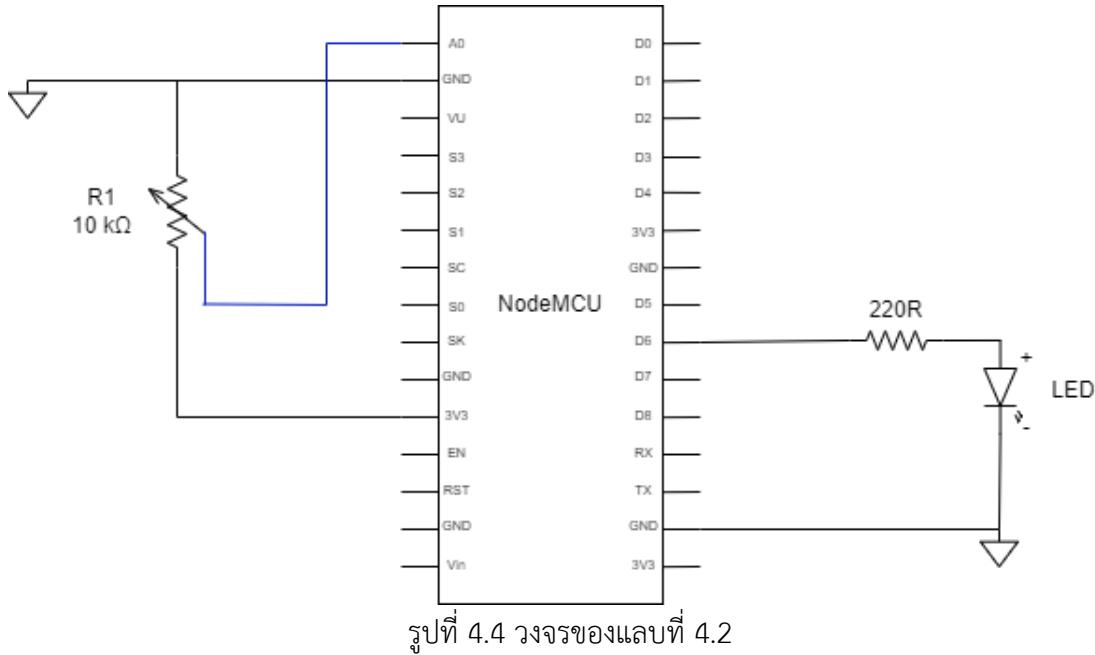
- บอร์ด NodeMCU 1 ตัว
- หลอด LED 1 อัน
- ตัวต้านทาน 220 Ω 1 ตัว
- ตัวต้านทานปรับค่าได้ขนาด 10 kΩ 1 ตัว
- สาย Jumper
- Breadboard 1 อัน



รูปที่ 4.3 อุปกรณ์ที่ต้องใช้สำหรับแลบ 4.2

ขั้นตอนในการทำ

- จากการจารเดิม ให้เพิ่มการเชื่อมหลอด LED ดังรูป



- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```

const int analogPin = A0;
int sensorValue = 0;
int newValue = 0;

void setup() {
    Serial.begin(115200);
    pinMode(D6, OUTPUT);
}

void loop() {
    sensorValue = analogRead(analogPin);
    newValue = map(sensorValue, 0, 1023, 100, 1500);

    Serial.println(newValue);

    digitalWrite(D6, HIGH);
    delay(newValue);
    digitalWrite(D6, LOW);
    delay(50);
}

```

- เปิด Serial Monitor พร้อมทำการปรับค่าตัวต้านทานปรับค่าได้โดยการหมุน
- ปิดหน้าต่าง Serial Monitor และเปิดหน้าต่าง Serial Plotter พร้อมทำการปรับค่าตัวต้านทานแบบปรับค่าได้โดยการหมุน

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

บทที่ 5

การเข้ารหัสข้อมูลระหว่างอุปกรณ์

ในการส่งต่อข้อมูลระหว่างอุปกรณ์จะมีการจัดทำเป็นรูปแบบของ binary คือ 0 กับ 1 โดยจะมีการเข้ารหัส (Encoding) ข้อมูลก่อนนำส่งไปยังอีกอุปกรณ์หนึ่ง ซึ่งการส่งผ่านข้อมูลจะถูกแบ่งออกเป็น 2 ประเภท คือ

1. การส่งข้อมูลแบบอนุกรม (Serial Transmission) เป็นการส่งแบบเรียงบิตทีละบิตไปยังปลายทาง ซึ่งสามารถแบ่งได้ 2 ประเภท คือ
 - a. การส่งข้อมูลแบบอะซิงโครนัส (Asynchronous Transmission) เป็นการส่งข้อมูลแบบไม่ออาศัยสัญญาณนาฬิกา
 - b. การส่งข้อมูลแบบซิงโครนัส (Synchronous Transmission) เป็นการส่งข้อมูลแบบอาศัยสัญญาณนาฬิกา
2. การส่งข้อมูลแบบขนาน (Parallel Transmission) เป็นการส่งข้อมูลทีละหลาย ๆ บิตไปยังปลายทาง ใน 1 รอบสัญญาณนาฬิกา

NodeMCU ESP8266 จะมีการสื่อสารระหว่างโมดูลหรืออุปกรณ์อื่น ๆ โดยใช้วิธีการส่งข้อมูลแบบอนุกรม ผ่านโปรโตคอล (Protocol) ซึ่งเป็นมาตรฐานร่วมกัน ซึ่งมีนิยมอยู่ 3 รูปแบบ คือ

1. Universal Asynchronous Receiver Transmitter (UART): เป็นการสื่อสารอนุกรมแบบไม่ใช้สัญญาณนาฬิกา แต่ใช้ Baud Rate ในการกำหนดอัตราการรับส่งข้อมูล โดยใช้ขา RX เป็นขาเข้า และขา TX เป็นขาออก
2. Serial Peripheral Interface (SPI): เป็นการสื่อสารอนุกรมแบบใช้สัญญาณนาฬิกา สามารถรับส่งข้อมูลได้ 2 ทาง (Full-Duplex) โดยจะมีการเข้ารหัสข้อมูลแบบ Master-Slave
3. Inter-Integrated Circuit (I2C): เป็นการสื่อสารอนุกรมแบบใช้สัญญาณนาฬิกา เป็นการรับสื่อสารข้อมูลแบบกึ่งสองทาง (Half-Duplex) โดยจะมีการเข้ารหัสข้อมูลแบบ Master-Slave

Lab 5.1: การสื่อสารแบบ UART

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว

ขั้นตอนในการทำ

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
void setup() {
    Serial.begin(9600);
    delay(100);
    Serial.println("Fill your name and then press enter");
}

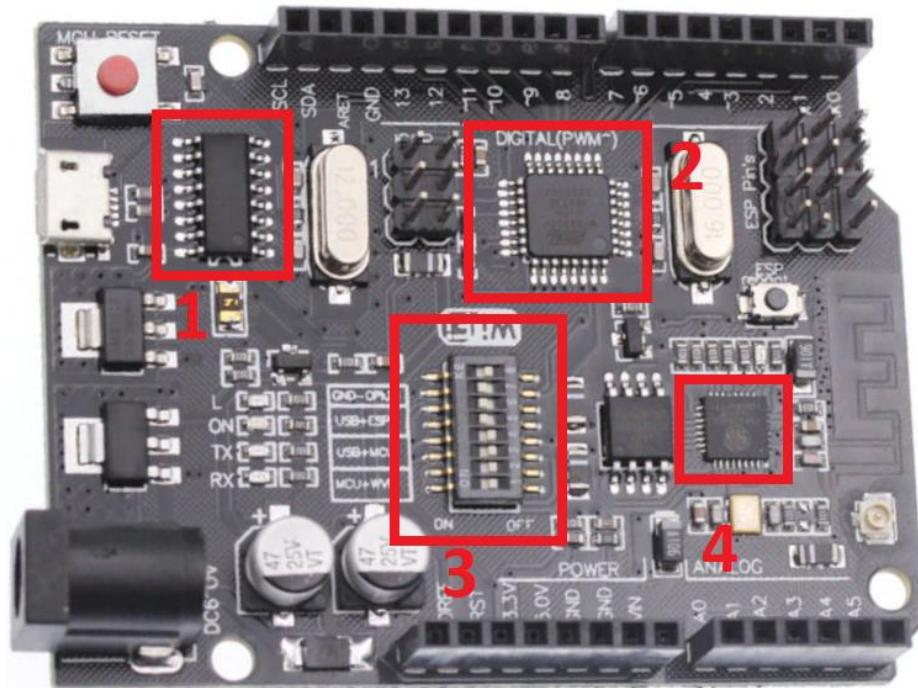
void loop() {
    if (Serial.available()) {
        String name = Serial.readString();
        Serial.print("My name is ");
        Serial.println(name);
    }
}
```

- เปิด Serial Monitor ตั้งค่า Baud Rate ให้ตรงกับโค้ด
- ทดลองกดปุ่ม RST บนบอร์ด
- ทดลองพิมพ์ชื่อลงใน Serial Monitor และกด Enter เพื่อส่งข้อมูลไปยังบอร์ด

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

Lab 5.2: การสื่อสารแบบ SPI

การทดลองนี้จะเป็นการทดสอบการส่งข้อมูลไปมาระหว่าง NodeMCU ESP8266 กับ Arduino Uno ผ่าน SPI bus โดยจะกำหนดให้ NodeMCU ESP8266 เป็น Master และ Arduino Uno เป็น Slave โดยจะทำการทดลองบนบอร์ด built-in WiFi module Uno R3 + WiFi ATmega328P + ESP8266



รูปที่ 5.1 บอร์ด built-in WiFi module Uno R3 + WiFi ATmega328P + ESP8266 พร้อมแสดงรายละเอียดที่สำคัญของบอร์ด ที่มา: <https://blog.devgenuis.io/programming-arduino-uno-clone-with-built-in-wifi-module-uno-r3-wifi-atmega328p-esp8266-9494d9a90cfa>

จากรูปที่ 5.1 จะแสดงองค์ประกอบสำคัญบนบอร์ด ได้แก่

1. CH340 USB-to-serial adapter: ชิปโมดูลในการแปลง USB เป็น UART TTL
2. ATmega328P: ชิปไมโครคอนโทรลเลอร์สำหรับ Arduino Uno
3. Dip Switch: สวิตซ์สำหรับสลับโหมดการทำงานของบอร์ด
4. ESP8266: ชิปไมโครคอนโทรลเลอร์สำหรับ NodeMCU

เนื่องจากมีชิปบนบอร์ดทั้งหมด 2 ชิป ในการ flash โปรแกรมลงบนชิปซึ่งจะต้องทำผ่านพอร์ต USB และทำการแปลงโดยใช้ชิป CH340 ซึ่งมีอยู่เพียงชุดเดียวบนบอร์ดจึงไม่สามารถทำพร้อมกันได้ Dip Switch จึงเป็นอุปกรณ์ที่นำมาช่วยในการจัดการดังที่แสดงในตารางด้านล่าง [4]

ตารางที่ 5.1 ตารางแสดงการสับ Dip Switch เพื่อให้ทำงานในโหมดต่าง ๆ

Mode	1	2	3	4	5	6	7
CH340<-->ESP8266 for uploading sketch	OFF	OFF	OFF	OFF	ON	ON	ON
CH340<-->ESP8266 serial connection	OFF	OFF	OFF	OFF	ON	ON	OFF
CH340<-->ATmega328 for uploading sketch	OFF	OFF	ON	ON	OFF	OFF	OFF
Mega328<-->ESP8266	ON	ON	OFF	OFF	OFF	OFF	OFF
No connection, all chips work independently	OFF						

อุปกรณ์ที่ต้องใช้

- บอร์ด built-in WiFi module Uno R3 + WiFi ATmega328P + ESP8266 1 ตัว

ขั้นตอนในการทำ

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด

```
//NodeMCU Master
#include <SPI.h>
#define CS 10

void setup() {
    pinMode(CS, OUTPUT);
    digitalWrite(CS, HIGH);
    SPI.begin();
    Serial.begin(115200);
    delay(100);
}

void loop() {
    digitalWrite(CS, LOW);
    byte received = SPI.transfer(0x02);
    digitalWrite(CS, HIGH);
    Serial.print("Received from Slave: ");
    Serial.println(received);
    delay(1000);
}
```

- ตั้งค่าบอร์ดในโปรแกรม Arduino IDE ให้เป็นบอร์ด NodeMCU 1.0
- สลับสวิตซ์บนบอร์ดให้อยู่ในโหมด CH340<-->ESP8266 for uploading sketch
- อัปโหลดโค้ดลงบนบอร์ด
- เมื่ออัปโหลดเสร็จ สร้างไฟล์ใหม่ใน Arduino IDE
- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด

```
//Arduino UNO R3 Slave
#include <SPI.h>

volatile byte data;

void setup() {
    pinMode(MISO, OUTPUT
    SPCR |= _BV(SPE);
    Serial.begin(115200);
    delay(100);
}

ISR(SPI_STC_vect) {
    data = SPDR;
    SPDR = data + 1;
}

void loop() {
    Serial.print("Data received from master: ");
    Serial.println(data);
    delay(1000);
}
```

- ตั้งค่าบอร์ดในโปรแกรม Arduino IDE ให้เป็นบอร์ด Arduino Uno WiFi
- สลับสวิตซ์บนบอร์ดให้อยู่ในโหมด CH340<-->ATmega328 for uploading sketch
- อัปโหลดโค้ดลงบนบอร์ด
- เมื่ออัปโหลดเสร็จให้เปิด Serial Monitor ในไฟล์โค้ดผู้ง slave เพื่อดูผลลัพธ์ (หากไม่มีข้อมูลขึ้นบน serial monitor ให้กดปุ่ม RST บนบอร์ด 1 ครั้งเพื่อ reset ให้บอร์ดเริ่มทำงานใหม่)
- ปิด Serial Monitor ของโค้ดผู้ง slave
- สลับสวิตซ์บนบอร์ดให้อยู่ในโหมด CH340<-->ESP8266 serial connection
- ที่โค้ดผู้ง master เปิด Serial Monitor เพื่อดูผลลัพธ์ (หากไม่มีข้อมูลขึ้นบน serial monitor ให้กดปุ่ม RST บนบอร์ด 1 ครั้งเพื่อ reset ให้บอร์ดเริ่มทำงานใหม่)

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

5.3 การสื่อสารแบบ I2C

อุปกรณ์ที่ต้องใช้

- บอร์ด built-in WiFi module Uno R3 + WiFi ATmega328P + ESP8266 1 ตัว
- บอร์ด บอร์ด NodeMCU 1 ตัว
- หลอด LED 1 อัน
- ตัวต้านทานขนาด $220\ \Omega$ 1 อัน
- สาย Jumper
- Breadboard 1 อัน

ขั้นตอนในการทำ

- ใช้บอร์ด built-in WiFi module Uno R3 + WiFi ATmega328P + ESP8266 สำหรับอุปกรณ์ Slave
- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด

```
//Arduino UNO R3 Slave
#include <Wire.h>

void setup() {
    pinMode(13, OUTPUT);
    Wire.begin(43);
}

void loop() {
    Wire.onReceive(myHandler);
    delay(300);
}

void myHandler(int numByte) {
    while (Wire.available()) {
        char c = Wire.read();
        if (c == '1') {
            digitalWrite(13, HIGH);
        } else if (c == '0') {
            digitalWrite(13, LOW);
        }
    }
}
```

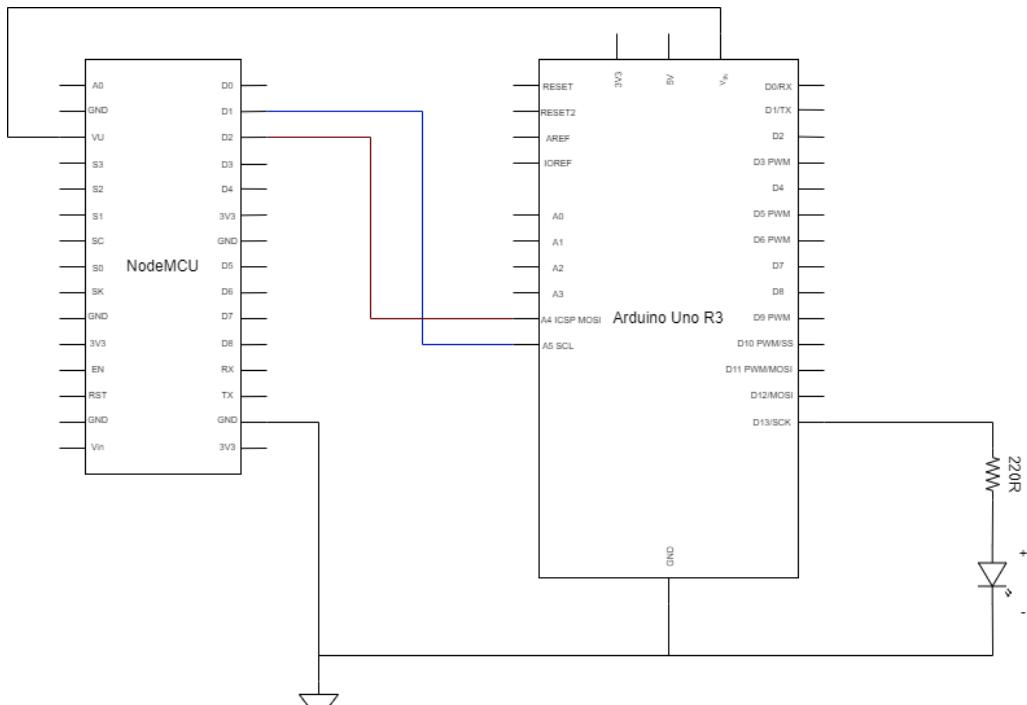
- ตั้งค่าบอร์ดในโปรแกรม Arduino IDE ให้เป็นบอร์ด Arduino Uno WiFi
- สลับสวิตซ์บนบอร์ดให้อยู่ในโหมด CH340<-->ATmega328 for uploading sketch
- อัปโหลดโค้ดลงบนบอร์ด
- ถอดสาย USB จากบอร์ดเดิมแล้วเปลี่ยนมาเสียบที่บอร์ด NodeMCU เพื่อสร้างไฟล์สำหรับอุปกรณ์ master
- สร้างไฟล์ใหม่ใน Arduino IDE
- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด

```
//NodeMCU Master
#include <Wire.h>

void setup() {
    Serial.begin(115200);
    Wire.begin();
}

void loop() {
    if (Serial.available()) {
        String data = Serial.readString();
        Wire.beginTransmission(43);
        if (data.indexOf("ON") != -1) {
            Wire.write('1');
        } else if (data.indexOf("OFF") != -1) {
            Wire.write('0');
        }
        Serial.print("You sent: ");
        Serial.println(data);
        Wire.endTransmission();
    }
}
```

- ตั้งค่าบอร์ดในโปรแกรม Arduino IDE ให้เป็นบอร์ด NodeMCU 1.0
- อัปโหลดโค้ดลงบนบอร์ด
- ต่อวงจรดังที่แสดงในรูป



รูปที่ 5.2 วงจรของแลบที่ 5.3

- เปิด Serial Monitor
 - ทดลองพิมพ์ ON และ OFF ส่งผ่าน Serial Monitor ไปยังบอร์ด Master สังเกตผลที่เกิดขึ้น

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

บทที่ 6

การขัดการทำงาน (Interrupt)

การขัดการทำงาน (Interrupt) คือ การร้องขอให้ตัวประมวลผลหยุดทำงานในขณะที่กำลังดำเนินการอยู่ โดยทำการเก็บค่าที่ดำเนินการอยู่แล้วสุดไว้ในหน่วยความจำ และเปลี่ยนไปทำงานคำสั่งที่ถูกร้องขอเข้ามา เมื่อดำเนินการเสร็จ ระบบจะกลับไปประมวลผลงานเดิมที่ค้างอยู่ ซึ่งสามารถนำไปประยุกต์ใช้งานได้หลายรูปแบบ ทำให้ระบบมีประสิทธิภาพในการทำงานโดยไม่สูญเสียเวลา many ซึ่งการขัดการทำงานสามารถแบ่งได้เป็น 2 ประเภท คือ

1. External Interrupt: การขัดการทำงานจากภายนอก เช่น การเปลี่ยนสถานะ loyal ของพาร์ตไดพอร์ตหนึ่ง
2. Internal Interrupt: การขัดการทำงานจากภายใน เช่น การขัดการทำงานที่เกิดจากตัวนับเวลา (Timer)

โดยสามารถควบคุมการขัดการทำงานได้ 2 รูปแบบ คือ

1. Disable Interrupt: การควบคุมให้ค่อนໂටຣເລອວ່າມີຕອບສນອງຕ່ອງการขัดการทำงาน เมื่อเกิดการขัดการทำงาน ค่อนໂටຣເລອວ່າຈະປ່ອຍຜ່ານ
2. Enable Interrupt: การควบคุมให้ค่อนໂටຣເລອວ່າຕອບສນອງຕ່ອງการขัดการทำงาน

เงื่อนไขในการขัดการทำงานมี 3 รูปแบบ คือ

1. CHANGE: การเปลี่ยนสถานะจาก 0 เป็น 1 หรือจาก 1 เป็น 0
2. RISING: การเปลี่ยนสถานะจาก 0 เป็น 1
3. FALLING: การเปลี่ยนสถานะจาก 1 เป็น 0

ชิ้นบอร์ด ESP8266 สามารถตั้งการขัดการทำงานได้ทุกขา GPIO ยกเว้นขา GPIO16 และมีคำสั่งที่ใช้สำหรับการขัดการทำงาน 2 คำสั่ง คือ

1. attachInterrupt() ใช้สำหรับการเริ่มตรวจสอบ Interrupt
2. detachInterrupt() ใช้สำหรับการยกเลิกตรวจสอบ Interrupt

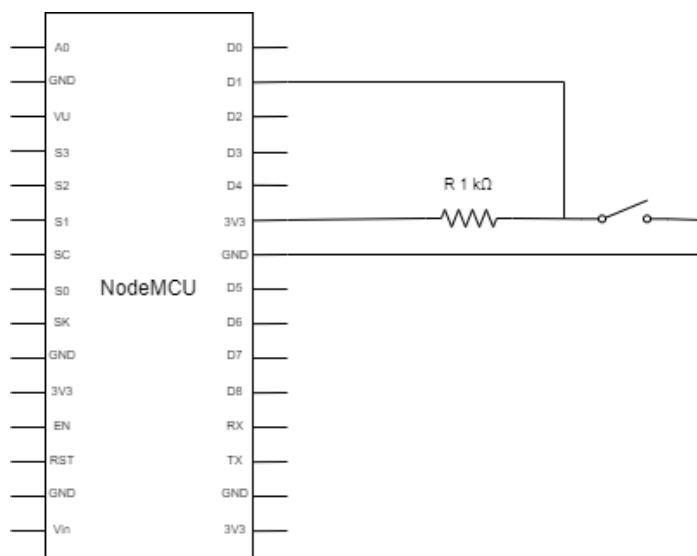
Lab 6.1: Interrupt CHANGE

อุปกรณ์ที่ต้องใช้

- บอร์ด บอร์ด NodeMCU 1 ตัว
- สวิตช์กดติดปล่อยดับ 1 อัน
- ตัวต้านทานขนาด $1\text{ k}\Omega$ 1 อัน
- สาย Jumper
- Breadboard 1 อัน

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 6.1 วงจรสำหรับแล็บที่ 6.1

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```

const int buttonPin = D1;
volatile bool interruptFlag = false;
volatile int lastState = HIGH;

void ICACHE_RAM_ATTR handleInterrupt() {
    interruptFlag = true;
}

void setup() {
    Serial.begin(115200);
    pinMode(buttonPin, INPUT_PULLUP);

    attachInterrupt(digitalPinToInterrupt(buttonPin), handleInterrupt, CHANGE);
}

void loop() {
    if (interruptFlag) {
        int currentState = digitalRead(buttonPin);
        if (currentState != lastState) {
            Serial.print("State changed to: ");
            Serial.println(currentState == HIGH ? "HIGH" : "LOW");
            lastState = currentState;
        }
        interruptFlag = false;
    }
}

```

- เปิด Serial Monitor ตั้งค่า Baud Rate ให้ตรงกับโค้ดแล้วทดลองกดปุ่ม

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

Lab 6.2: Interrupt FALLING

ขั้นตอนในการทำ

- ใช้งานเดิมกับแลบที่ 6.1
- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
const int buttonPin = D1;
volatile bool interruptFlag = false;

void ICACHE_RAM_ATTR handleInterrupt() {
    interruptFlag = true;
}

void setup() {
    Serial.begin(115200);
    pinMode(buttonPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(buttonPin), handleInterrupt,
    FALLING);
}

void loop() {
    if (interruptFlag) {
        Serial.println("Interrupt Triggered!");
        interruptFlag = false;
    }
}
```

- เปิด Serial Monitor ตั้งค่า Baud Rate ให้ตรงกับโค้ดแล้วทดลองกดปุ่ม

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

Lab 6.3: Interrupt RISING

ขั้นตอนในการทำ

- ใช้งานเดิมกับแลบที่ 6.1
- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
const int buttonPin = D1;
volatile bool interruptFlag = false;

void ICACHE_RAM_ATTR handleInterrupt() {
    interruptFlag = true;
}

void setup() {
    Serial.begin(115200);
    pinMode(buttonPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(buttonPin), handleInterrupt, RISING);
}

void loop() {
    if (interruptFlag) {
        Serial.println("Rising edge detected!");
        interruptFlag = false;
    }
}
```

- เปิด Serial Monitor ตั้งค่า Baud Rate ให้ตรงกับโค้ดแล้วทดลองกดปุ่ม

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

บทที่ 7

การแสดงผลผ่านโมดูลจอ

สำหรับระบบ IoT การแสดงผลผ่านจอ LCD นั้นได้รับความนิยมและนำไปใช้อย่างกว้างขวาง ซึ่งปัจจุบันมีโมดูลจอ LCD ที่นิยมใช้อยู่ 2 ประเภท คือ

1. Character LCD เป็นจอที่ใช้แสดงผลตัวอักษร ซึ่งมีหลายขนาดที่วางขายตามท้องตลาด เช่น โมดูลจอ 16x2 หรือ 1602 Character LCD เป็นต้น
2. Graphic LCD หรือ OLED เป็นจอที่ใช้ในการแสดงผลเป็นตัวอักษรและภาพกราฟิก ซึ่งมีความละเอียดและสามารถแสดงเป็นสีสันได้ โดยระบุขนาดเป็นจำนวนจุด (pixel) เช่น โมดูลจอ OLED 128x64 0.96" คือ โมดูลจอ OLED ที่มีจำนวนจุดแนวนอน 128 จุด และแนวตั้ง 64 จุด โดยมีขนาดความยาวหน้าจอ 0.96 นิ้ว เป็นต้น

การสื่อสารระหว่างโมดูลจอ Character LCD กับไมโครคอนโทรลเลอร์ ตามปกตินั้นจะเป็นการส่งข้อมูลแบบขนาน (Parallel Transmission) ซึ่งต้องใช้ขาทั้งหมด 16 ขาในการสื่อสาร ซึ่งเป็นวิธีที่ไม่เป็นที่นิยมเนื่องจากมีจำนวนสายที่ค่อนข้างมาก จึงทำให้เกิดอุปกรณ์เสริมเพื่อทำการแปลงการสื่อสารเป็นรูปแบบอนุกรม (Serial Transmission) โดยทำการเชื่อมต่อผ่านโปรโตคอล I2C ส่วนการสื่อสารระหว่างโมดูลจอ OLED กับไมโครคอนโทรลเลอร์นั้นสามารถสื่อสารผ่านโปรโตคอล SPI หรือ I2C ก็ได้ เนื่องจากชิปในโมดูลนั้นรองรับทั้ง 2 โปรโตคอล



(ก)



(ข)

รูปที่ 7.1 โมดูลจอ LCD (ก) 1602 Character LCD (ข) Graphic LCD หรือ OLED ที่มา:

<https://www.thaieasyelec.com/category/lcd-oled/96>

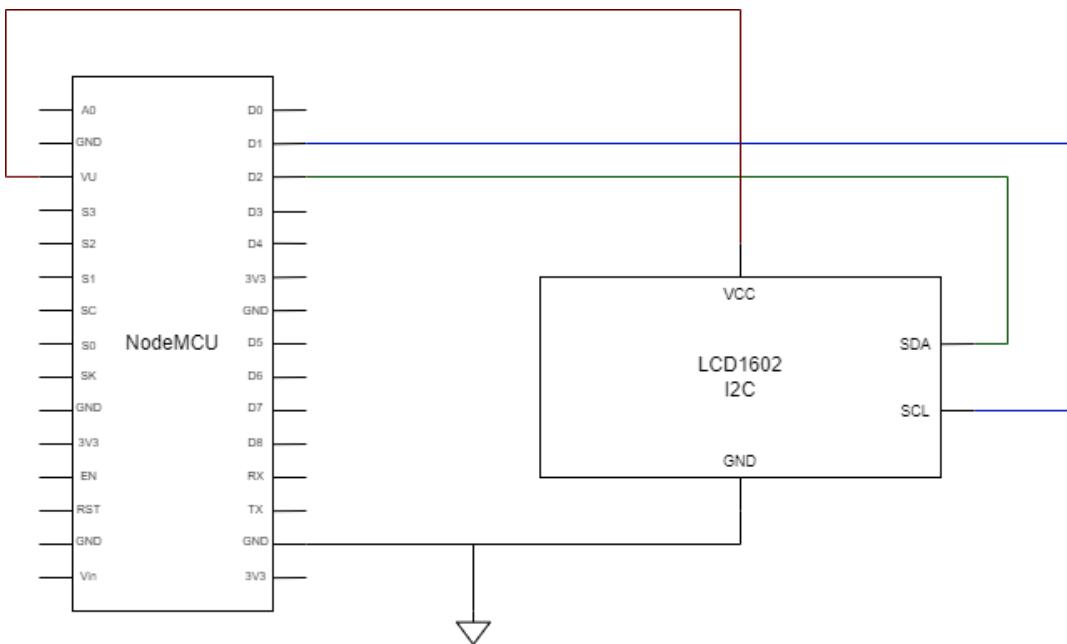
Lab 7.1: การคืนหา Address ของโมดูลจอ LCD

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- โมดูลจอ 1603 Character LCD 1 ตัว
- สาย Jumper
- Breadboard 1 อัน

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 7.1 วงจรสำหรับแล็บที่ 7.1

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#include <Wire.h>

void setup() {
    Wire.begin();
    Serial.begin(115200);
    while (!Serial);
}

void loop() {
    Serial.println("\nScanning...");
    for (byte i = 8; i < 120; i++) {
        Wire.beginTransmission(i);

        if (Wire.endTransmission() == 0) {
            Serial.print("LCD Module Address: ");
            Serial.print(i, DEC);
            Serial.print(" (0x");
            Serial.print(i, HEX);
            Serial.println(")\n");
        }
    }
    delay(5000);
}
```

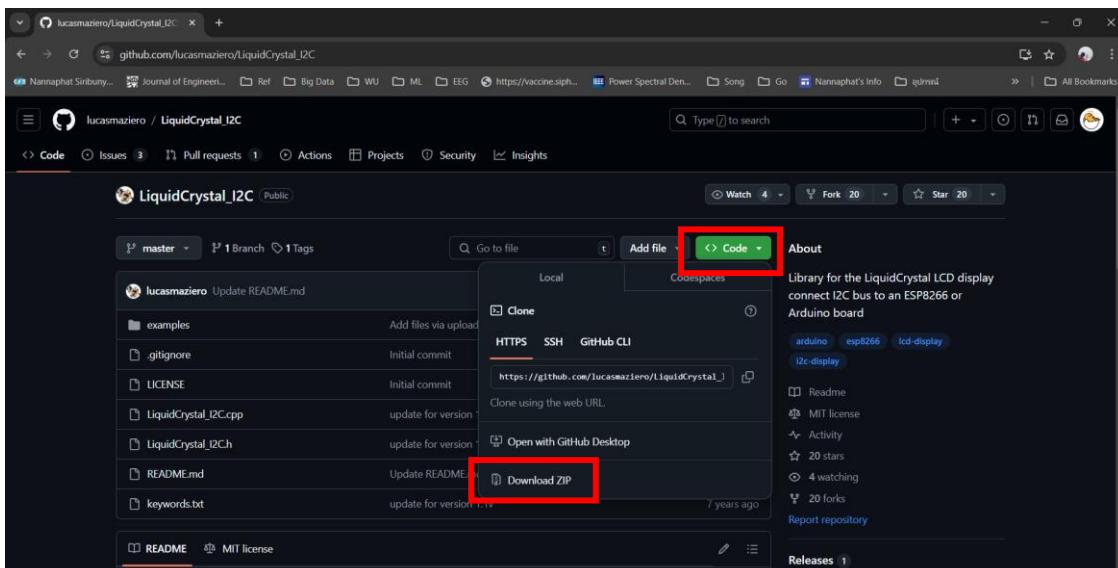
- เปิด Serial Monitor เพื่อดูผลลัพธ์ที่เกิดขึ้น

ผลการทำงานหลังจากอปป์โหลดโค้ดลงบอร์ด

Lab 7.2: การแสดงผลบนจอ LCD

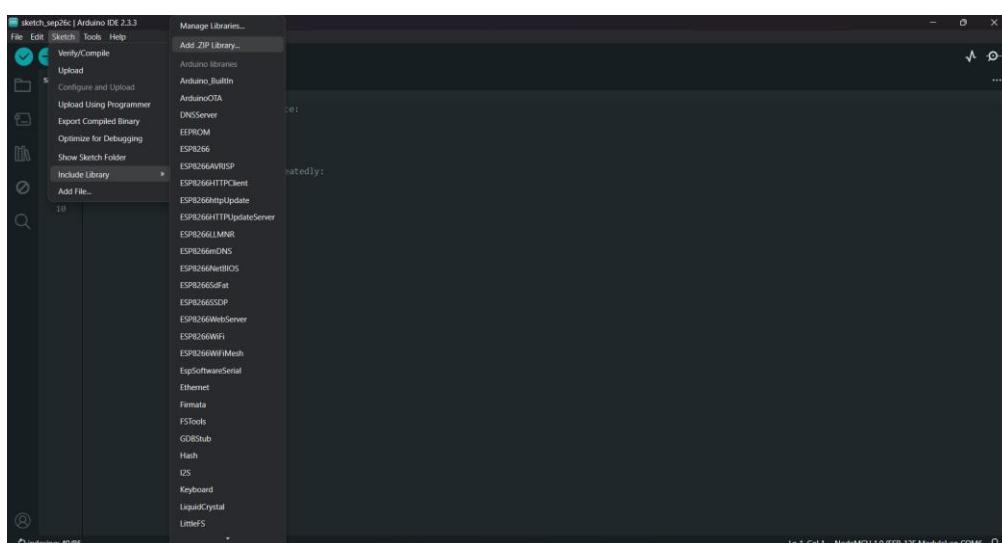
ขั้นตอนในการทำ

- ใช้งานจริงเดิมจากแลบที่ 7.1
- ไปที่เว็บไซต์ https://github.com/lucasmaziero/LiquidCrystal_I2C จากนั้นกดที่ปุ่ม <> Code แล้วทำการดาวน์โหลดไฟล์ ZIP ของไลบรารี LiquidCrystal_I2C



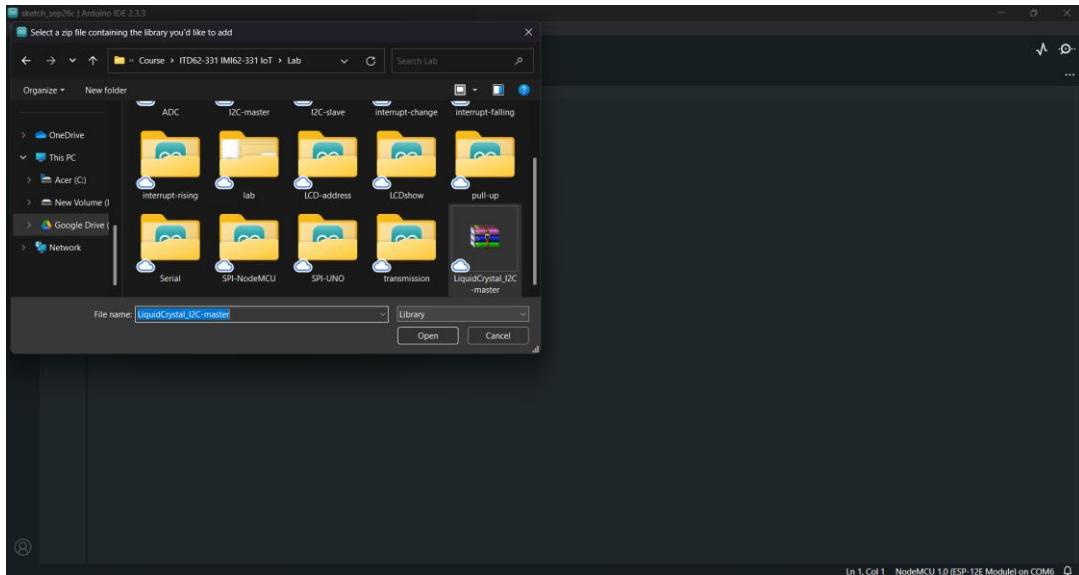
รูปที่ 7.2 เว็บไซต์สำหรับดาวน์โหลดไลบรารี LiquidCrystal_I2C

- สร้างไฟล์ใหม่ใน Arduino IDE ไปที่เมนู Sketch -> Include Library -> Add .ZIP Library...



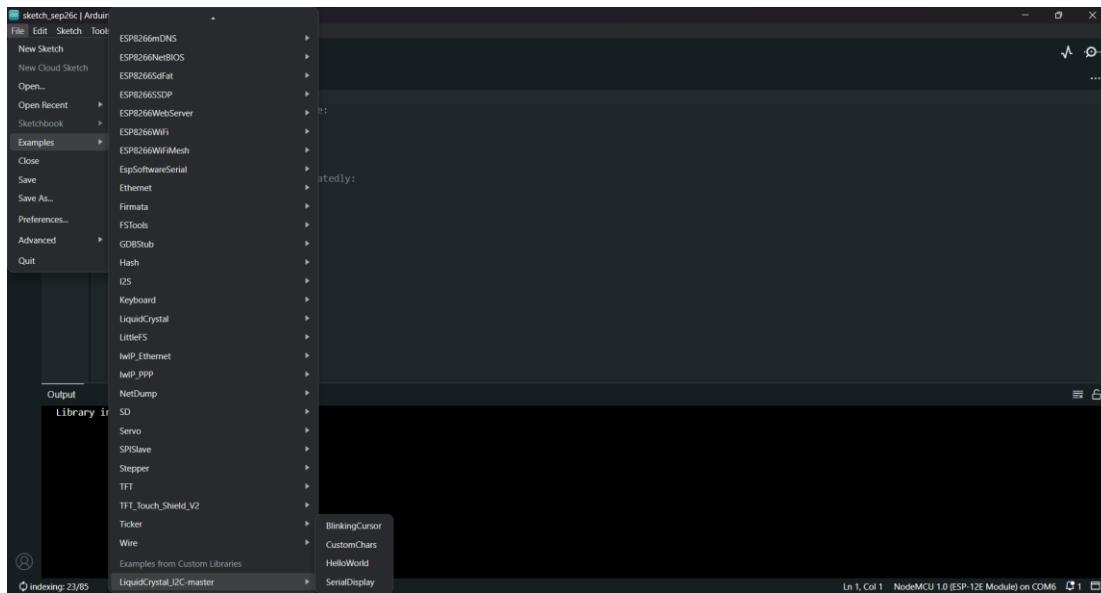
รูปที่ 7.3 การเข้าเมนูเพื่อติดตั้งไลบรารีที่ดาวน์โหลดมา

- เลือกไฟล์ไลบรารีของ LiquidCrystal_I2C ที่ดาวน์โหลดมา (ในตัวอย่าง คือ LiquidCrystal_I2C-master) จากนั้นกด Open



รูปที่ 7.4 การติดตั้งไลบรารี

- ไปที่เมนู File -> Examples เลือนหาชื่อไลบรารีที่ติดตั้ง หากพบ แสดงว่าการติดตั้งเสร็จสมบูรณ์



รูปที่ 7.4 การตรวจสอบการติดตั้งไลบรารี

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
byte heart[8] = {0x00, 0x0A, 0x1F, 0x1F, 0x0E, 0x04,
0x00, 0x00};

void setup() {
    lcd.begin();
}

void loop() {
    lcd.backlight();
    lcd.display();
    lcd.home();

    delay(1000);
    lcd.print("We love");
    delay(2000);

    lcd.setCursor(0, 1);
    lcd.print("IoT");
    delay(2000);

    lcd.createChar(0, heart);
    lcd.setCursor(4, 1);
    lcd.write(0);
    delay(2000);

    for (int i = 0; i <= 15; i++) {
        lcd.scrollDisplayLeft();
        delay(200);
    }
    for (int i = 0; i <= 32; i++) {
        lcd.scrollDisplayRight();
        delay(200);
    }
    for (int i = 0; i <= 16; i++) {
        lcd.scrollDisplayLeft();
        delay(200);
    }
    delay(2000);
    lcd.noDisplay();
    delay(500);
    lcd.display();
    delay(500);
    lcd.noDisplay();
    delay(500);
    lcd.display();
    delay(500);
    lcd.noDisplay();
    delay(500);
    lcd.display();
    delay(1000);

    lcd.clear();
    delay(1000);
    lcd.noDisplay();
    delay(1000);
    lcd.noBacklight();
    delay(2000);
}
```

ผลการทำงานหลังจากอปเปิลเด็คิดลงบอร์ด

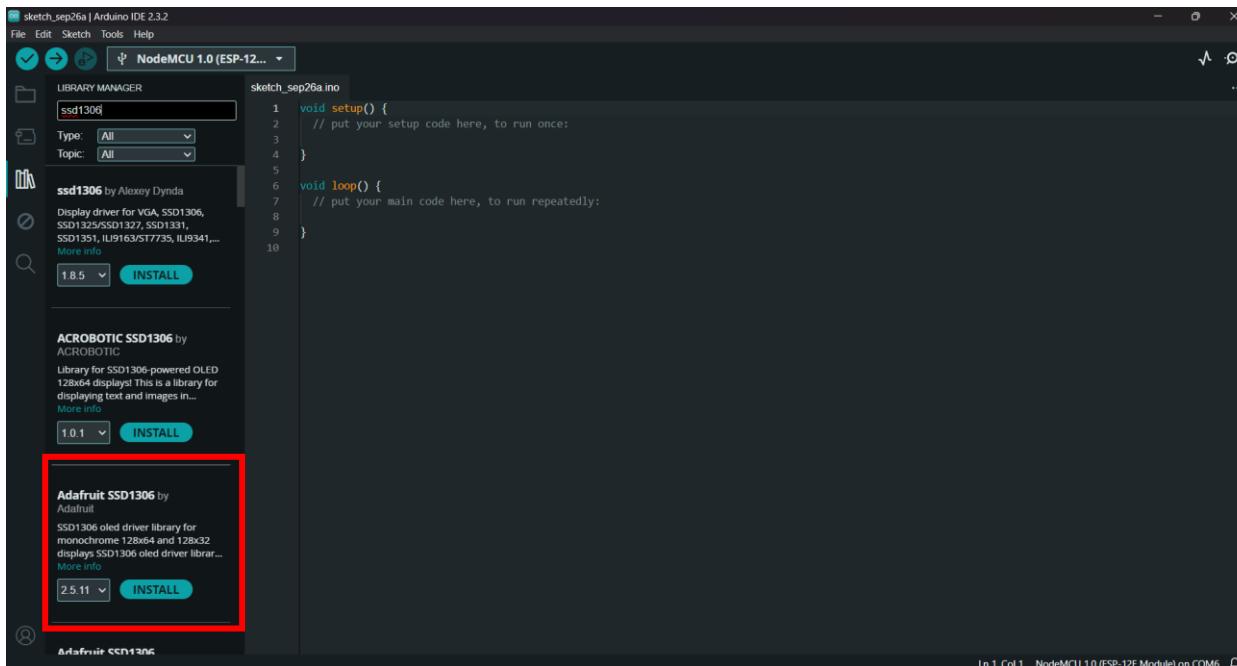
Lab 7.3: การแสดงผลบนโมดูลจอ OLED

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- โมดูลจอ 1603 Character LCD 1 ตัว
- สาย Jumper
- Breadboard 1 อัน

ขั้นตอนในการทำ

- ที่ Arduino IDE คลิกเมนู Tools -> Manage Library...
- ค้นหา ssd1306 และทำการติดตั้งライบรารี Adafruit SSD1306



รูปที่ 7.5 ไลบรารี Adafruit SSD1306

- ไปที่ตำแหน่งเก็บไฟล์ไลบรารี ไปที่ไฟล์เดอร์ไลบรารี Adafruit_SSD1306 เปิดไฟล์ Adafruit_SSD1306.h ด้วย Notepad
- แก้ไขไฟล์ดังนี้แล้วทำการเซฟ
 - //define SSD1306_128_64 -> #define SSD1306_128_64
 - #define SSD1306_128_32 -> //define SSD1306_128_32

```

File Edit View
File Edit View
* This is part of Adafruit's SSD1306 library for monochrome
* OLED displays: http://www.adafruit.com/category/63_98
*
* These displays use I2C or SPI to communicate. I2C requires 2 pins
* (SCL/SDA) and optionally a RESET pin. SPI requires 4 pins (MOSI, SCK,
* select, data/command) and optionally a reset pin. Hardware SPI or
* 'bitbang' software SPI are both supported.
*
* Adafruit invests time and resources providing this open source code,
* please support Adafruit and open-source hardware by purchasing
* products from Adafruit!
*
* Written by Limor Fried/ladyada for Adafruit Industries, with
* contributions from the open source community.
*
* BSD license, all text above, and the splash screen header file,
* must be included in any redistribution.
*
*/
#ifndef Adafruit_SSD1306_H
#define Adafruit_SSD1306_H

#define SSD1306_128_64 /* DEPRECATED: old way to specify 128x64 screen
//define SSD1306_128_32 /* DEPRECATED: old way to specify 128x32 screen
//define SSD1306_96_16 /* DEPRECATED: old way to specify 96x16 screen
//define SSD1306_64_16 /* DEPRECATED: old way to specify 64x16 screen
// (NEW CODE SHOULD IGNORE THIS, USE THE CONSTRUCTORS THAT ACCEPT WIDTH
// AND HEIGHT ARGUMENTS).

// Uncomment to disable Adafruit splash logo
//define SSD1306_NO_SPLASH

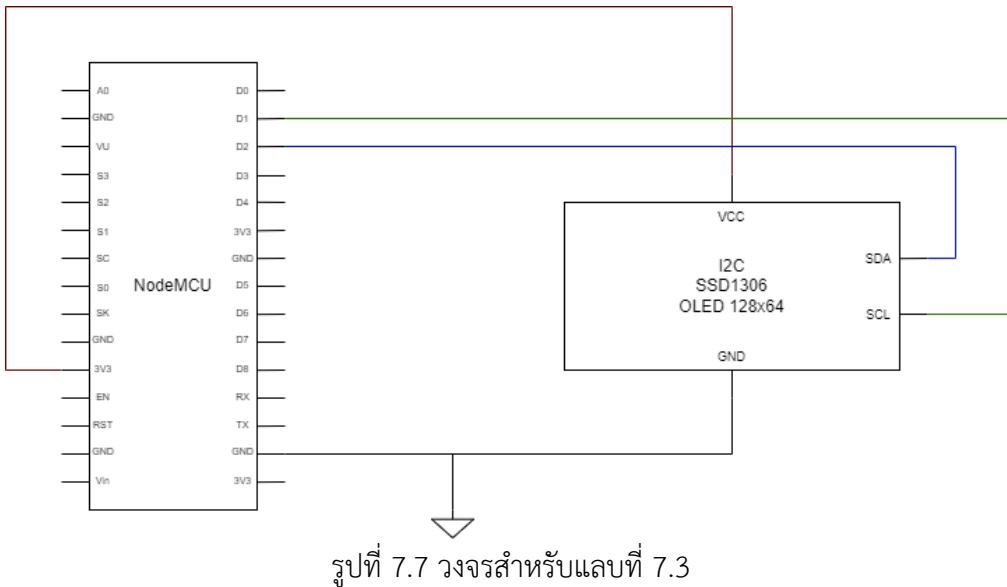
#if defined(ARDUINO_STM32_FEATHER)
typedef class HardwareSPI SPIClass;
#endif

#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>

```

รูปที่ 7.6 แก้ไขไฟล์ Adafruit_SSD1306.

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 7.7 วงจรสำหรับแลบที่ 7.3

- ทำการตรวจสอบ Address ของโมดูลจอ OLED จาก Lab 7.1
- พิมพ์โค้ดต่อไปนี้พร้อมเติมค่า Address ที่ได้ทำการหักขั้นตอนก่อนหน้า แล้วทำการ Verify โค้ด แล้วอัปโหลดโค้ดลงบอร์ด

```
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS เดิมค่า Address ตรงนี้
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, OLED_RESET);

void setup() {
    display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
    display.clearDisplay();

    display.setTextSize(2);
    display.setTextColor(WHITE);
    display.setCursor(0, 20);
    display.println("Hello");

    display.display();
}

void loop() {
}
```

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

- ทำการแก้ไขโค้ดในบรรทัดที่ 17 โดยเปลี่ยนจาก WHITE เป็น BLACK, WHITE แล้วทำการ Verify โค้ดแล้วอัปโหลดโค้ดลงบอร์ด

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

- จากโค้ดด้านบน แก้ไขโค้ดในบรรทัดที่ 17 กลับมาเป็น WHITE เพิ่มเติมโค้ดต่อไปนี้โดยเริ่มใส่ที่บรรทัดที่ 22 ซึ่งจะต้องอยู่ในพังก์ชัน void setup() จากนั้นทำการ Verify โค้ด แล้วอัปโหลดโค้ดลงบอร์ด

```
delay(2000);
display.clearDisplay();

display.setTextSize(1);
display.setCursor(0, 28);
display.print("0x");
display.print(0xE6, HEX);
display.print("(HEX) = ");
display.print(0xE6, DEC);
display.println("(DEC)");
display.display();
```

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

- จากโค้ดด้านบน เพิ่มเติมโค้ดต่อไปนี้โดยเริ่มใส่ที่บรรทัดที่ 33 ซึ่งจะต้องอยู่ในฟังก์ชัน void setup()
จากนั้นทำการ Verify โค้ด แล้วอปโหลดโค้ดลงบอร์ด

```
delay(2000);
display.clearDisplay();

display.setCursor(0, 0);
display.println("Show");
display.println("Display");
display.println("Scrolling!");
display.display();
display.startscrollright(0x00, 0x07);
delay(4500);
display.stopscroll();
delay(1000);
display.startscrollleft(0x00, 0x07);
delay(4500);
display.stopscroll();
delay(1000);
display.startscrollright(0x00, 0x00);
delay(6000);
```

ผลการทำงานหลังจากอปโหลดโค้ดลงบอร์ด

- จากโค้ดด้านบน เพิ่มเติมโค้ดต่อไปนี้โดยเริ่มใส่ที่บรรทัดที่ 51 ซึ่งจะต้องอยู่ในฟังก์ชัน void setup()
จากนั้นทำการ Verify โค้ด แล้วอปโหลดโค้ดลงบอร์ด

```
display.clearDisplay();

display.setTextColor(WHITE);
display.setCursor(0, 0);
display.println("Filled Circle");
display.fillCircle(64, 35, 20, WHITE);

display.display();
delay(2000);
display.clearDisplay();
```

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

Lab 7.4: การแสดงภาพกราฟิกบนโมดูลจอ OLED

ขั้นตอนในการทำ

- ใช้งานเดิมจากแล็บที่ 7.3
- ไปที่เว็บไซต์ <https://javl.github.io/image2cpp/>
- เปิดรูปภาพที่ต้องการจะแสดงบนโมดูลจอ OLED และตั้งค่าดังที่แสดงในรูป

image2cpp

image2cpp is a simple tool to change images into byte arrays (or arrays back into images) for use with (monochrome) displays such as OLEDs on your Arduino or Raspberry Pi. It was originally made to work with the Adafruit OLED library (for which you can find an example sketch for Arduino [here](#)) but has been expanded by the community to be useful in all kind of (embedded) projects.

More info (and credits) can be found in the [Github repository](#). This is also where you can report any [issues](#) you might come across.

Did you find this tool useful? Feel free to support my open source software on Github

1. Select image

All processing is done locally in your browser; your images are not uploaded or stored anywhere online.

Choose Files **aesthetic-c...av5u06.jpg**
aesthetic-computer-4k-megrsbupwyav5u06.jpg [remove](#)

1. Paste byte array

128 x 64 px
[Read as horizontal](#) [Read as vertical](#)
Read images appear at step 3 below

2. Image Settings

Canvas size(s): aesthetic-computer-4k-megrsbupwyav5u06.jpg (file resolution: 900 x 506)
128 x 64 glyph [remove](#)

Background color: White Black Transparent

Invert image colors:

Dithering:

Brightness / alpha threshold: 128
0 - 255; If the brightness of a pixel is above the given level the pixel becomes white, otherwise they become black. When using alpha, opaque and transparent are used instead.

Scaling:
 horizontally vertically

Center image: Centering the image only works when using a canvas larger than the original image.

Rotate image: 0 degrees
 horizontally vertically

Flip image:

3. Preview



4. Output

Code output format

Arduino code

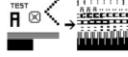
Adds some extra Arduino code around the output for easy copy-paste into [this example](#). If multiple images are loaded, generates a byte array for each and appends a counter to the identifier.

Identifier/Prefix: test

Draw mode:

Horizontal - 1 bit per pixel

If your image looks all messed up on your display, like the image below, try using a different mode.



Swap bits in byte:

swap

Useful when working with the u8g2 library.

รูปที่ 7.8 การตั้งค่ารูปเพื่อแปลงเป็นโค้ดภาษาซี

- กดปุ่ม Generate Code และกดปุ่ม Copy Output เพื่อคัดลอกโค้ดที่ทำการแปลงแล้ว

Generate code **Copy Output** Download as binary file (.bin)

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x07, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0x1f, 0xff, 0xe1, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01f, 0xff, 0xff, 0xf0, 0x0f, 0x81, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xe0, 0x0f, 0x81, 0x80
};

// Array of all bitmaps for convenience. (Total bytes used to store images in PROGMEM = 1040)
const int testallArray_LEN = 1;
const unsigned char* testallArray[1] = {
    testaesthetic_computer_4k_megrbsupwyav5u06
};
```

รูปที่ 7.9 ตัวอย่างรูปที่ถูกแปลงเป็นโค้ดแล้ว

- พิมพ์โค้ดต่อไปนี้แล้วนำโค้ดรูปภาพที่ได้จากเว็บไซต์มาใส่ในบรรทัดที่ 13 และแก้ (identifier) ให้เป็นชื่อตามที่โค้ดกำหนด จากนั้นทำการ Verify โค้ด แล้วอัปโหลดโค้ดลงบอร์ด

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C // ใส่ค่า Address ของโมดูลจอ OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// นำไปต่อรูปภาพที่แปลงมาใส่ตั้งแต่บรรทัดที่ 13

void setup() {
    display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
    display.clearDisplay();

    //identifier ได้จากการแปลงรูปภาพแล้วจะอยู่ในบรรทัดต่อจาก const unsigned char* testallArray[1]
    display.drawBitmap(0, 0, (identifier), 128, 64, WHITE);

    display.display();
}

void loop() {
    // put your main code here, to run repeatedly:
}
```

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

บทที่ 8

การใช้งานเซ็นเซอร์

เซ็นเซอร์เป็นอุปกรณ์ที่ใช้ในการตรวจจับสภาพแวดล้อมต่าง ๆ ที่อยู่รอบตัว เปรียบเทียบได้กับประสิทธิภาพในการรับรู้ของมนุษย์ ซึ่งโดยทั่วไปจะเป็นข้อมูลแบบแอนalog ทำให้ต้องมีการแปลงสัญญาณจากแอนalog เป็นดิจิทัลจึงสามารถนำมาประมวลผลในระบบ IoT ได้ ซึ่งในการทดลองนี้จะนำเอาเซ็นเซอร์พื้นฐานมาใช้ในการเรียนรู้

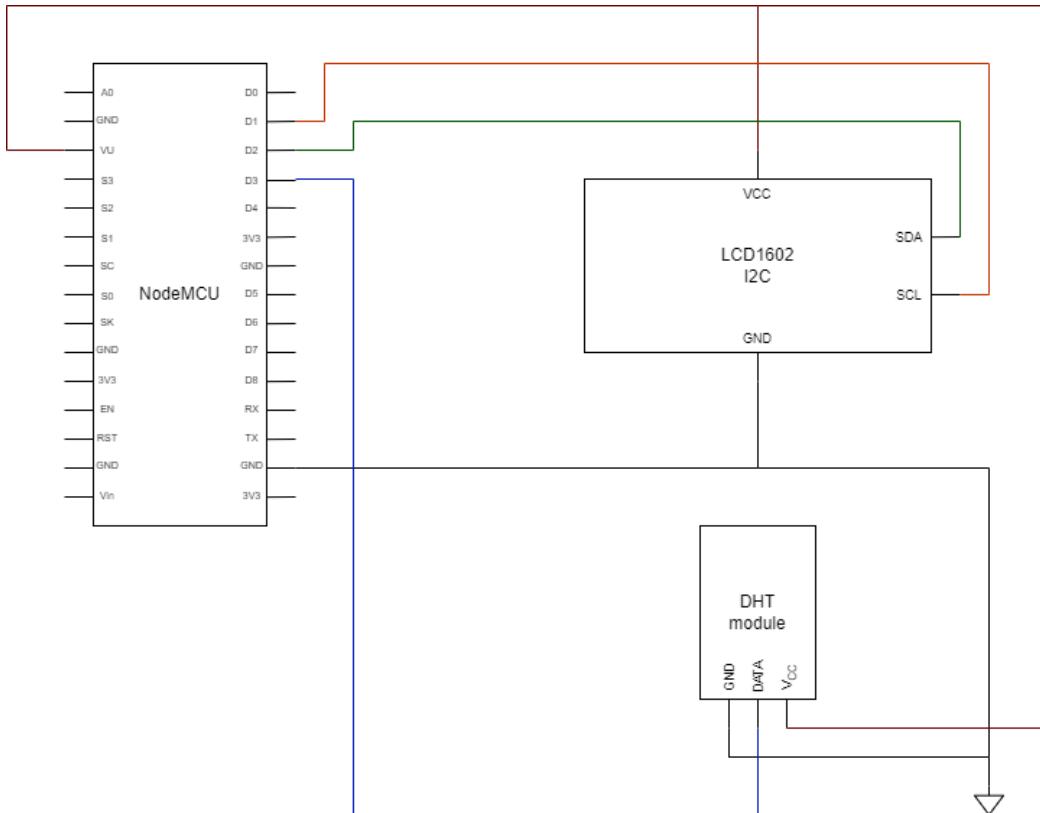
Lab 8.1: การวัดอุณหภูมิและความชื้นในอากาศด้วยโมดูล DHT

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- โมดูลจอ 1603 Character LCD 1 ตัว
- โมดูล DHT 1 ตัว
- สาย Jumper
- Breadboard 1 อัน

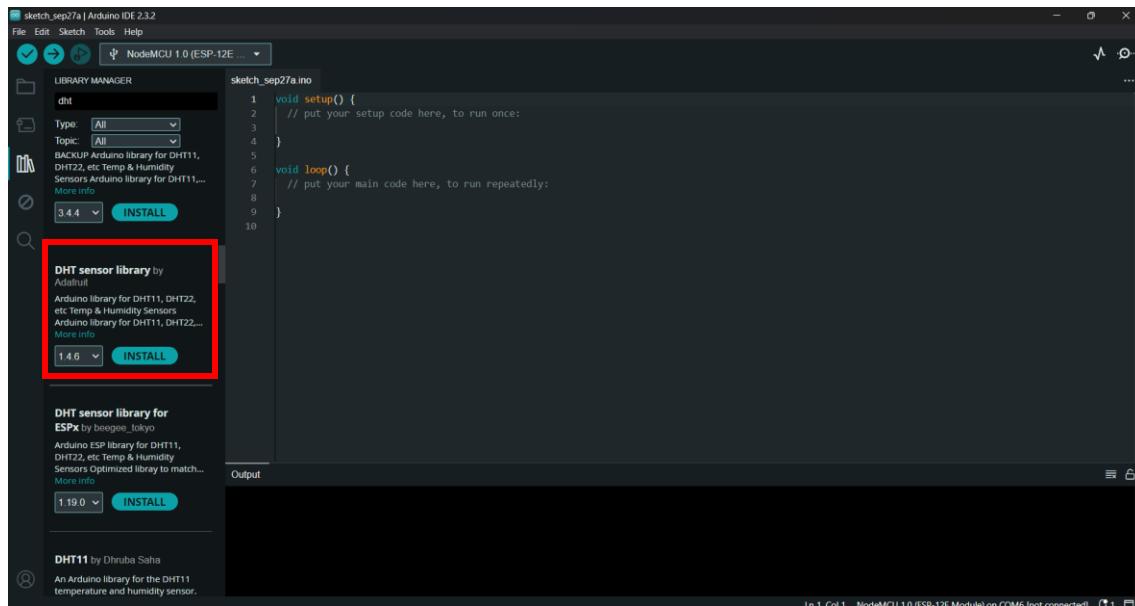
ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 8.1 วงจรสำหรับแลบที่ 8.1

- ไปที่ Tools -> Manage Libraries... ค้นหาคำว่า dht และทำการติดตั้งไลบรารี dht ของ Adafruit ที่ชื่อว่า DHT sensor library by Adafruit



รูปที่ 8.2 ไลบรารีสำหรับโมดูล DHT

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

#define DHTPIN D3
#define DHTTYPE DHT11

LiquidCrystal_I2C lcd(0x27, 16, 2);

DHT dht(DHTPIN, DHTTYPE);

void setup() {
    dht.begin();
    lcd.begin();
    lcd.backlight();
    lcd.print("DHT11 TEST");
}

void loop() {
    delay(2000);
    float t = dht.readTemperature();
    float h = dht.readHumidity();

    if (isnan(t) || isnan(h)) {
        lcd.clear();
        lcd.print("DHT Failed");
    } else {
        lcd.setCursor(0, 0);
        lcd.print("Temp: ");
        lcd.setCursor(7, 0);
        lcd.print(t);
        lcd.setCursor(13, 0);
        lcd.print("*C");
        lcd.setCursor(0, 1);
        lcd.print("Humid: ");
        lcd.print(h);
        lcd.setCursor(14, 1);
        lcd.print("%");
    }
    delay(1000);
}
```

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

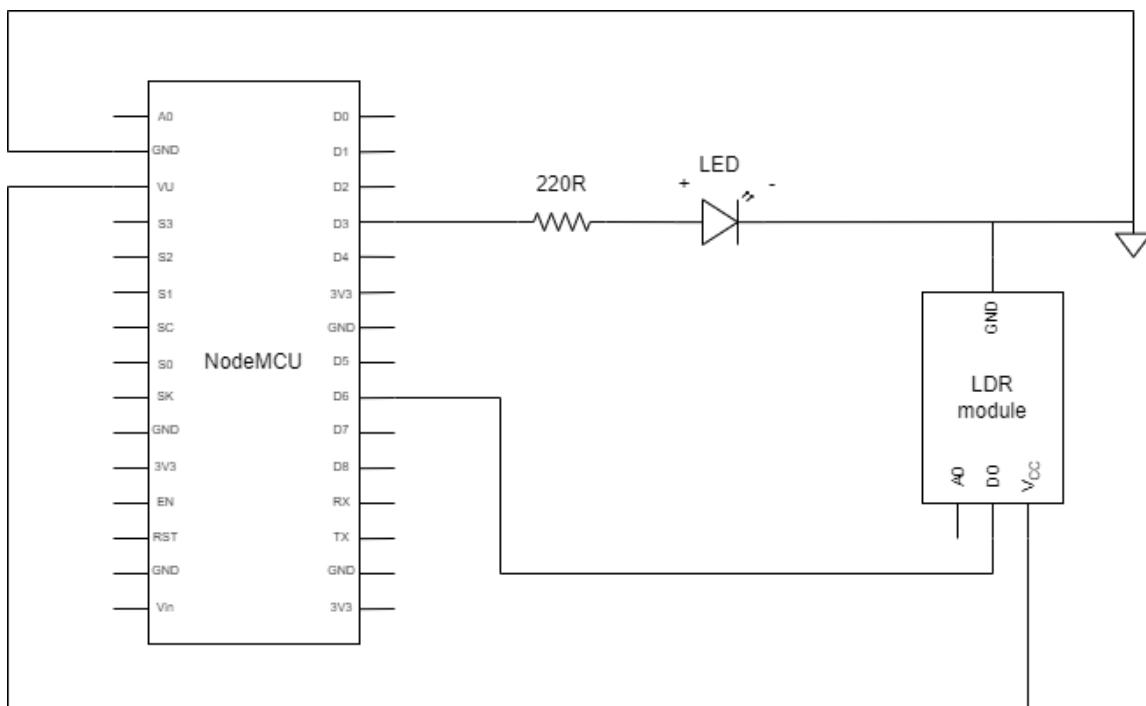
Lab 8.2.: การเปิด-ปิดหลอดไฟด้วยโมดูล LDR

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- หลอด LED 1 อัน
- โมดูล LDR 1 ตัว
- ตัวต้านทานขนาด 220Ω 1 อัน
- สาย Jumper
- Breadboard 1 อัน

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 8.3 วงจรสำหรับแลบที่ 8.2

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
const int ledPin = D3;
const int ldrPin = D6;

void setup() {
    Serial.begin(112500);
    pinMode(ledPin, OUTPUT);
    pinMode(ldrPin, INPUT);
}

void loop() {
    int ldrStatus = digitalRead(ldrPin);
    if (ldrStatus == 1) {
        digitalWrite(ledPin, HIGH);
        Serial.println("LDR --> Dark, LED --> ON");
    } else {
        digitalWrite(ledPin, LOW);
        Serial.println("LDR --> Bright, LED --> OFF");
    }
}
```

- เปิด Serial Monitor เพื่อดูผลการเปลี่ยนแปลง ทดลองเอาไฟส่องที่เซ็นเซอร์หรือปิดที่เซ็นเซอร์ รวมถึง ทดลองปรับค่าตัวต้านทานปรับค่าได้ที่โมดูลเซ็นเซอร์ สังเกตการเปลี่ยนแปลง

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

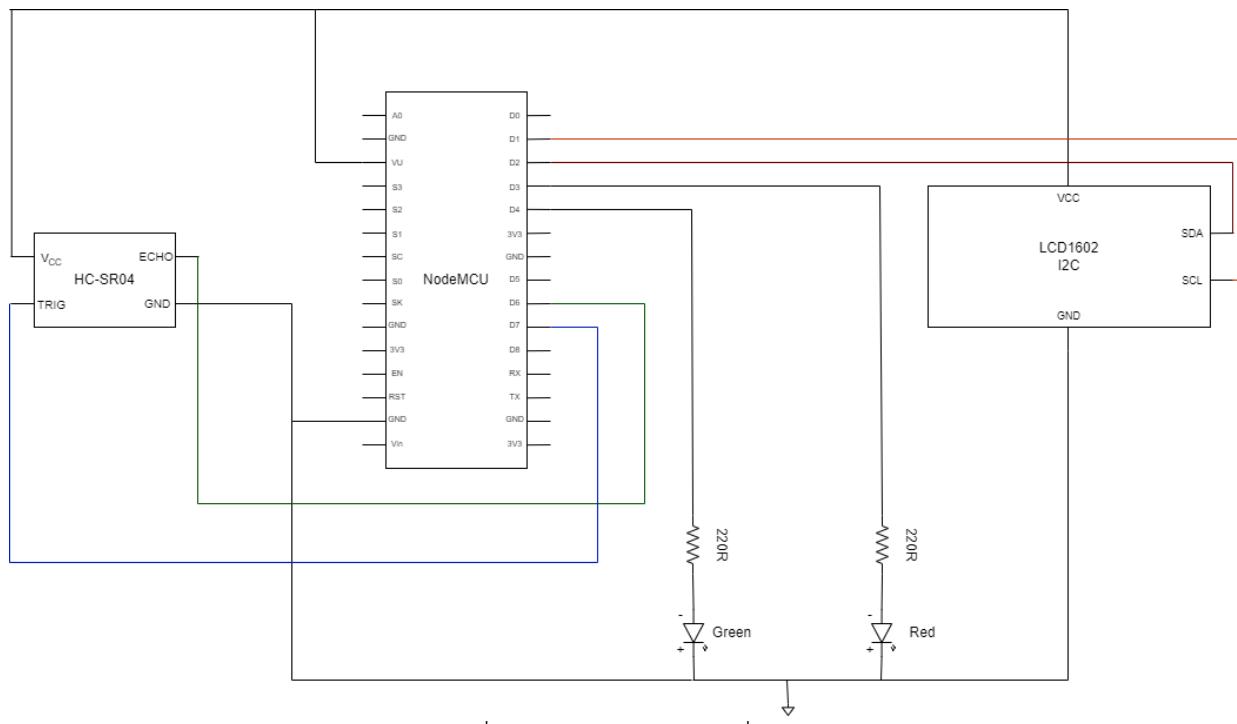
Lab 8.3: การวัดระยะทางและแจ้งเตือนด้วยโมดูล Ultrasonic Sensor

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- โมดูลจอ 1603 Character LCD 1 ตัว
- โมดูล Ultrasonic Sensor HC-SR04 1 ตัว
- หลอด LED สีแดงและสีเขียว อายุ่งละ 1 หลอด
- ตัวต้านทานขนาด $220\ \Omega$ 2 อัน
- สาย Jumper
- Breadboard 1 อัน

ขั้นตอนในการทำ

- ต่อวงจรตั้งที่แสดงในรูป



รูปที่ 8.4 วงจรสำหรับแล็บที่ 8.3

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

#define TRIG D7
#define ECHO D6
#define RED_PIN D3
#define GREEN_PIN D4

long duration, distance;

void setup() {
    pinMode(TRIG, OUTPUT);
    pinMode(ECHO, INPUT);
    pinMode(RED_PIN, OUTPUT);
    pinMode(GREEN_PIN, OUTPUT);

    lcd.begin();
    lcd.backlight();
    lcd.home();

    lcd.print("Check Distance");
    lcd.setCursor(0, 1);
    lcd.print("Ready");
    lcd.clear();
    delay(2000);
}

void loop() {
    digitalWrite(TRIG, LOW);
    delayMicroseconds(5);

    digitalWrite(TRIG, HIGH);
    delayMicroseconds(10);

    digitalWrite(TRIG, LOW);

    duration = pulseIn(ECHO, HIGH);
    distance = duration * 0.034 / 2;

    lcd.setCursor(0, 0);
    lcd.print("Distance: ");
    lcd.print(distance);
    lcd.print("cm. ");

    if (distance >= 15) {
        digitalWrite(GREEN_PIN, HIGH);
        digitalWrite(RED_PIN, LOW);
        lcd.setCursor(0, 1);
        lcd.print("Safe");
    } else {
        digitalWrite(GREEN_PIN, LOW);
        digitalWrite(RED_PIN, HIGH);
        lcd.setCursor(0, 1);
        lcd.print("Stop");
    }

    delay(500);
}
```

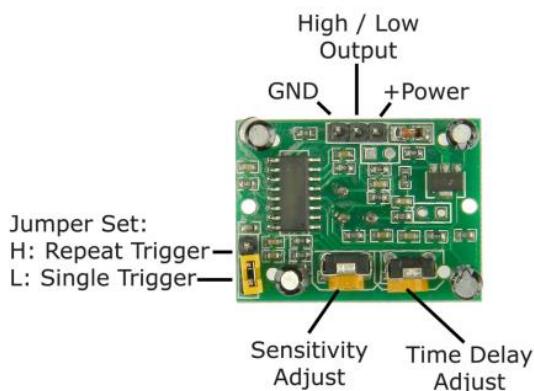
- ทดลองเอวัตถุกับบริเวณเซ็นเซอร์ในแนวตั้งจากกับเซ็นเซอร์ แล้วขับเข้าออก

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

Lab 8.4: ตรวจจับการเคลื่อนไหวด้วยโมดูล PIR Motion Sensor

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- โมดูล PIR HC-SR501 1 ตัว
- Buzzer 1 ตัว
- หลอด LED 1 หลอด
- ตัวต้านทานขนาด $220\ \Omega$ 1 อัน
- สาย Jumper
- Breadboard 1 อัน



รูปที่ 8.4 องค์ประกอบของโมดูล PIR HC-SR501

สำหรับ Sensitive Adjust กับ Time Delay Adjust เป็นตัวต้านทานที่สามารถหมุนเพื่อปรับค่าได้ โดยมีความหมายดังนี้

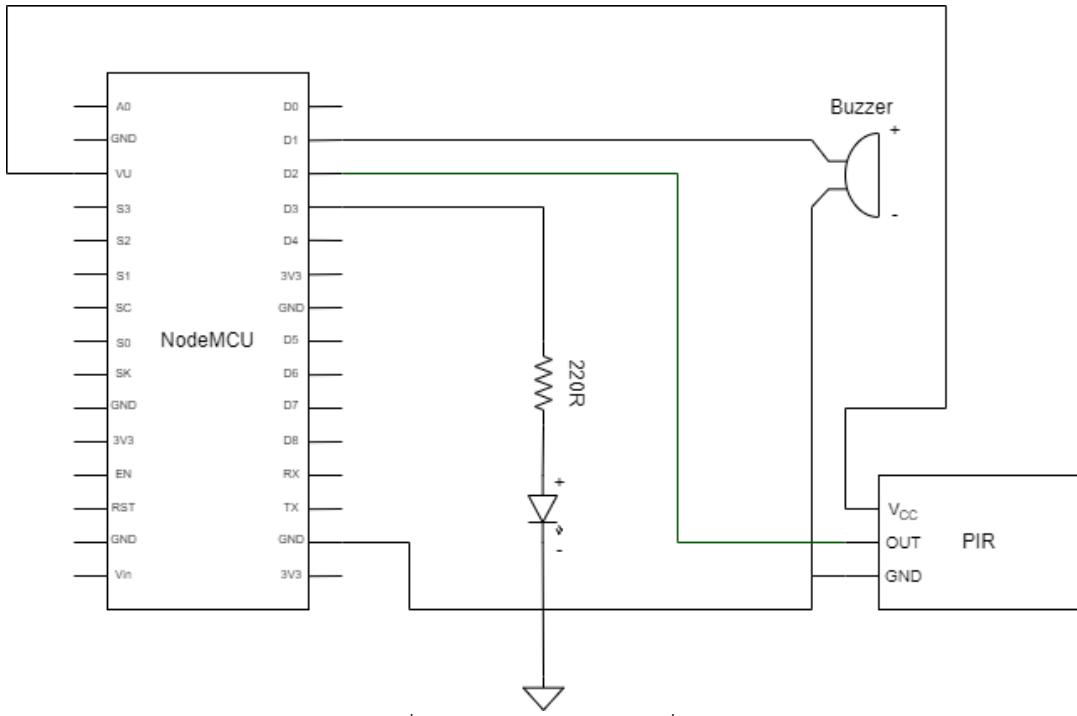
ตารางที่ 8.1 การปรับค่าความไวและการหน่วงเวลา

ตัวต้านทาน	หมุนตามเข็มนาฬิกา (Clockwise)	หมุนทวนเข็มนาฬิกา (Anti Clockwise)
Sensitive Adjust	เพิ่มค่าความไว ตรวจจับในรัศมีระยะประมาณ 7 เมตร	ลดค่าความไว ตรวจจับในรัศมีระยะประมาณ 3 เมตร
Time Delay Adjust	เพิ่มระยะเวลาคงสถานะของขา Digital OUT สูงสุดประมาณ 300 วินาที	ลดระยะเวลาคงสถานะของขา Digital OUT ต่ำสุดประมาณ 3 วินาที

Jumper Set ใช้ในการทำหนدโหมดในการทำงานของเซ็นเซอร์ ถ้ากำหนดเป็น H (Repeat Trigger) เมื่อเซ็นเซอร์ตรวจจับการเคลื่อนไหวอย่างต่อเนื่องจะคงสถานะ HIGH จนกว่าจะหยุดเคลื่อนไหวหรือพื้นรัศมีการตรวจจับ ส่วนถ้ากำหนดเป็น L (Single Trigger) เมื่อเซ็นเซอร์ตรวจจับการเคลื่อนไหว จะให้สถานะเอาต์พุต HIGH เพียงครั้งเดียว และมีการหน่วงเวลาตามค่า Time Delay และจึงคืนสถานะกลับมาเป็น LOW

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 8.4 วงจรสำหรับแลบที่ 8.4

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#define buzzerPin D1
#define pirPin D2
#define ledPin D3

int val = 0;

bool motionState = false;

void setup() {
    pinMode(buzzerPin, OUTPUT);
    pinMode(pirPin, INPUT);
    pinMode(ledPin, OUTPUT);
    Serial.begin(115200);
}

void loop() {
    val = digitalRead(pirPin);

    if (val == HIGH) {
        digitalWrite(ledPin, HIGH);
        alarm(500, 2000);

        delay(150);

        if (motionState == false) {
            Serial.println("Motion Detected");
            motionState = true;
        }
    } else {
        digitalWrite(ledPin, LOW);
        noTone(buzzerPin);

        delay(150);

        if (motionState == true) {
            Serial.println("No Motion");
            motionState = false;
        }
    }
}

void alarm (long duration, int freq) {
    tone(buzzerPin, freq);
    delay(duration);
    noTone(buzzerPin);
}
```

- ทดลองขยายบอร์ดโดยมีอ่อนเชื่อมต่อชิปเซอร์ สามารถปรับตัวต้านทานปรับค่าได้เพื่อปรับความไวของเซ็นเซอร์ได้

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

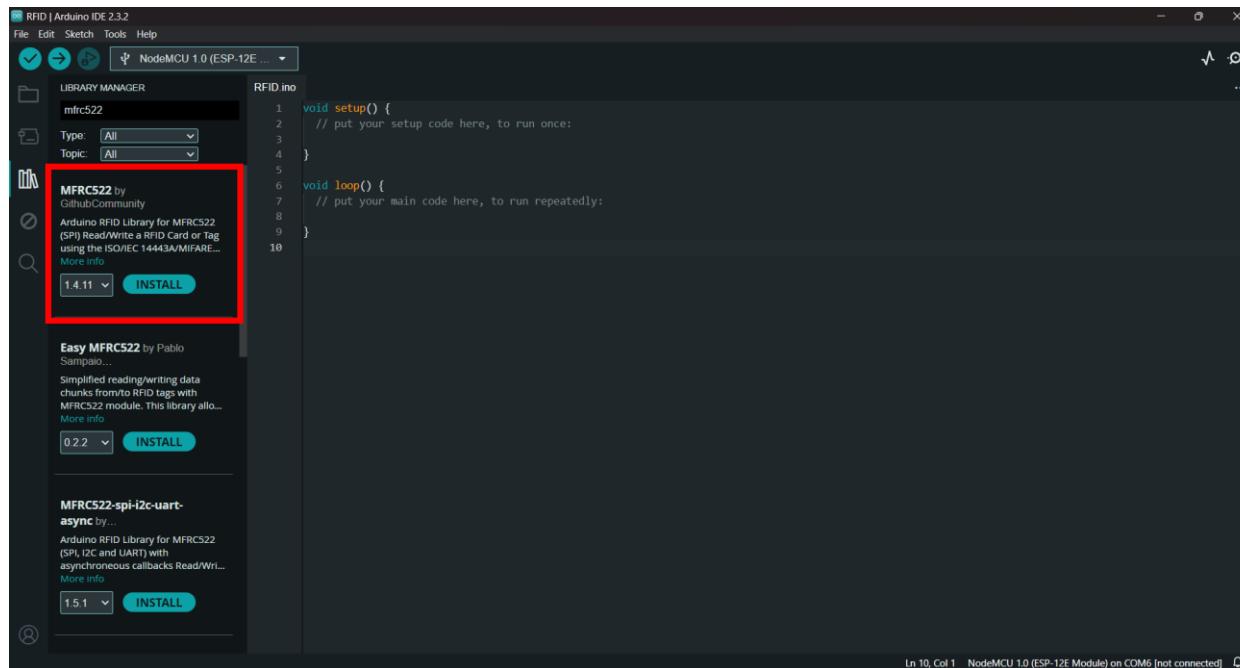
Lab 8.5: การอ่านค่าด้วย RFID Tag

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- โมดูล RFID-RC522 1 ตัว
- Buzzer 1 ตัว
- สาย Jumper
- Breadboard 1 อัน

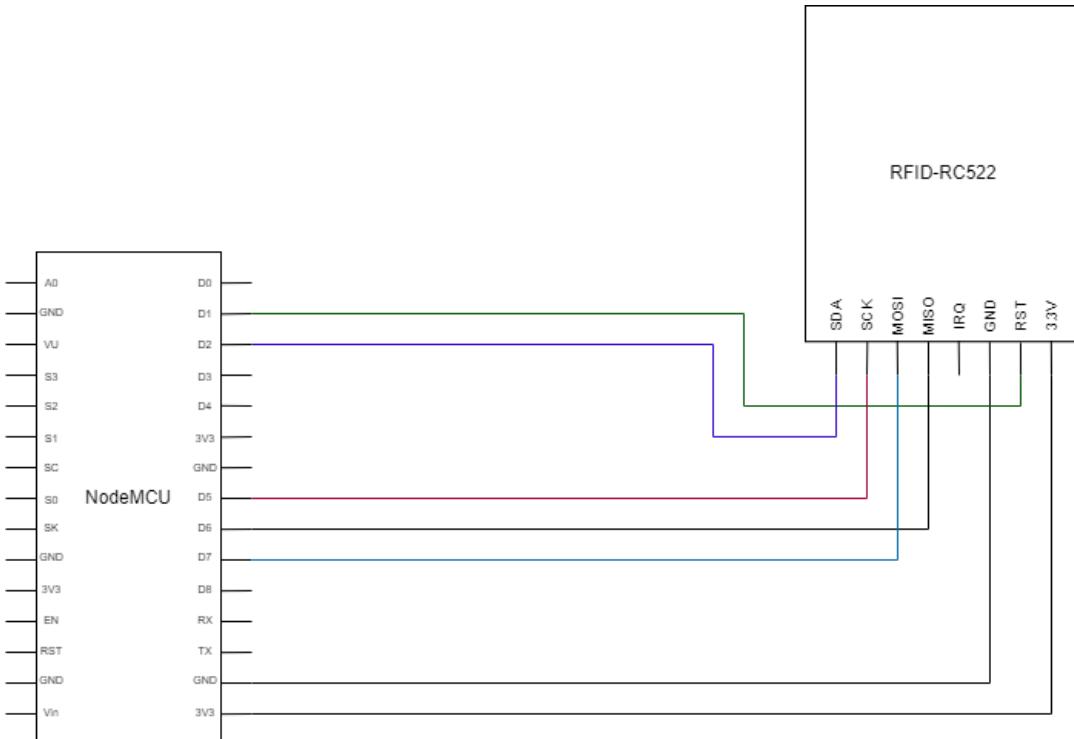
ขั้นตอนในการทำ

- ที่ Arduino IDE คลิกเมนู Tools -> Manage Library...
- ค้นหา mfrc522 และทำการติดตั้งไลบรารี MFRC522 by GitHub Community



รูปที่ 8.5 ไฟล์ราร์กีสำหรับโมดูล RFID

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 8.6 วงจรสำหรับแลบที่ 8.5

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#include <MFRC522.h>
#include <SPI.h>

#define RST_PIN D1
#define SS_PIN D2

MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {
    Serial.begin(115200);
    SPI.begin();
    mfrc522.PCD_Init();
    Serial.println("Start");
    delay(1000);
}

void loop() {
    if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial())
    {
        String rfid_in = rfid_read();
        Serial.print(">>> ");
        Serial.println(rfid_in);
        delay(1000);
    }
    delay(1);

    String rfid_read() {
        String content = "";
        for (byte i = 0; i < mfrc522.uid.size; i++) {
            content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
            content.concat(String(mfrc522.uid.uidByte[i], HEX));
        }
        content.toUpperCase();
        return content.substring(1);
    }
}
```

- เปิด Serial Monitor และนำการ์ดมาแตะที่โมดูล RFID เพื่อดูผลลัพธ์ที่เกิดขึ้น

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

บทที่ 9

การพัฒนาแอปพลิเคชันเพื่อเชื่อมต่อและควบคุมอุปกรณ์ผ่านอินเทอร์เน็ต

ในการพัฒนาระบบ IoT การสื่อสารและควบคุมอุปกรณ์ในระยะไกลนับเป็นองค์ประกอบที่สำคัญที่ต้องพัฒนา ซึ่งจะทำการควบคุมผ่านการสื่อสารรูปแบบต่าง ๆ โดยการใช้เครือข่ายอินเทอร์เน็ตที่ปัจจุบันใช้ในการควบคุมอุปกรณ์อย่างแพร่หลาย โดยอาศัย Cloud Server เป็นตัวกลางผ่านการพัฒนา Mobile Application บนสมาร์ทโฟน

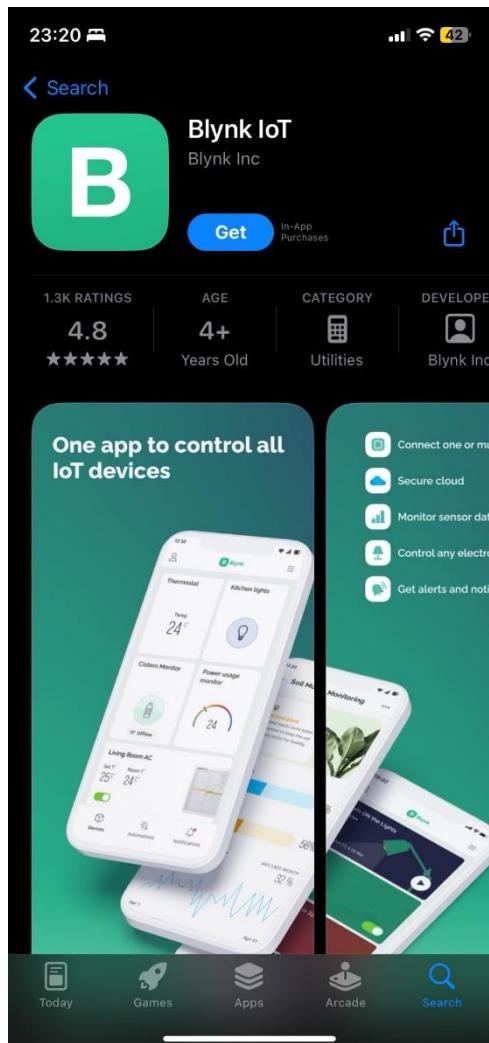
Blynk เป็นแพลตฟอร์มที่สามารถนำมาใช้ในการพัฒนาแอปพลิเคชันสำหรับการเชื่อมต่อและควบคุมอุปกรณ์ IoT ซึ่งแพลตฟอร์ม Blynk ประกอบไปด้วย 3 ส่วน คือ

1. Blynk Application: Mobile Application ที่รองรับทั้งระบบ Android และ iOS ใช้สำหรับเป็นแพงควบคุมอุปกรณ์ IoT ผ่านสมาร์ทโฟน
2. Blynk Server: Cloud Service ซึ่งเป็นตัวกลางในการสื่อสารระหว่างอุปกรณ์ IoT และ Mobile Application
3. Blynk Library: ไลบรารีสำหรับนำໄไปใช้ในการพัฒนาโค้ดเพื่อให้สามารถเชื่อมต่อระหว่างอุปกรณ์ IoT และ Mobile Application ได้

การเริ่มต้นการใช้งาน Blynk

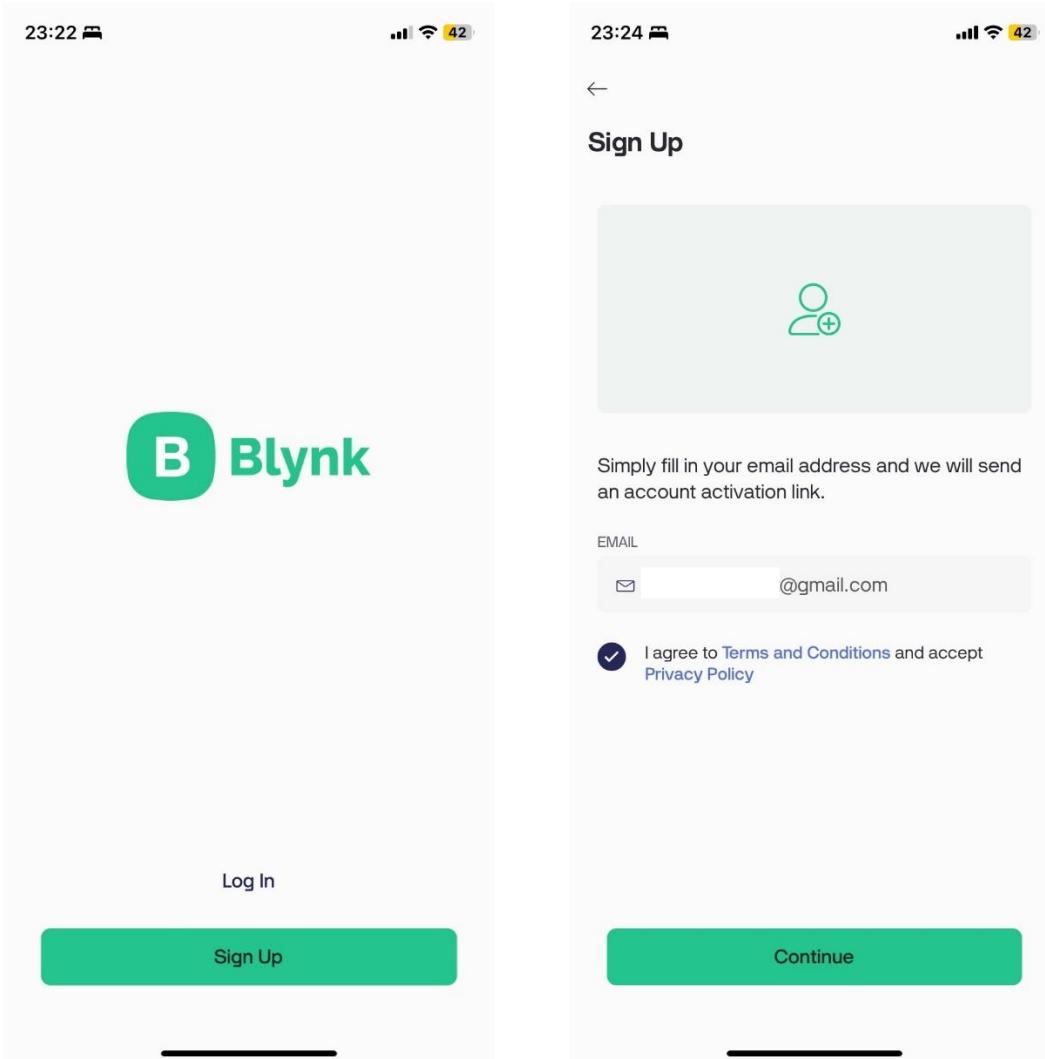
ในการเริ่มต้นการใช้งาน Blynk จะต้องดำเนินการตามขั้นตอนดังนี้

- ดาวน์โหลดและติดตั้ง Blynk App บนสมาร์ทโฟน โดยดาวน์โหลดผ่าน App Store (สำหรับ iOS) หรือ Play Store (สำหรับ Android)



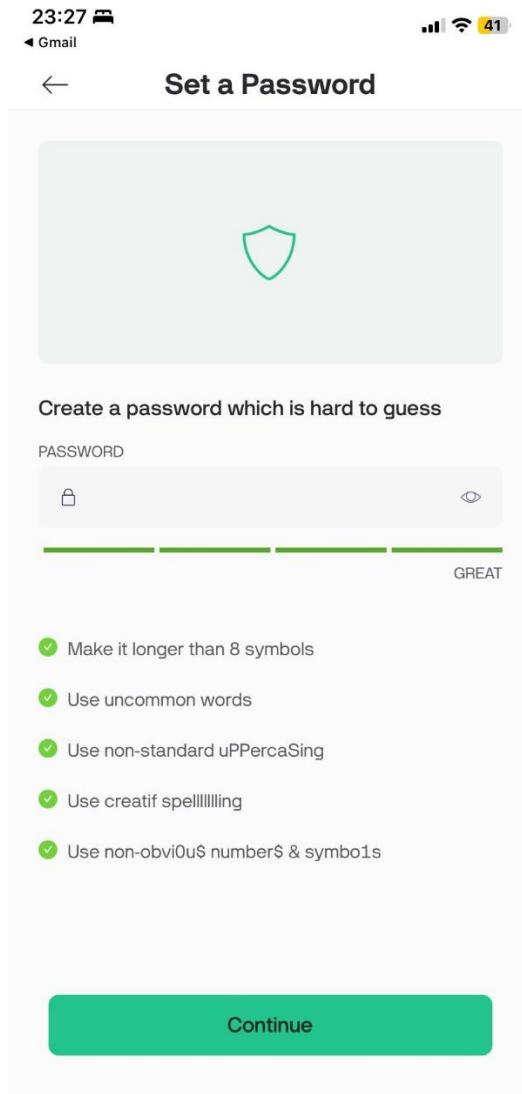
รูปที่ 9.1 แอปพลิเคชัน Blynk บน App Store

- ลงทะเบียน Account ผ่านอีเมล โดยกดที่ Sign Up กรอกอีเมล กดยอมรับนโยบายการทำงาน แล้วกด Continue



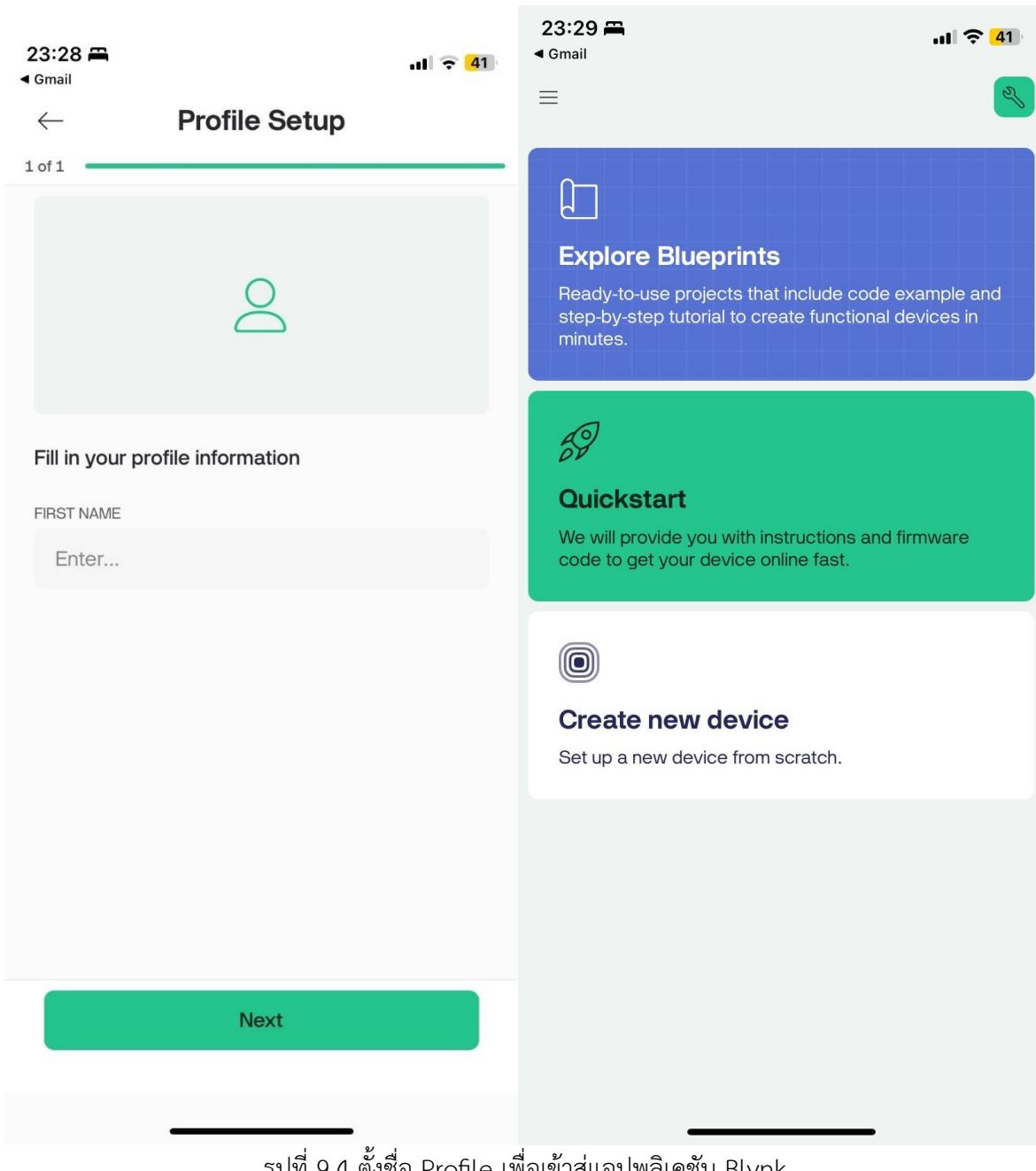
รูปที่ 9.2 ขั้นตอนการลงทะเบียนบนแอปพลิเคชัน Blynk

- เข้าไปยังอีเมลที่ใช้ในการลงทะเบียน จะได้รับอีเมลจาก Blynk ให้กด Create Password เพื่อสร้างรหัสผ่าน และทำการตั้งค่ารหัสผ่านให้ตรงกับเงื่อนไขที่แอปพลิเคชันกำหนด และกด Continue



รูปที่ 9.3 กำหนดรหัสผ่านบนแอปพลิเคชัน Blynk

- ตั้งค่าชื่อ Profile ตามที่แอปพลิเคชันกำหนด แล้วกด Next เป็นการเสร็จการติดตั้งในสมาร์ทโฟน



รูปที่ 9.4 ตั้งชื่อ Profile เพื่อเข้าสู่แอปพลิเคชัน Blynk

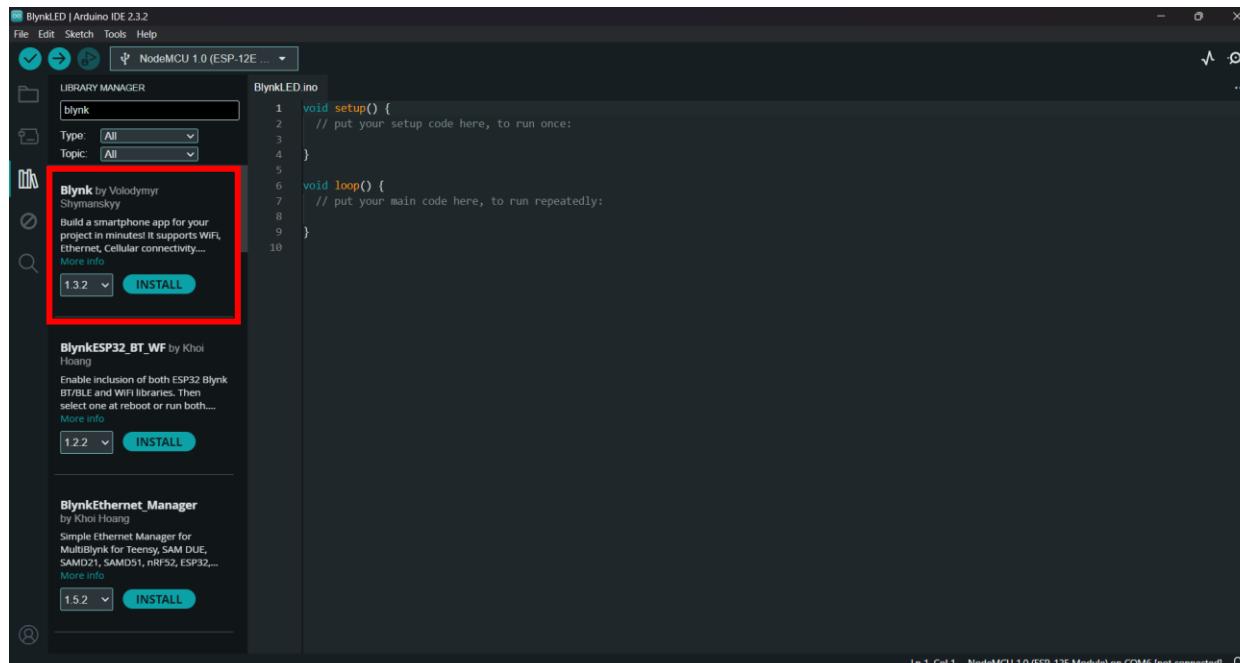
Lab 9.1: ระบบเปิด/ปิดหลอดไฟผ่าน Blynk

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- หลอด LED 1 อัน
- ตัวต้านทานขนาด 220 Ω 1 อัน
- สาย Jumper
- Breadboard 1 อัน
- แอปพลิเคชัน Blynk

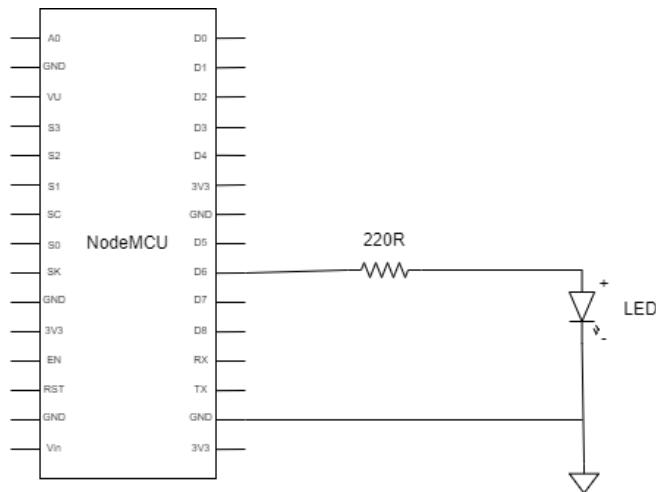
ขั้นตอนในการทำ

- ที่ Arduino IDE คลิกเมนู Tools -> Manage Library...
- ค้นหา blynk และทำการติดตั้งไลบรารี blynk by Volodymyr Shymanskyy



รูปที่ 9.5 ไลบรารีสำหรับ blynk

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 9.6 วงจรสำหรับแลบที่ 9.1

- พิมพ์คัดต่อไปนี้

```
#define BLYNK_TEMPLATE_ID          "YourTemplateID"
#define BLYNK_TEMPLATE_NAME         "Quickstart Device"
#define BLYNK_AUTH_TOKEN            "YourAuthToken"

#define BLYNK_PRINT Serial

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

#define ledPin D6

char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

BlynkTimer timer;

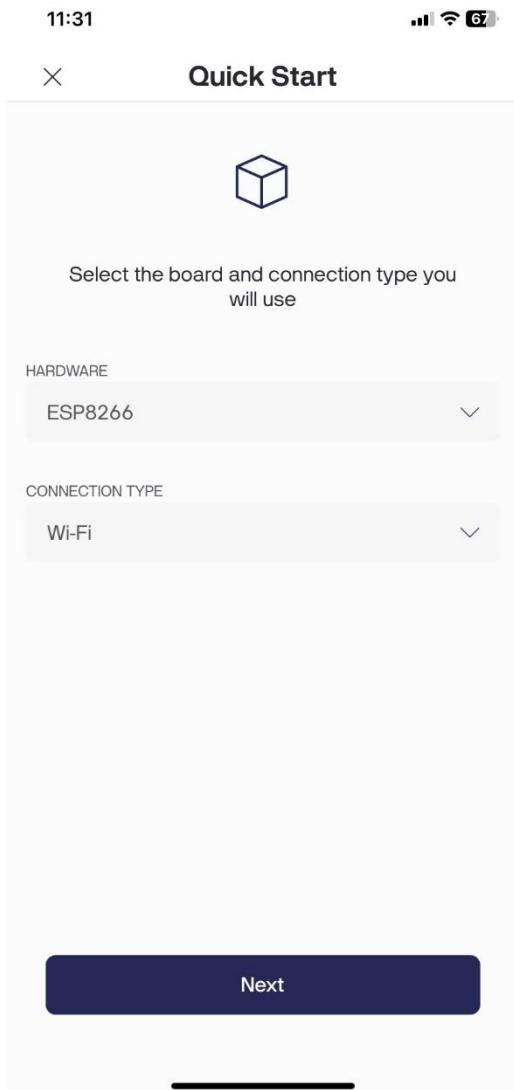
BLYNK_WRITE(V0)
{
    int pinValue = param.asInt();

    if (pinValue == 1) {
        digitalWrite(ledPin, HIGH);
        Serial.println("LED On");
    } else {
        digitalWrite(ledPin, LOW);
        Serial.println("LED Off");
    }
}

void setup()
{
    Serial.begin(115200);
    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);
}

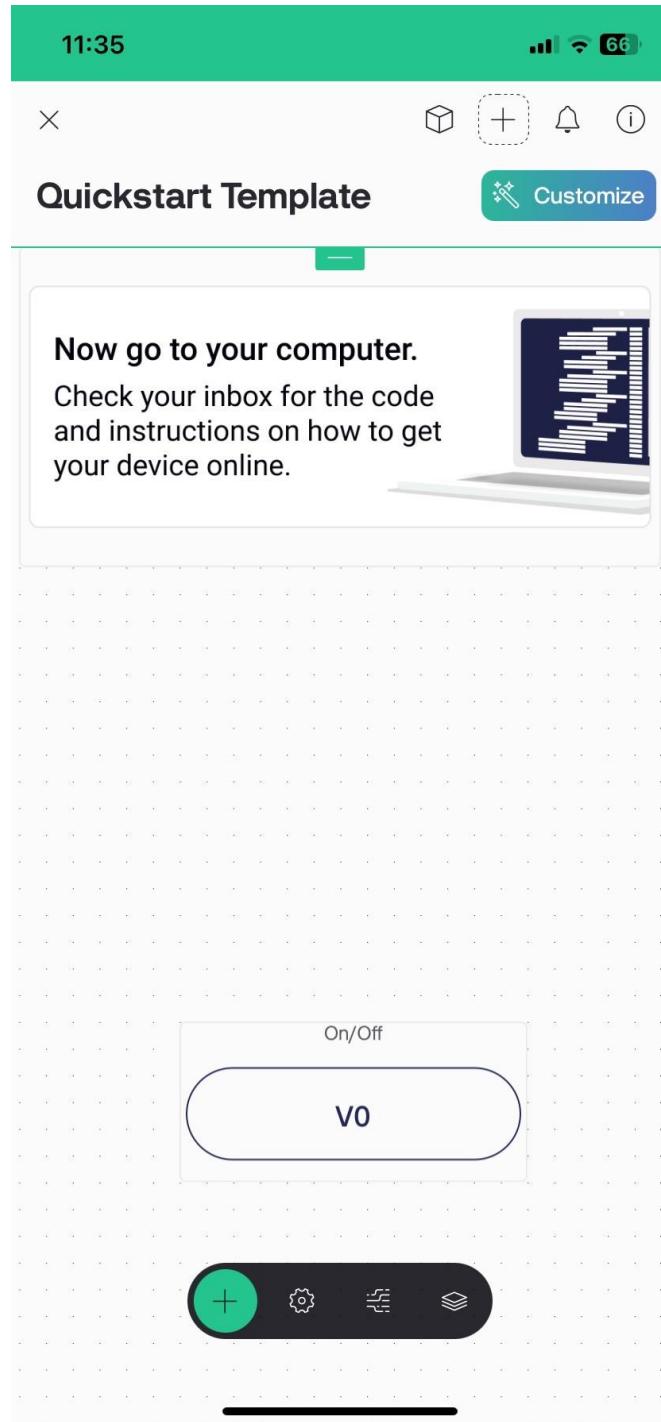
void loop()
{
    Blynk.run();
    timer.run();
}
```

- YourNetworkName ให้กรอกชื่อเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม แนะนำให้ใช้ Hotspot ของโทรศัพท์มือถือ
- YourPassword ให้กรอกรหัสผ่านของเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม
- YourAuthToken ให้นำ Token จากขั้นตอนถัดไปมากรอก
- YourTemplateID ให้นำ Template ID จากขั้นตอนถัดไปมากรอก
- 'ไปที่แอปพลิเคชัน Blynk และที่ Quick Start จะปรากฏหน้าตั้งค่า ให้เลือก Hardware เป็น ESP8266 และ Connection Type เป็น Wi-Fi จากนั้นกด Next'



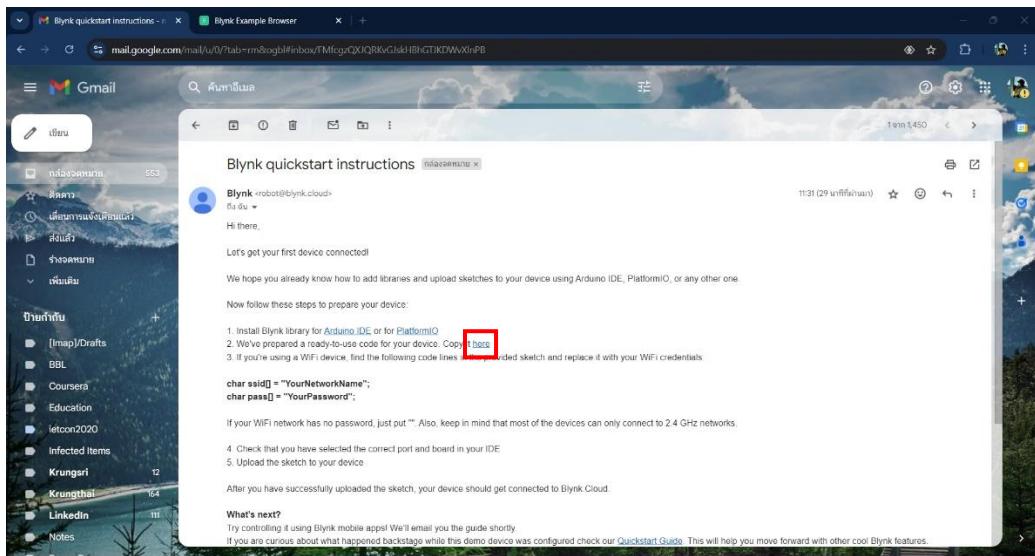
รูปที่ 9.7 การตั้งค่า Hardware บนแอปพลิเคชัน Blynk

- แอปพลิเคชันจะแสดงหน้าต่างบนมือถือ ซึ่งจะเป็นส่วนที่ใช้ในการควบคุมฮาร์ดแวร์ ซึ่งสามารถตั้งค่าและปรับแต่งหน้า UI ได้ โดยใน Quick Start จะมีสวิตซ์กดติดกดดับ (V0) กำหนดมาให้ 1 อัน



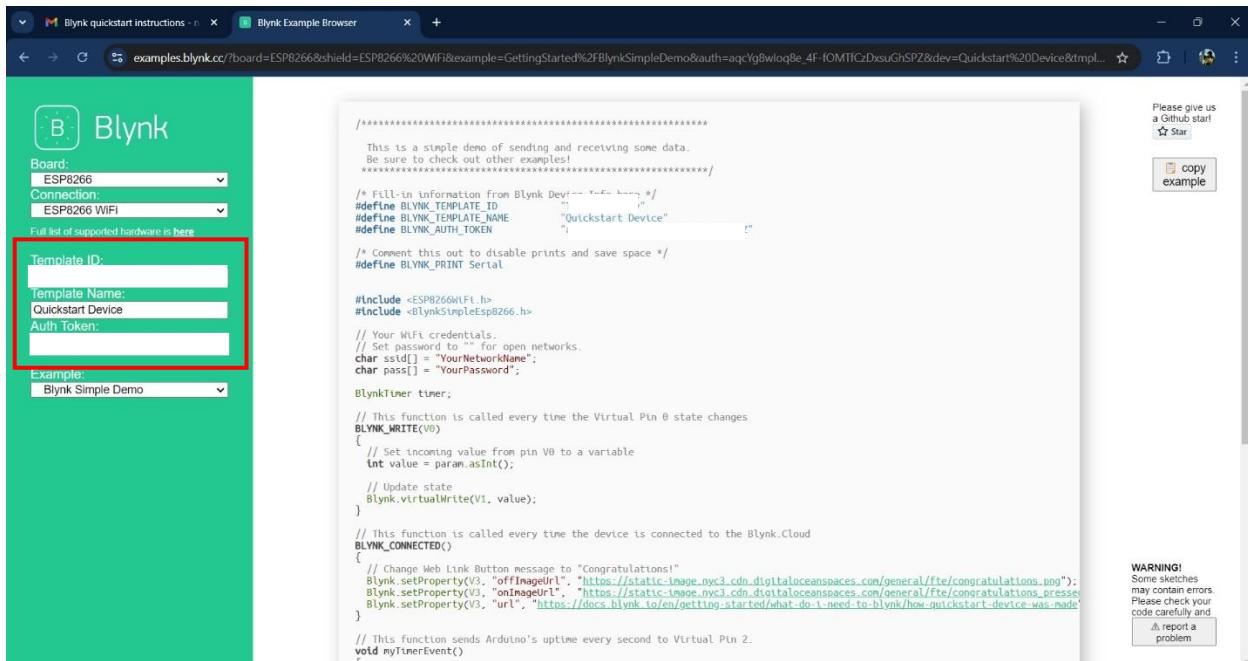
รูปที่ 9.8 หน้า UI บนแอปพลิเคชัน Blynk

- เมื่อสร้างแอปพลิเคชันจาก Quick Start เสร็จ จะมีอีเมลส่งไปยังอีเมลที่ใช้ในการสมัครแอปพลิเคชัน ให้ไปที่อีเมลแล้วคลิกลิงก์ที่คำว่า here



รูปที่ 9.9 อีเมลเพื่อขอ Auth Token และ Template ID

- ลิงก์จะพาไปสู่หน้า Template ของ Quick Start ให้คัดลอก Auth Token และ Template ID ไปใส่ในโค้ดที่อยู่ใน Arduino IDE



รูปที่ 9.10 หน้าตัวอย่างโค้ดสำหรับโปรแกรมลงบนบอร์ด ESP8266

- ทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด
- เปิดหน้า Serial Monitor
- ทดลองกดเปิดปิดสวิตซ์บนโทรศัพท์แล้วสังเกตผลลัพธ์ที่เกิดขึ้น

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

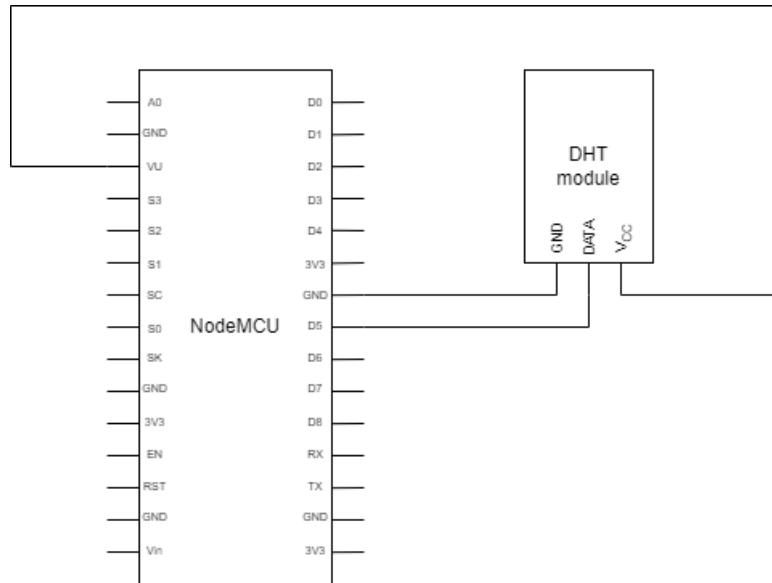
Lab 9.2: การสร้าง Dashboard ในการรายงานอุณหภูมิและความชื้นบนแอปพลิเคชัน Blynk

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- โมดูล DHT 1 ตัว
- สาย Jumper
- Breadboard 1 อัน
- แอปพลิเคชัน Blynk

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 9.11 วงจรสำหรับแลบที่ 9.2

- พิมพ์โค้ดต่อไปนี้

```
#define BLYNK_TEMPLATE_ID          "YourTemplateID"
#define BLYNK_TEMPLATE_NAME        "Quickstart Device"
#define BLYNK_AUTH_TOKEN           "YourAuthToken"

#define BLYNK_PRINT Serial

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

char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

#define DHTPIN D5
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
    Serial.begin(115200);
    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
    dht.begin();
}

void loop() {
    Blynk.run();
    float t = dht.readTemperature();
    float h = dht.readHumidity();

    if (isnan(t) || isnan(h)) {
        Serial.println("DHT sensor failure.");
        return;
    }

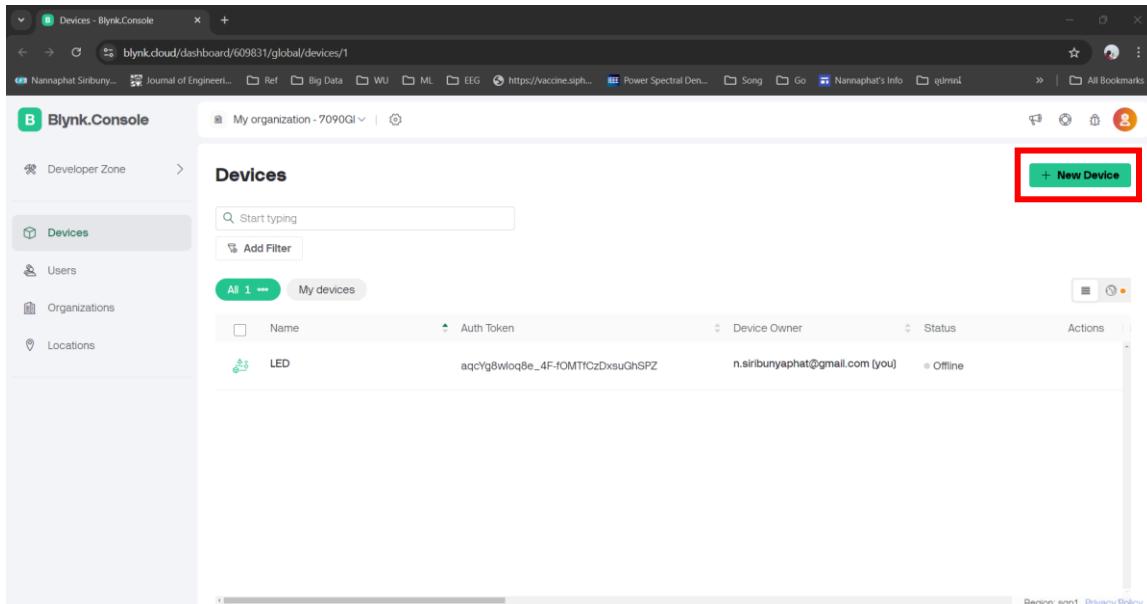
    Blynk.virtualWrite(V0, t);
    Blynk.virtualWrite(V1, h);

    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.print("°C | ");
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.println("%");

    delay(2000);
}
```

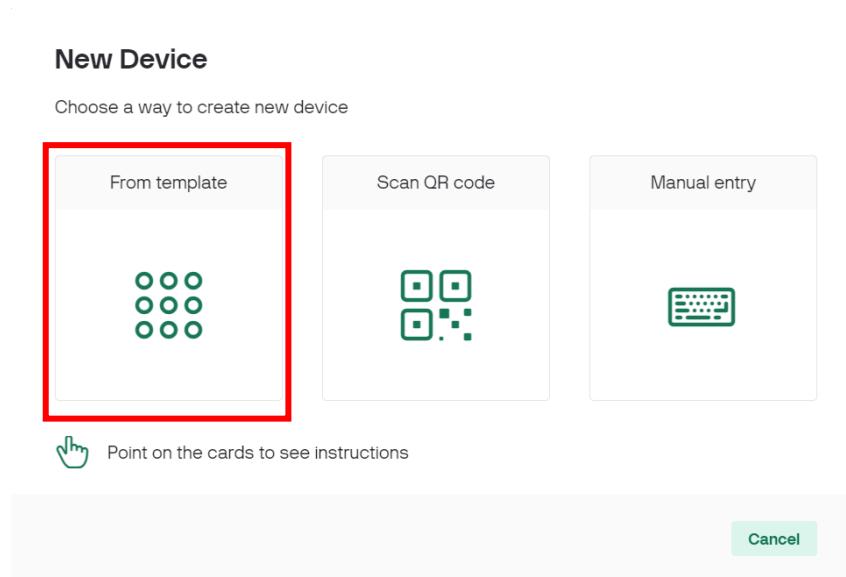
- YourNetworkName ให้กรอกชื่อเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม แนะนำให้ใช้ Hotspot ของโทรศัพท์มือถือ
- YourPassword ให้กรอกรหัสผ่านของเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม
- YourAuthToken ให้นำ Token จากขั้นตอนถัดไปมากรอก
- YourTemplateID ให้นำ Template ID จากขั้นตอนถัดไปมากรอก

- เข้าไปที่เว็บไซต์ <https://blynk.io/> และทำการล็อกอินด้วยอีเมลที่สมัครเอาไว้ในตอนแรก
- เมื่อล็อกอินแล้วจะเข้าสู่หน้า Blynk.Console ที่แท็บ Devices ให้กดปุ่ม New Device เพื่อสร้างอุปกรณ์ใหม่



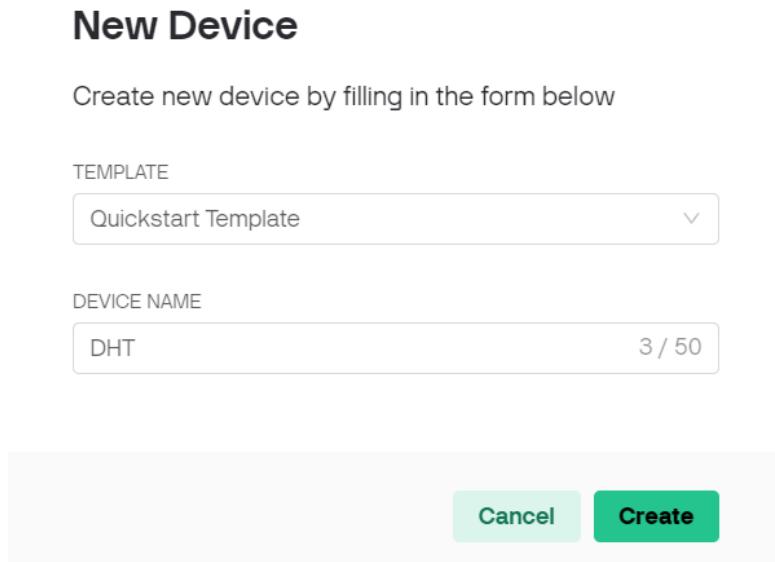
รูปที่ 9.11 หน้า Website Blynk.Console

- เมื่อหน้า New Device ปรากฏขึ้น ให้เลือก From Template



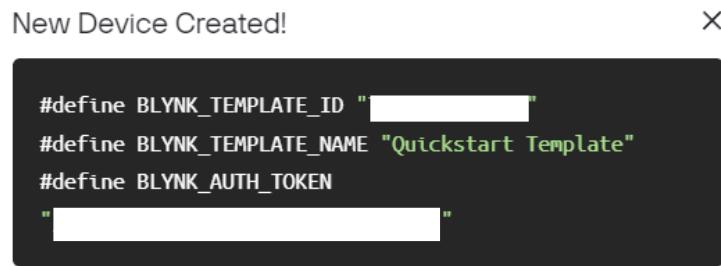
รูปที่ 9.12 เลือก Template สำหรับสร้างอุปกรณ์ใหม่

- เลือก Template เป็น Quickstart Template และตั้งชื่อ Device Name เป็น DHT จากนั้นกดปุ่ม Create



รูปที่ 9.13 กรอกค่าสำหรับอุปกรณ์ใหม่

- เมื่อสร้างอุปกรณ์ใหม่สำเร็จ มุมขวาจะมี Template ID และ Auth Token ปรากฏขึ้นมา ให้คัดลอกแล้วนำไปใส่ในโค้ดในขั้นตอนก่อนหน้า

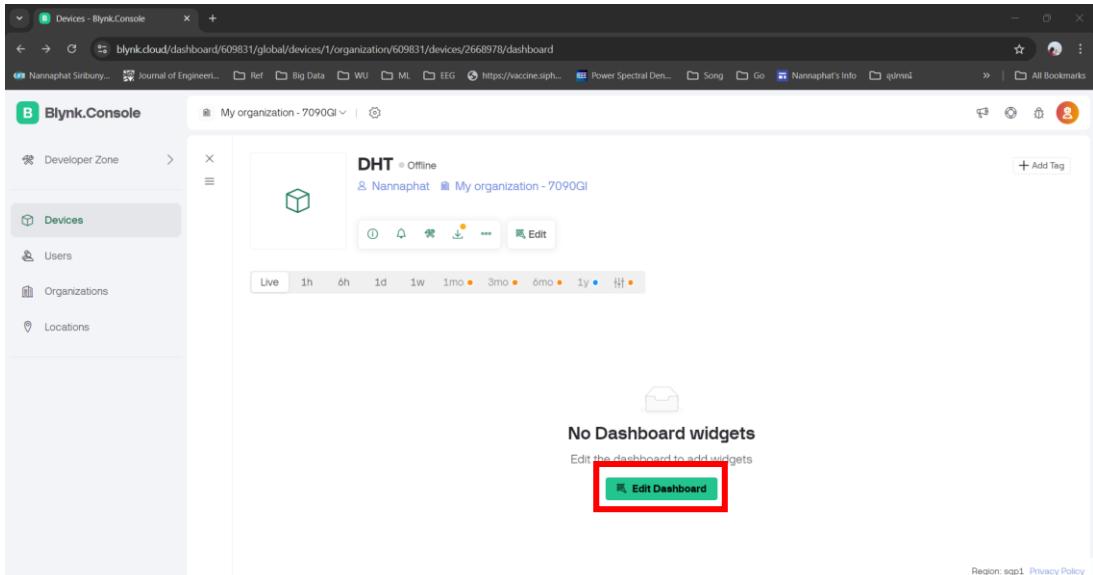


Template ID, Template Name, and AuthToken should be declared at the very top of the firmware code.

[Documentation](#) [Copy to clipboard](#)

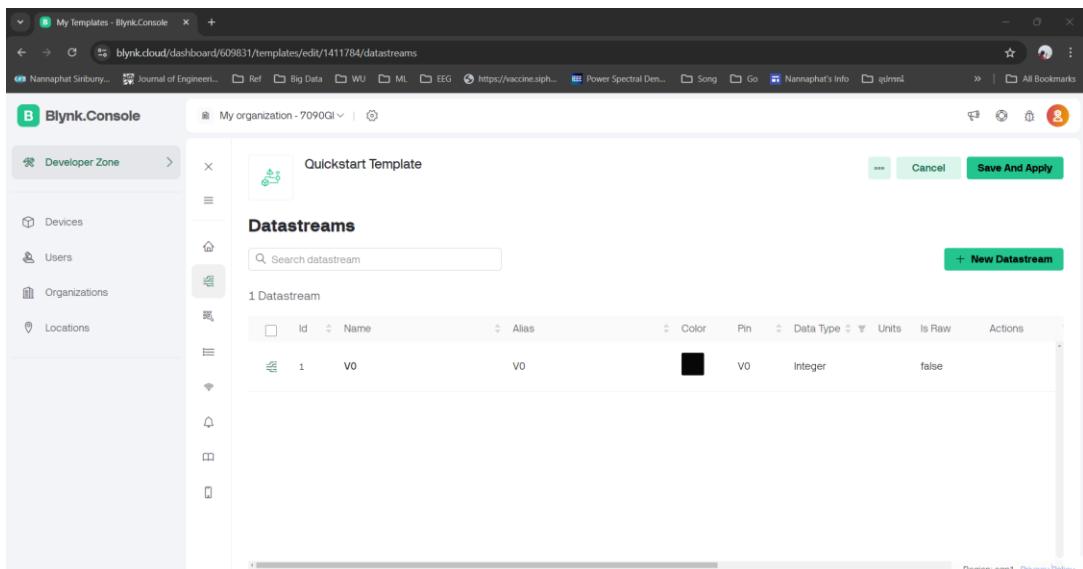
รูปที่ 9.14 Template ID และ Auth Token สำหรับอุปกรณ์ใหม่

- กดปุ่ม Edit Dashboard เพื่อทำการแก้ไข



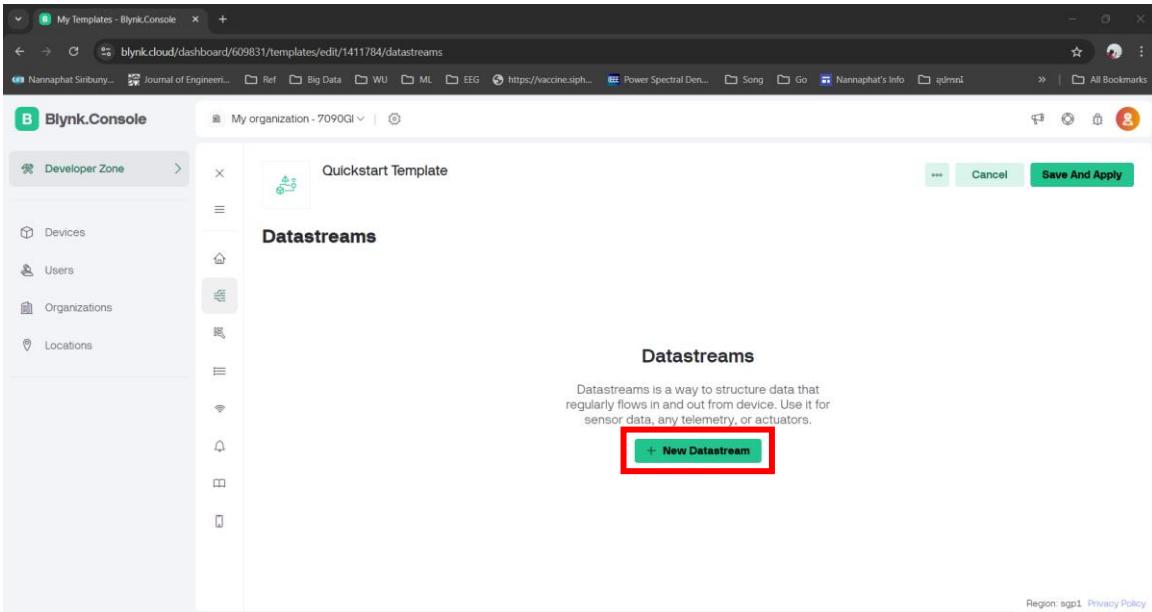
รูปที่ 9.14 หน้า Dashboard ของอุปกรณ์ใหม่

- ที่หน้า Web Dashboard ให้คลิกที่แท็บ Datastreams



รูปที่ 9.15 หน้า Datastreams

- เนื่องจาก Blynk ในเวอร์ชันฟรีจะสามารถสร้าง Datastream ได้สูงสุด 5 Datastream เท่านั้น ให้เลือกมาใส่ไปที่ Action แล้วทำการลบ Datastream ที่มีออก จากนั้นกดปุ่ม New Datastreams และเลือก Virtual Pin



รูปที่ 9.15 หน้า Datastreams ที่ลับ Datastream ออกแล้ว

- ทำการตั้งค่า Datastream ดังรูปที่ 9.16 จากนั้นกด Create

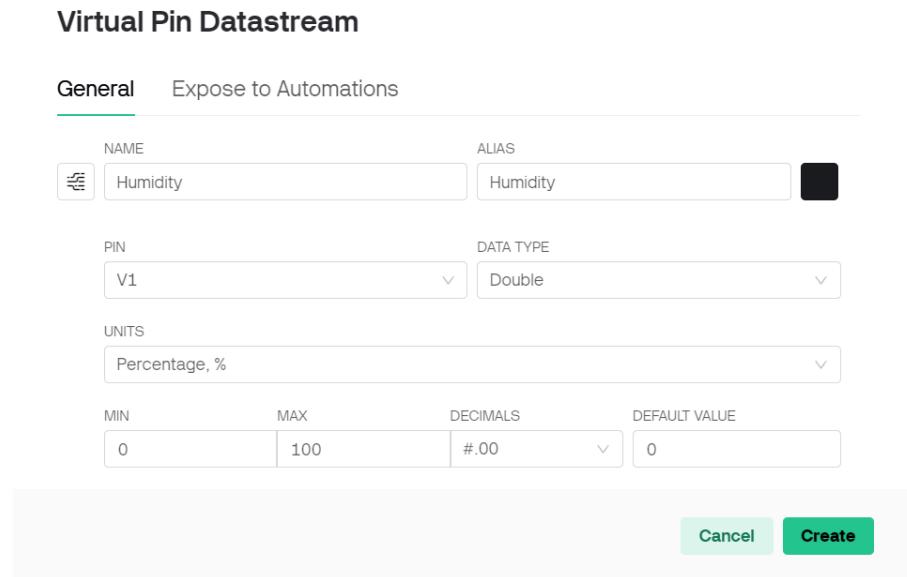
Virtual Pin Datastream

[General](#) [Expose to Automations](#)

NAME	ALIAS		
Temperature	Temperature		
PIN	DATA TYPE		
V0	Double		
UNITS			
Celsius, °C			
MIN	MAX	DECIMALS	DEFAULT VALUE
0	100	#.00	0
<input checked="" type="checkbox"/> Enable history data			
Cancel Create			

รูปที่ 9.16 การตั้งค่า Datastream สำหรับค่าอุณหภูมิ

- คลิกที่ New Datastream อีกครั้ง และเลือก Virtual Pin จากนั้นทำการตั้งค่า Datastream ดังรูปที่ 9.17 จากนั้นกด Create



รูปที่ 9.17 การตั้งค่า Datastream สำหรับค่าความชื้น

- จะได้ Datastream ใหม่ 2 Datastreams

My Templates - Blynk.Console

blynk.cloud/dashboard/609831/templates/edit/1411784/datastreams

B Blynk.Console My organization - 7090Qi | ⚙️

Developer Zone

Devices

Users

Organizations

Locations

Quickstart Template

Datastreams

Search datastream

+ New Datastream

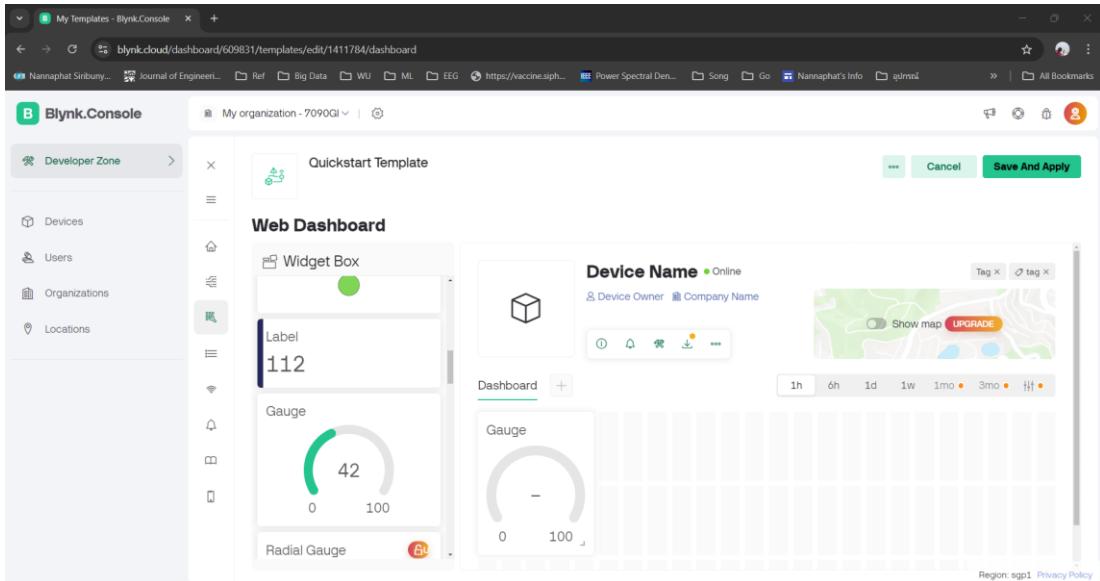
	ID	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Actions
1	Temperature	Temperature	Temperature	Black	V0	Double	*C	false	
2	Humidity	Humidity	Humidity	Black	V1	Double	%	false	

Collapse

Region: sgp1 Privacy Policy

รูปที่ 9.18 Datastreams สำหรับแลบที่ 9.2

- กลับมาที่หน้า Web Dashboard ที่ Widget Box ลาก Gauge มาใส่ที่หน้า Dashboard



รูปที่ 9.19 สร้าง Dashboard โดยใช้ Gauge

- วางแผนเครื่องเซอร์เบน Widget Gauge และคลิกที่ปุ่มพื้นเพื่อแก้ไข Widget จะปรากฏหน้า Gauge Settings ตั้งค่าดังที่แสดงในรูปที่ 9.20 และกด Save

Gauge Settings

TITLE (OPTIONAL)

Datasream

Delete

Override Datasream's Min/Max fields

MIN	MAX
0	100

Change color based on value

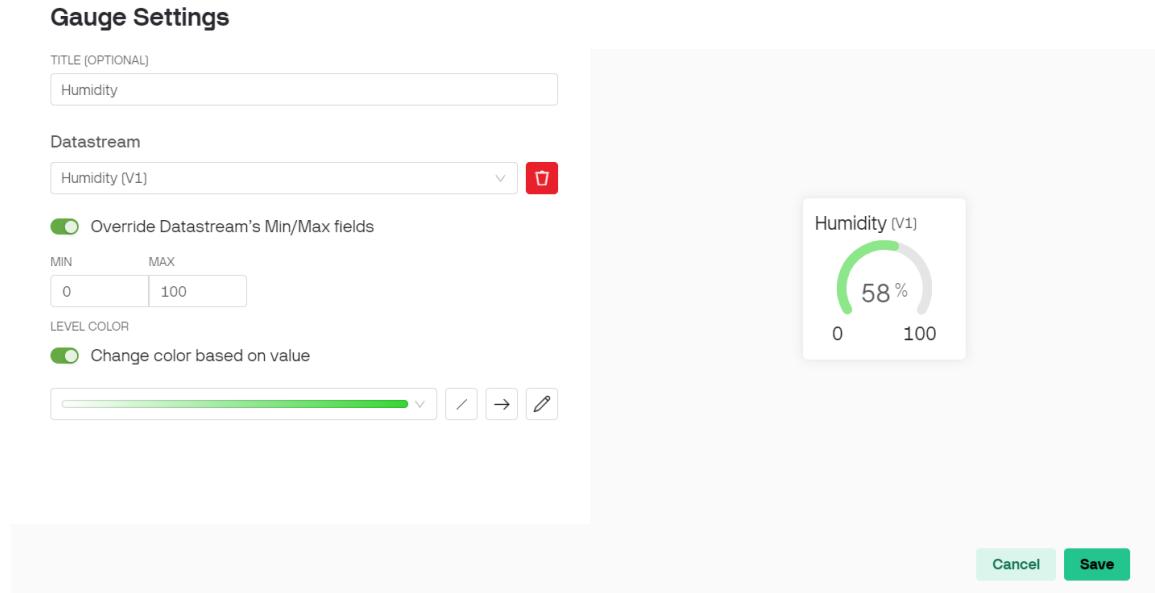
r⁺
→
Edit

Temperatu... (V0)

Cancel
Save

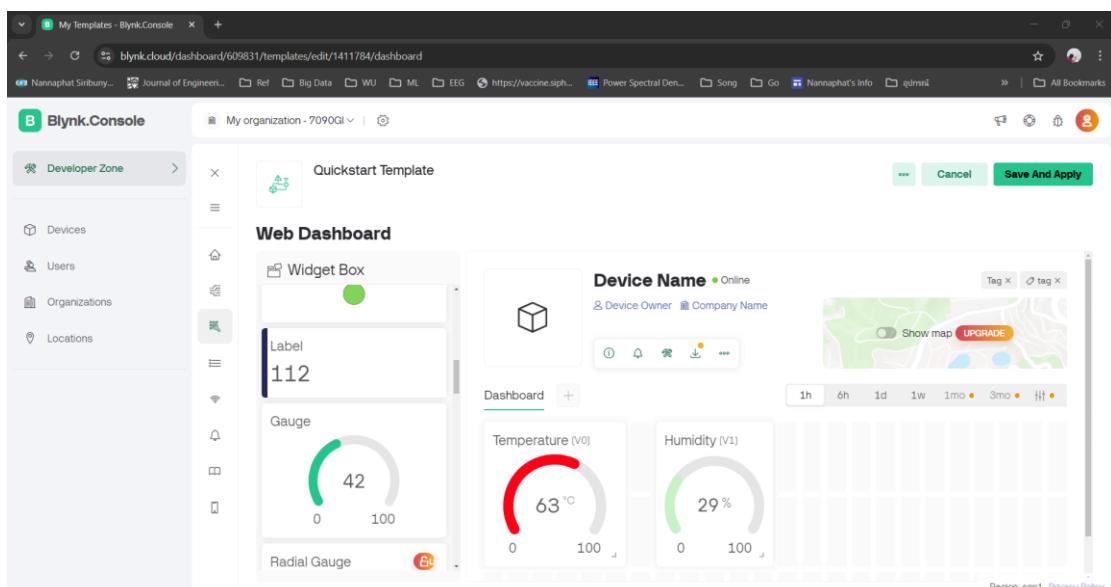
รูปที่ 9.20 การตั้งค่า Gauge สำหรับค่าอุณหภูมิ

- ทำตามโดยการลาก Gauge มาที่ Dashboard และทำการตั้งค่าสำหรับค่าความชื้น และกด Save



รูปที่ 9.21 การตั้งค่า Gauge สำหรับค่าความชื้น

- คลิก Save And Apply เพื่อบันทึกแอปพลิเคชัน



รูปที่ 9.22 Dashboard สำหรับแอปพลิเคชันตรวจสอบอุณหภูมิและความชื้น

- ทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

- เปิดหน้า Serial Monitor
- ตรวจสอบผลลัพธ์บนหน้าแอปพลิเคชัน Blynk ในโทรศัพท์และ Serial Monitor

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

บทที่ 10

การเชื่อมต่อระบบ IoT กับ Firebase

Firebase เป็นหนึ่งในบริการของ Google ที่ให้บริการฐานข้อมูลออนไลน์ในรูปแบบของ Real Time Database สำหรับ Application และ Web Application ซึ่งระบบ IoT สามารถนำ Firebase มาเป็นตัวกลางในการเชื่อมต่ออุปกรณ์เข้าด้วยกัน ซึ่ง Firebase จะเป็นฐานข้อมูลแบบ NoSQL โดยมีการเก็บข้อมูลแบบ JSON ทำให้มีความยืดหยุ่นในการใช้งาน

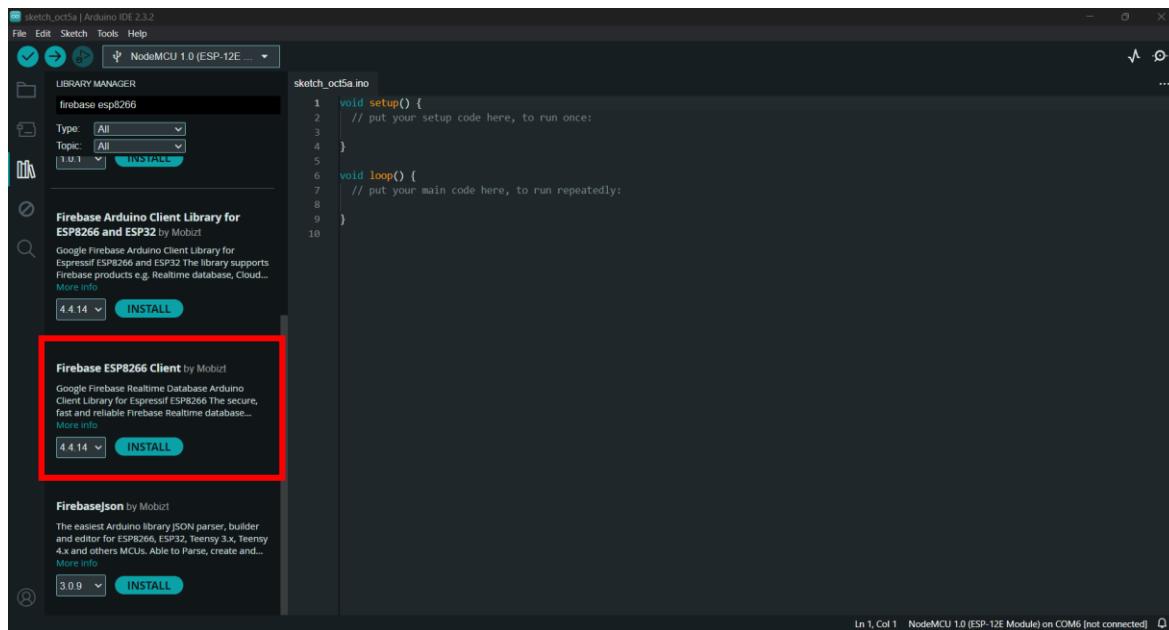
Lab 10.1: การเชื่อมต่อ NodeMCU กับ Firebase

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- Firebase

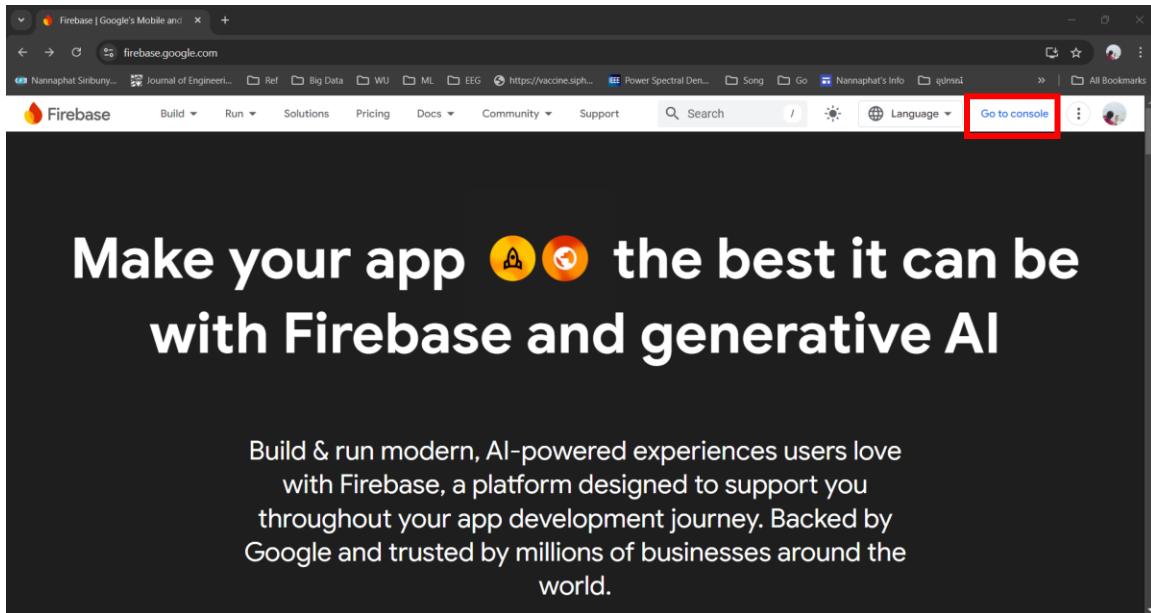
ขั้นตอนในการทำ

- ที่ Arduino IDE คลิกเมนู Tools -> Manage Library...
- ค้นหา firebase esp8266 แล้วทำการติดตั้งไลบรารี Firebase ESP8266 Client



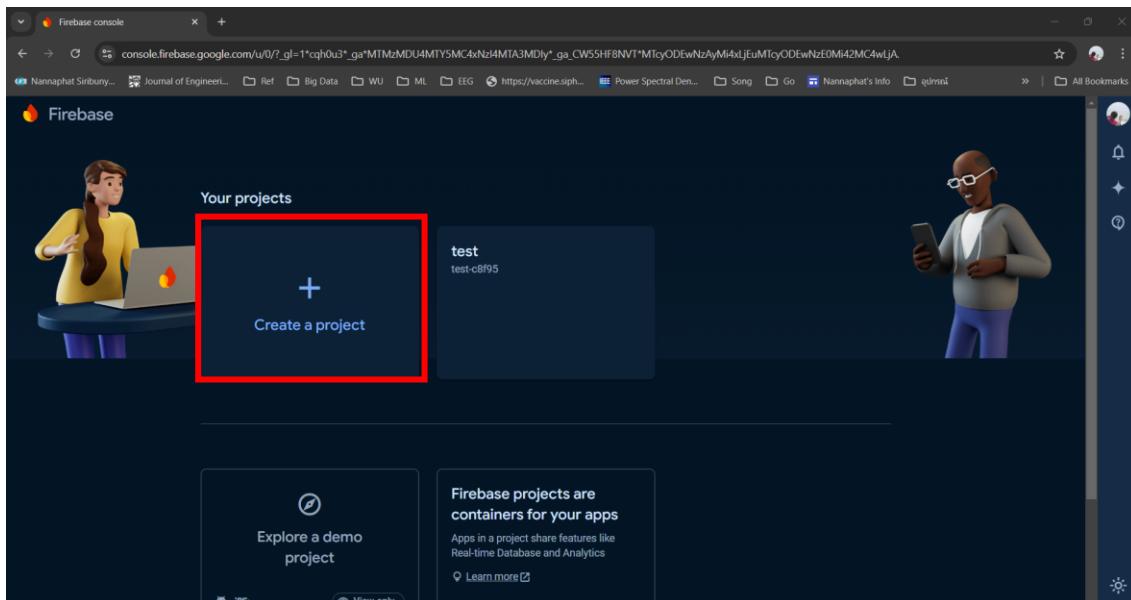
รูปที่ 10.1 ไลบรารี Firebase ESP8266 Client

- ไปที่ <https://firebase.google.com/> ทำการ sign in ด้วยแอคเคนท์ Google แล้วคลิกที่ Go to console



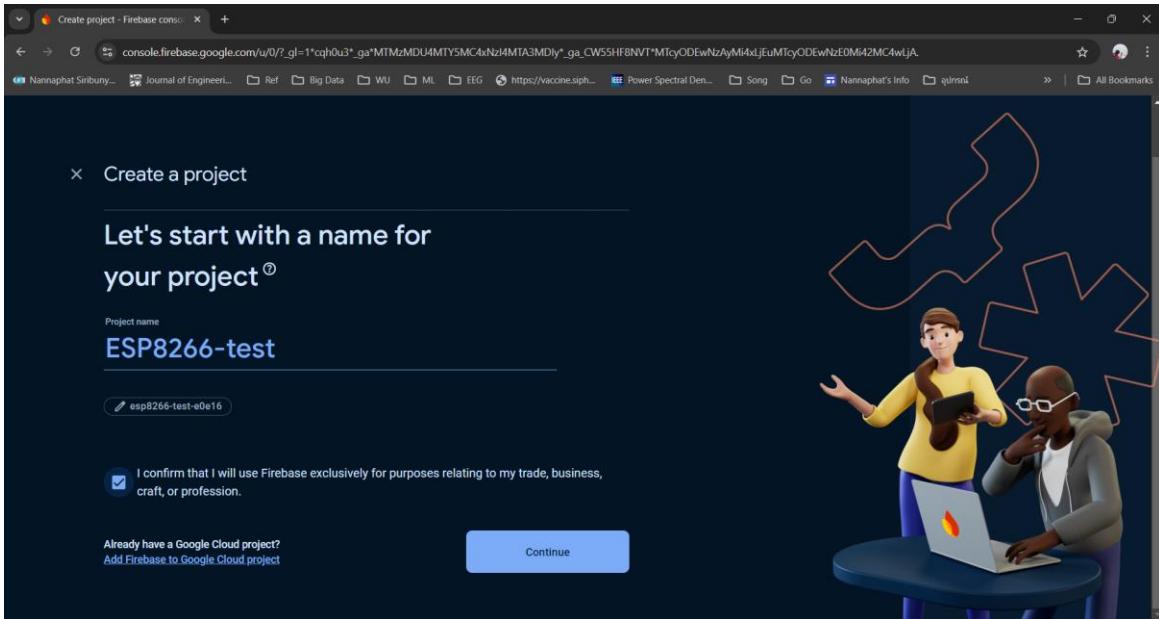
รูปที่ 10.2 หน้าเว็บ Firebase

- Click ที่ Create a project เพื่อสร้างโปรเจกต์ใหม่



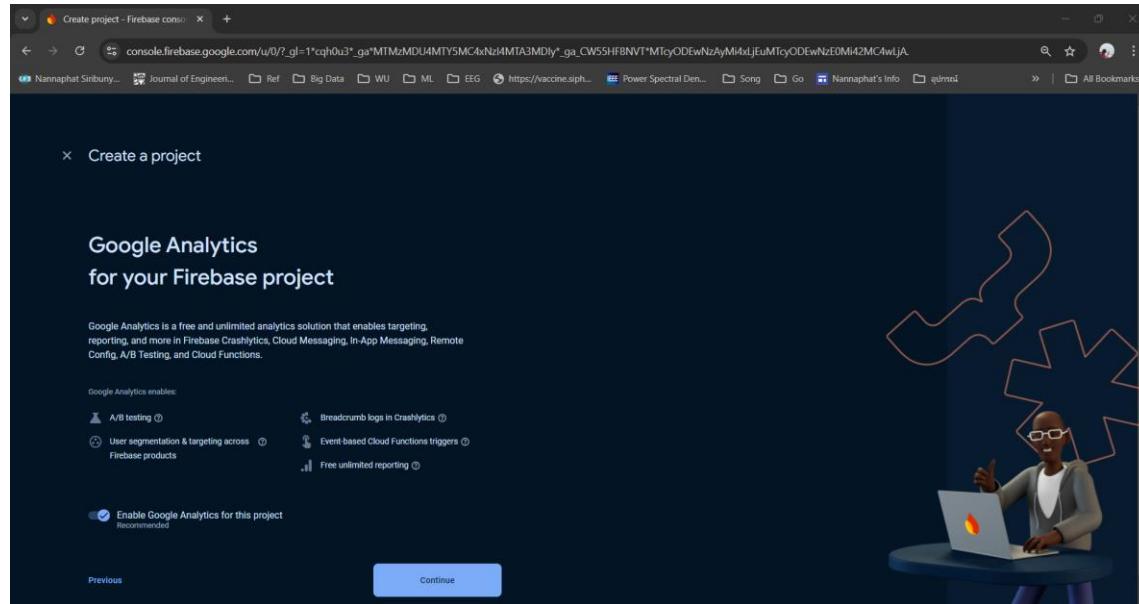
รูปที่ 10.3 หน้า console ของ Firebase

- ตั้งชื่อโปรเจกต์ เลือกยอมรับการใช้งานของ Firebase และกด Continue



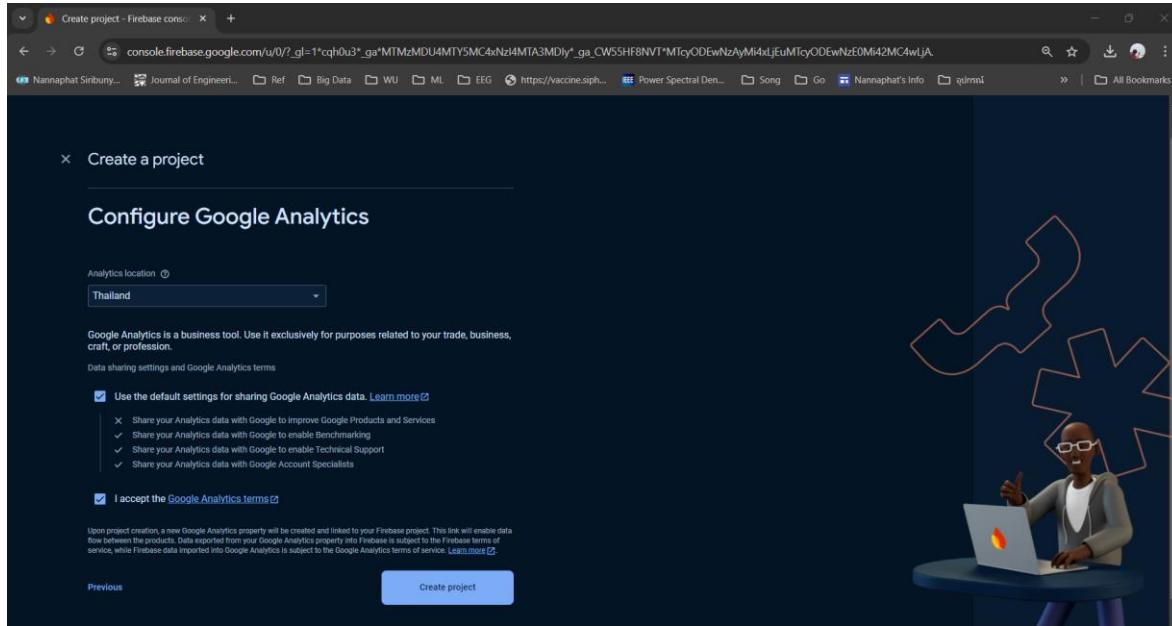
รูปที่ 10.4 การสร้างโปรเจกต์ใน Firebase

- ในหน้า Google Analytics ให้ Enable Google Analytics for this project และกด Continue



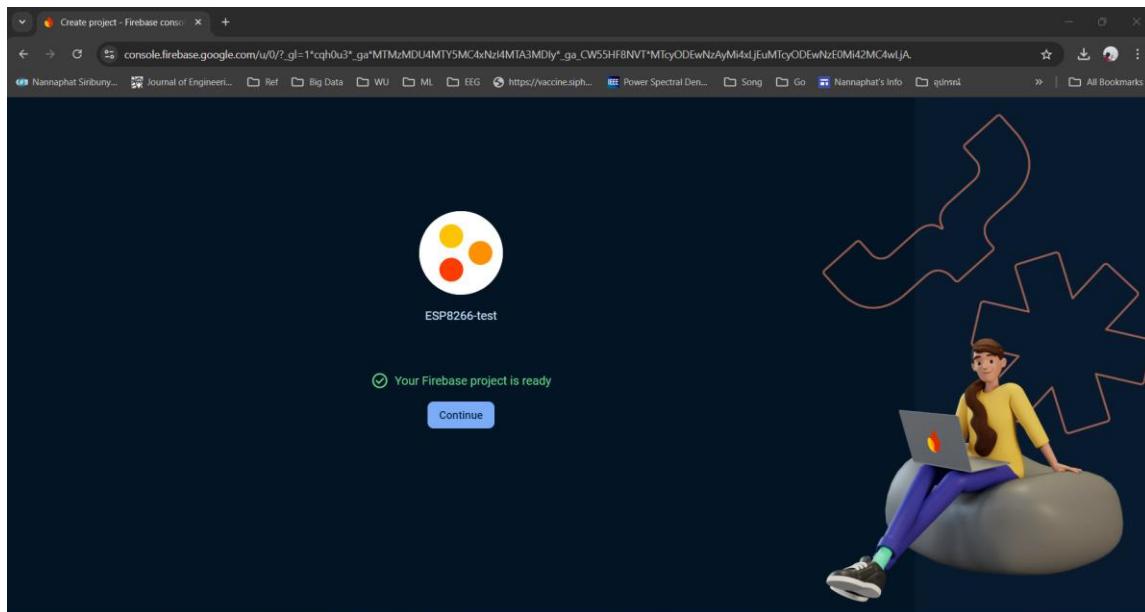
รูปที่ 10.5 ตั้งค่า Google Analytics ใน Firebase

- ตั้งค่า Analytics location เป็น Thailand เลือกยอมรับเงื่อนไขของ Google Analytics แล้วกด Create project



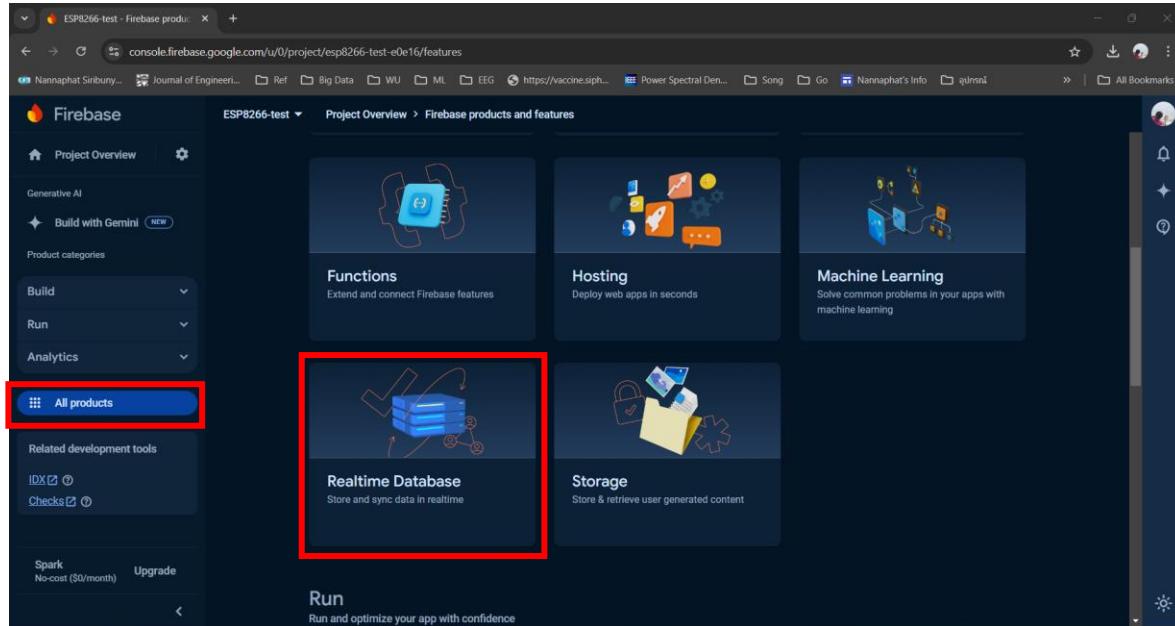
รูปที่ 10.6 ตั้งค่า Google Analytics Location ใน Firebase

- เมื่อระบบสร้างโปรเจกต์เสร็จให้กด Continue



รูปที่ 10.7 ระบบสร้างโปรเจกต์เสร็จสิ้น

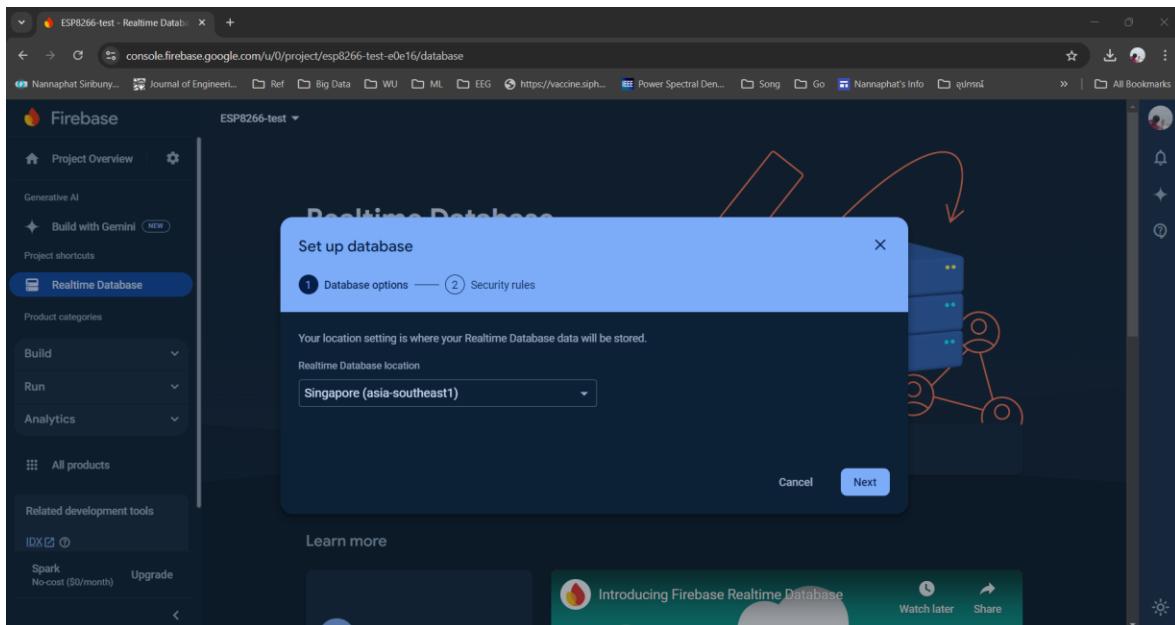
- ในหน้าโปรเจกต์ เลือกแท็บ All products ที่แถบด้านซ้ายมือ และเลือก Realtime Database และกด Create Database



รูปที่ 10.8 สร้าง Realtime Database

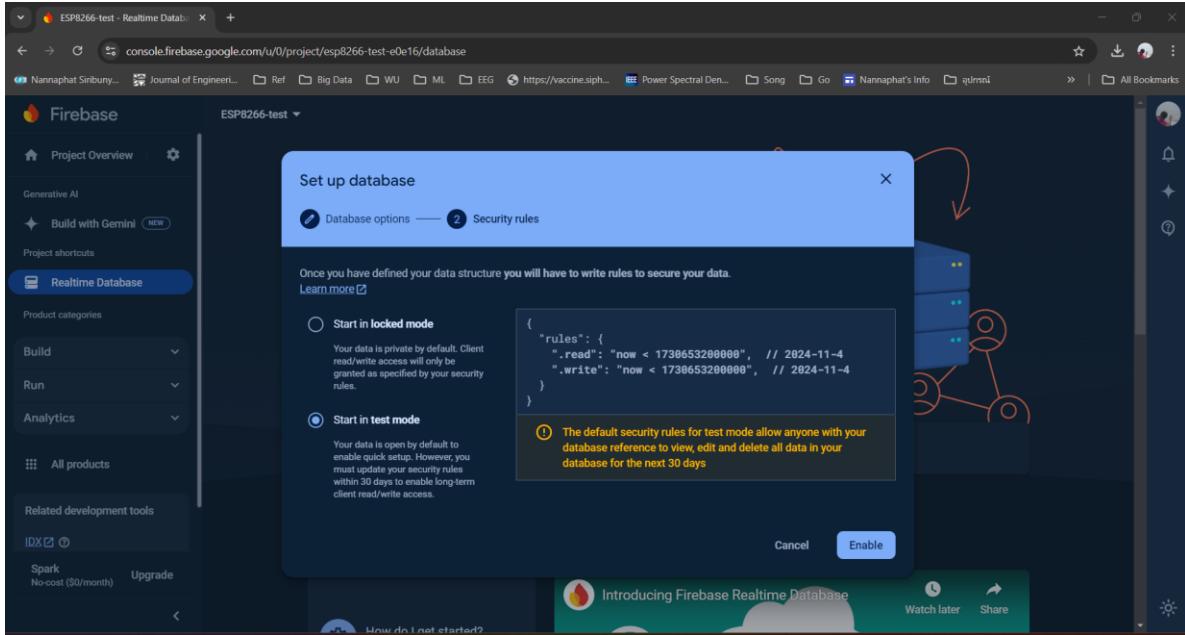
11

- ตั้งค่า Location เป็น Singapore (asia-southeast1) และกด next



รูปที่ 10.9 เลือกภูมิภาคของ Database

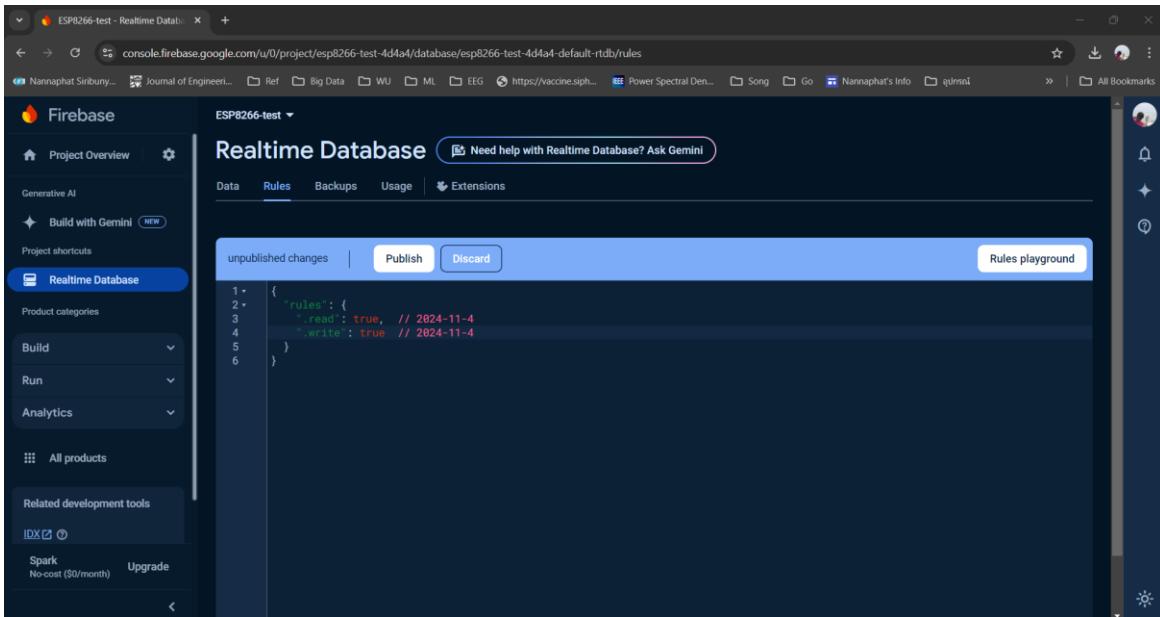
- เลือก Start in test mode และกด Enable



รูปที่ 10.10 ตั้งค่า Security rules ของ Database

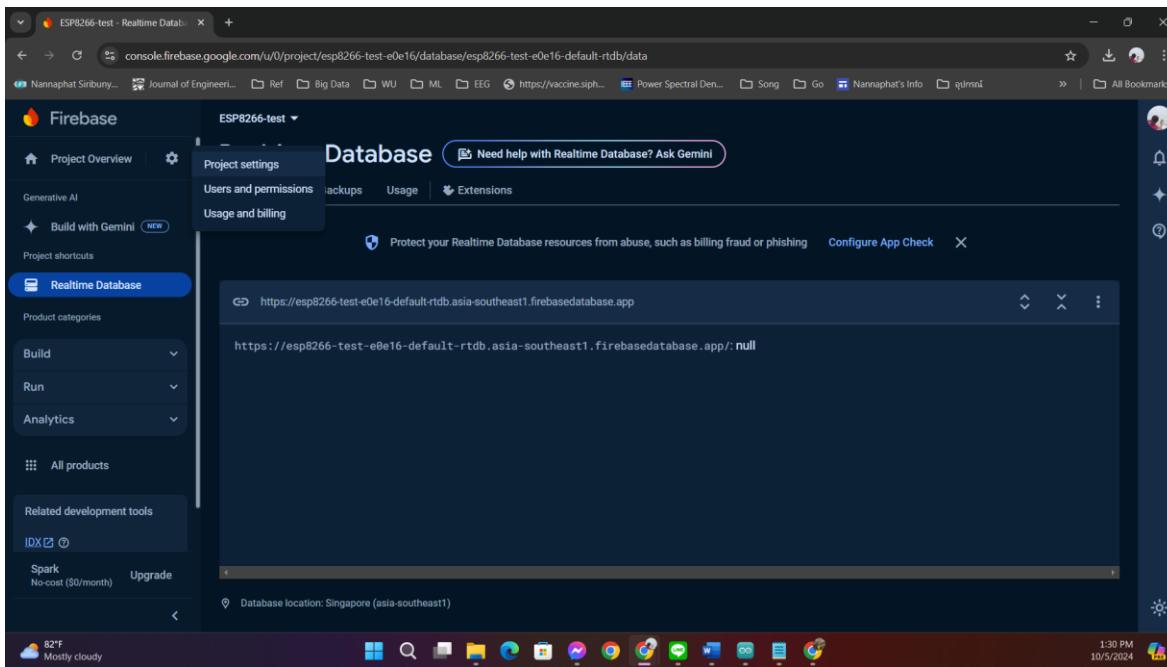
- เมื่อสร้าง Database เสร็จ คลิกที่แท็บ Rules และให้เปลี่ยน rules เป็น True ทั้ง read และ write ดัง

รูปที่ 10.11 และกด Publish



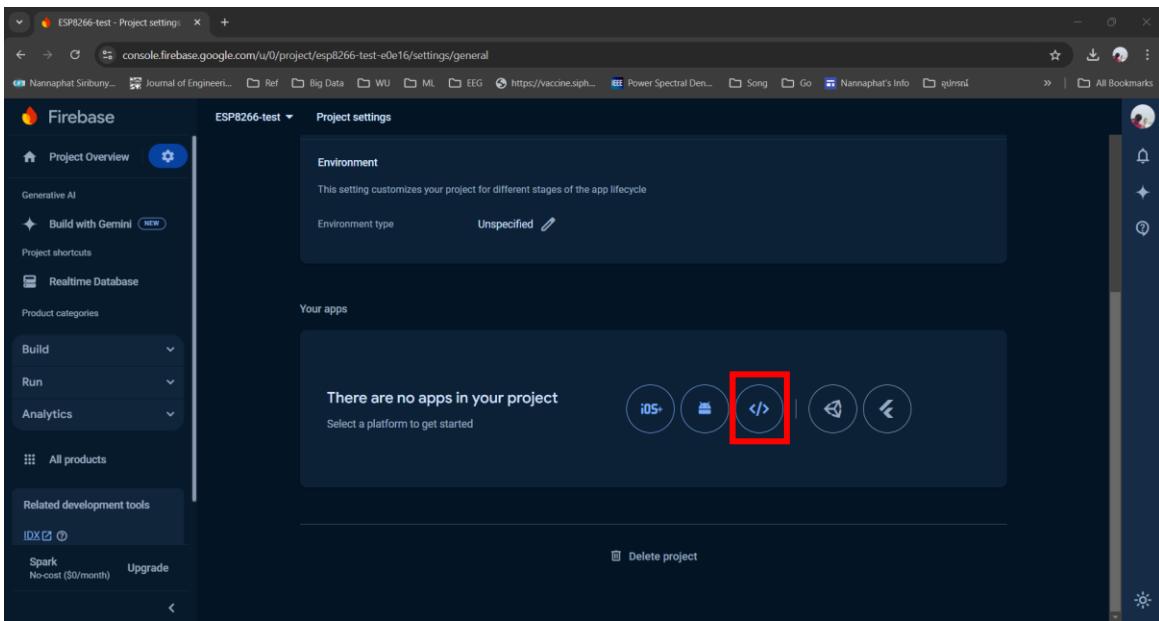
รูปที่ 10.11 ตั้งค่า rules สำหรับ Test mode

- คลิกที่รูปฟันเฟืองข้าง Project Overview เลือก Project settings



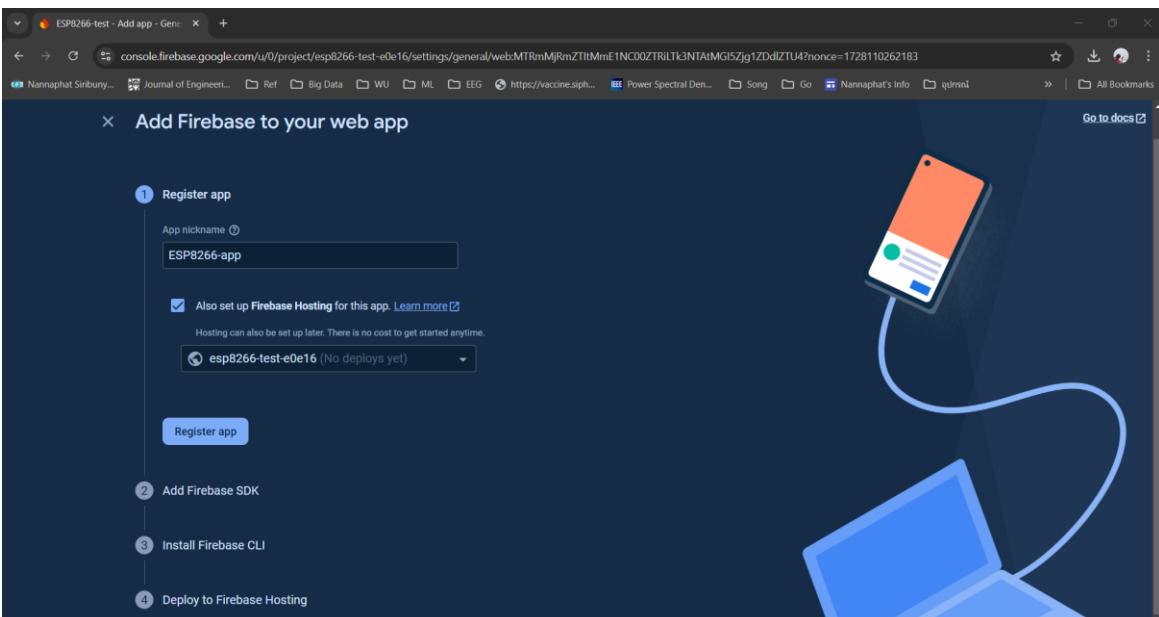
รูปที่ 10.12 Project Settings

- ในแท็บ General เลื่อนลงมาที่หัวข้อ Your apps และเลือก Web API Key



รูปที่ 10.13 Web API Key

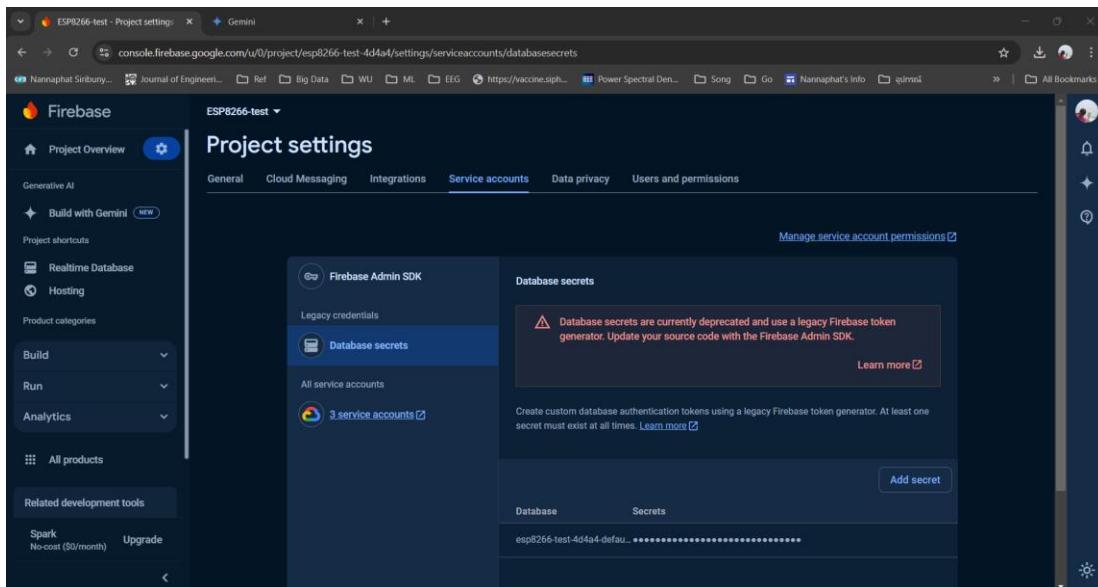
- ตั้งชื่อ app เลือก Also set up Firebase Hosting for this app และกด Register app



รูปที่ 10.14 Register App

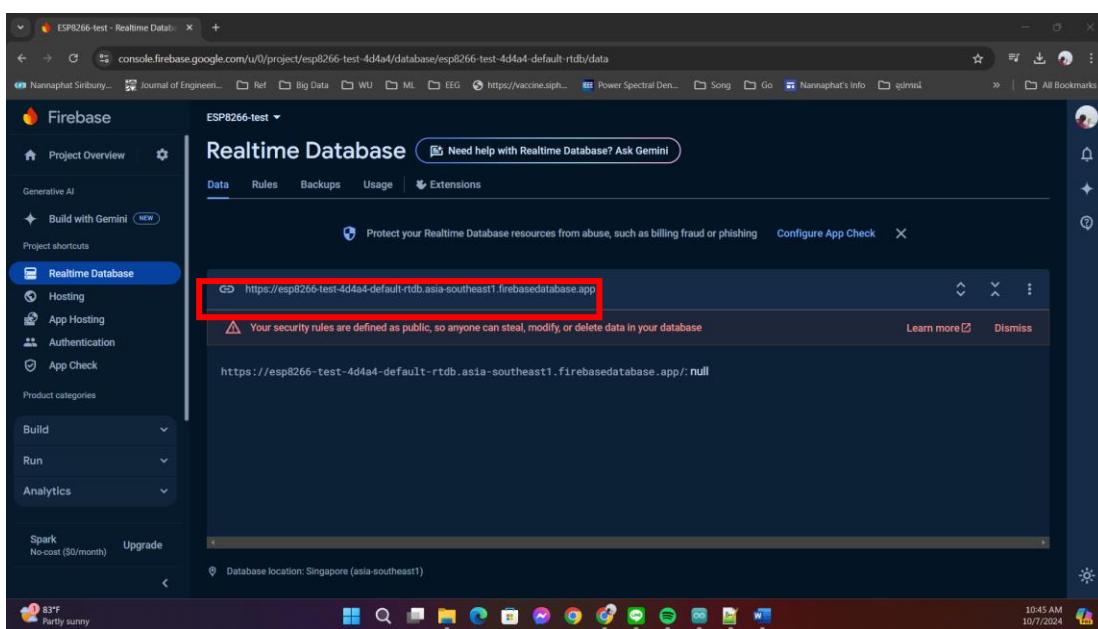
- ที่หัวข้อ Add Firebase SDK ให้กด Next
- ที่หัวข้อ Install Firebase CLI ให้กด Next

- ที่ Deploy to Firebase Hosting ให้กด Continue to console
- ที่หน้า Project Settings เลือกแท็บ Service accounts เลือก Database secrets จะเห็น Secrets Key ให้กด show และคัดลอกไว้เพื่อนำไปใส่ในโค้ดของ ESP8266



รูปที่ 10.15 Secret Key ของ Firebase

- กลับมาที่หน้า Realtime Database ที่แท็บ Data จะมี Database URL ให้คัดลอกไว้เพื่อนำไปใส่ในโค้ดของ ESP8266



รูปที่ 10.16 Database URL ของ Firebase

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

#include <addons/RTDBHelper.h>

#define WIFI_SSID "YourNetworkName"
#define WIFI_PASSWORD "YourPassword"

#define DATABASE_URL "DatabaseURL"
#define DATABASE_SECRET "DatabaseSecret"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

unsigned long dataMillis = 0;
int count = 0;

void setup()
{
    Serial.begin(115200);

    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

    Serial.print("Connecting to Wi-Fi");
    unsigned long ms = millis();
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(300);
    }
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();

    Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

    config.database_url = DATABASE_URL;
    config.signer.tokens.legacy_token = DATABASE_SECRET;

    Firebase.reconnectNetwork(true);

    fbdo.setBSSLBufferSize(4096 /* Rx buffer size in bytes from 512 - 16384 */, 1024 /* Tx buffer size in bytes from 512 - 16384 */);

    Firebase.begin(&config, &auth);
}

void loop()
{
    if (millis() - dataMillis > 5000)
    {
        dataMillis = millis();
        Serial.printf("Set int... %s\n", Firebase.setInt(fbdo, "/test/int", count++) ? "ok" : fbdo.errorReason().c_str());
    }
}
```

- YourNetworkName ให้กรอกชื่อเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม แนะนำให้ใช้ Hotspot ของโทรศัพท์มือถือ
- YourPassword ให้กรอกรหัสผ่านของเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม
- DatabaseURL ให้นำ Database URL ที่คัดลอกไว้มาใส่ในรูปแบบ

`<databaseName>.<region>.firebaseapp.firebaseio.com`
- DatabaseSecret ให้นำ Secret Key ของ Firebase ที่คัดลอกไว้มาใส่

- เปิดหน้า Serial Monitor
- เปิดหน้า Realtime Database ของ Firebase

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

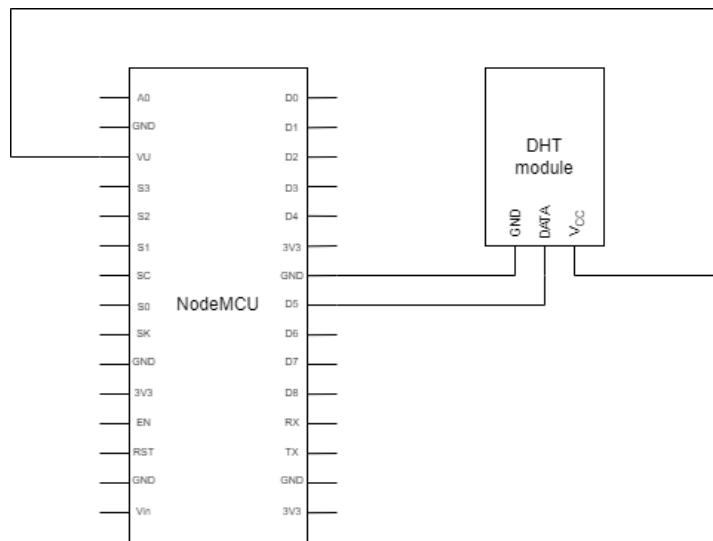
Lab 10.2: การบันทึกค่าอุณหภูมิและความชื้นลงใน Realtime Database ของ Firebase

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- โมดูล DHT 1 ตัว
- สาย Jumper
- Breadboard 1 อัน
- Firebase

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 10.17 วงจรสำหรับแลบที่ 10.2

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <DHT.h>

#include <addons/RTDBHelper.h>

#define WIFI_SSID "YourNetworkName"
#define WIFI_PASSWORD "YourPassword"

#define DATABASE_URL "DatabaseURL"
#define DATABASE_SECRET "DatabaseSecret"

#define DHTPIN D5
#define DHTTYPE DHT11

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
DHT dht(DHTPIN, DHTTYPE);

unsigned long dataMillis = 0;

void setup()
{
    Serial.begin(115200);

    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

    Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(300);
    }
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();

    Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
    config.database_url = DATABASE_URL;
    config.signer.tokens.legacy_token = DATABASE_SECRET;

    Firebase.reconnectNetwork(true);
    fbdo.setBSSLBufferSize(4096, 1024);
    Firebase.begin(&config, &auth);

    dht.begin();
}
```

```

void loop()
{
    if (millis() - dataMillis > 5000)
    {
        dataMillis = millis();

        float humidity = dht.readHumidity();
        float temperature = dht.readTemperature();

        if (isnan(humidity) || isnan(temperature))
        {
            Serial.println("Failed to read from DHT sensor!");
            return;
        }

        Serial.printf("Temperature: %.2f °C\n", temperature);
        Serial.printf("Humidity: %.2f %%\n", humidity);

        String timestamp = String(millis());

        if (Firebase.pushFloat(fbdo, "/sensor/temperature/" + timestamp, temperature))
        {
            Serial.println("Temperature uploaded successfully");
        }
        else
        {
            Serial.println(fbdo.errorReason());
        }

        if (Firebase.pushFloat(fbdo, "/sensor/humidity/" + timestamp, humidity))
        {
            Serial.println("Humidity uploaded successfully");
        }
        else
        {
            Serial.println(fbdo.errorReason());
        }
    }
}

```

- YourNetworkName ให้กรอกชื่อเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม แนะนำให้ใช้ Hotspot ของโทรศัพท์มือถือ
- YourPassword ให้กรอกรหัสผ่านของเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม
- DatabaseURL ให้นำ Database URL ที่คัดลอกไว้มาใส่ในรูปแบบ

 <databaseName>.<region>.firebase.database.app
- DatabaseSecret ให้นำ Secret Key ของ Firebase ที่คัดลอกไว้มาใส่
- เปิดหน้า Serial Monitor
- เปิดหน้า Realtime Database ของ Firebase

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

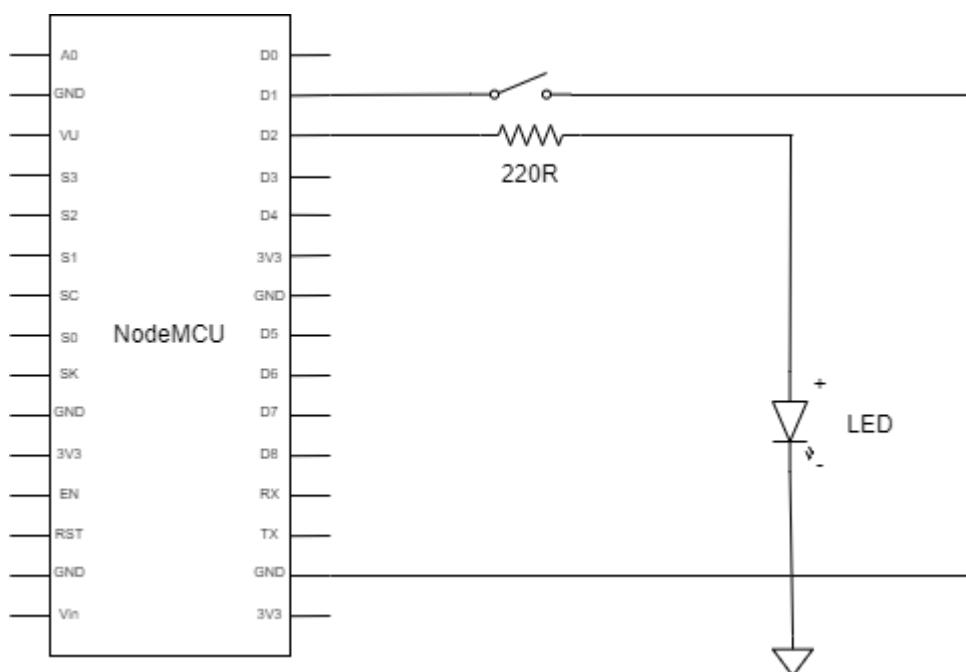
Lab 10.3: การบันทึกสถานะของสวิตซ์ลงใน Realtime Database ของ Firebase

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- หลอด LED 1 อัน
- ตัวต้านทานขนาด $220\ \Omega$ 1 อัน
- สาย Jumper
- Breadboard 1 อัน
- Tact Switch (สวิตซ์กดติดปล่อยดับ) 1 อัน
- Firebase

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 10.18 วงจรสำหรับแล็บที่ 10.3

- พิมพ์โค้ดต่อไปนี้แล้วทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <addons/RTDBHelper.h>

#define WIFI_SSID "YourNetworkName"
#define WIFI_PASSWORD "YourPassword"

#define DATABASE_URL "DatabaseURL"
#define DATABASE_SECRET "DatabaseSecret"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

unsigned long dataMillis = 0;
int switchState = 0;

#define SWITCH_PIN D1
#define LED_PIN D2

void setup()
{
    Serial.begin(115200);

    pinMode(SWITCH_PIN, INPUT_PULLUP);
    pinMode(LED_PIN, OUTPUT);

    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

    Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(300);
    }
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();

    Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

    config.database_url = DATABASE_URL;
    config.signer.tokens.legacy_token = DATABASE_SECRET;

    Firebase.reconnectNetwork(true);
    fbdo.setBSSLBufferSize(4096, 1024);

    Firebase.begin(&config, &auth);
}
```

```
void loop()
{
    int currentSwitchState = digitalRead(SWITCH_PIN);

    if (currentSwitchState != switchState)
    {
        switchState = currentSwitchState;

        String path = "/switchState";
        String stateStr = switchState == HIGH ? "OFF" : "ON";
        Serial.printf("Set switch state to %s... %s\n", stateStr.c_str(), Firebase.setString(fbdo, path, stateStr) ? "ok" : fbdo.errorReason().c_str());

        if (switchState == LOW)
        {
            digitalWrite(LED_PIN, HIGH);
        }
        else
        {
            digitalWrite(LED_PIN, LOW);
        }
    }

    delay(100);
}
```

- YourNetworkName ให้กรอกชื่อเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม แนะนำให้ใช้ Hotspot ของโทรศัพท์มือถือ
- YourPassword ให้กรอกรหัสผ่านของเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม
- DatabaseURL ให้นำ Database URL ที่คัดลอกไว้มาใส่ในรูปแบบ
`<databaseName>.<region>.firebase.database.app`
- DatabaseSecret ให้นำ Secret Key ของ Firebase ที่คัดลอกไว้มาใส่
- เปิดหน้า Serial Monitor
- เปิดหน้า Realtime Database ของ Firebase

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

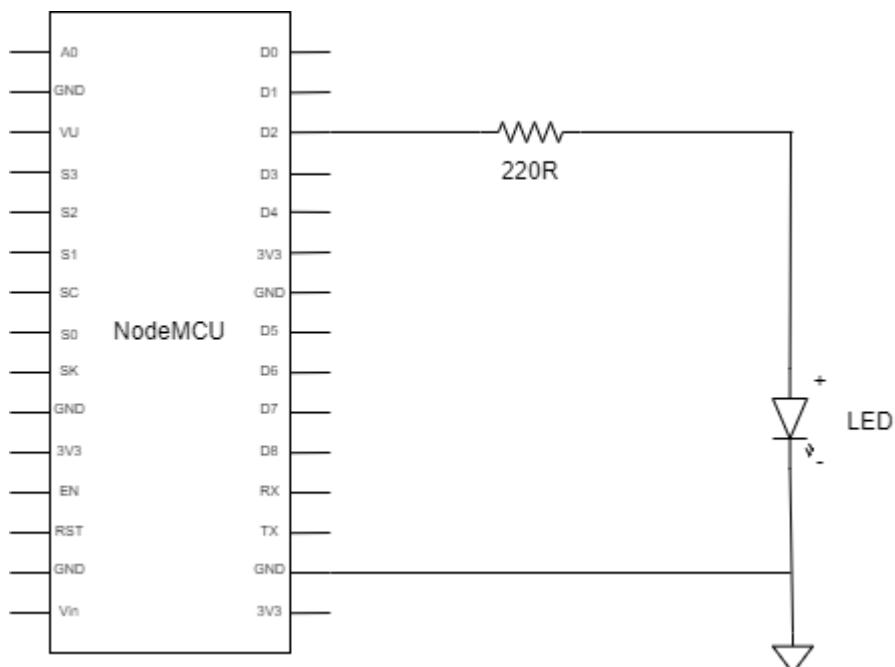
Lab 10.4: การบันทึกสถานะของสวิตช์ลงใน Realtime Database ของ Firebase ร่วมกับการสั่งการบน Blynk

อุปกรณ์ที่ต้องใช้

- บอร์ด NodeMCU 1 ตัว
- หลอด LED 1 อัน
- ตัวต้านทานขนาด 220Ω 1 อัน
- สาย Jumper
- Breadboard 1 อัน
- Blynk
- Firebase

ขั้นตอนในการทำ

- ต่อวงจรดังที่แสดงในรูป



รูปที่ 10.19 วงจรสำหรับแลบที่ 10.4

- พิมพ์โค้ดต่อไปนี้

```
#define BLYNK_TEMPLATE_ID      "YourTemplateID"
#define BLYNK_TEMPLATE_NAME    "Quickstart Device"
#define BLYNK_AUTH_TOKEN       "YourAuthToken"

#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <FirebaseESP8266.h>
#include <addons/RTDBHelper.h>

#define WIFI_SSID "YourNetworkName"
#define WIFI_PASSWORD "YourPassword"

#define DATABASE_URL "databaseURL"
#define DATABASE_SECRET "DatabaseSecret"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

int ledState = LOW;
#define LED_PIN D2

void setup()
{
  Serial.begin(115200);

  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, ledState);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());

  config.database.url = DATABASE_URL;
  config.signer.tokens.legacy_token = DATABASE_SECRET;
  Firebase.reconnectNetwork(true);
  Firebase.begin(&config, &auth);

  Blynk.begin(BLYNK_AUTH_TOKEN, WIFI_SSID, WIFI_PASSWORD);
}

BLYNK_WRITE(V2)
{
  ledState = param.asInt();

  digitalWrite(LED_PIN, ledState);

  String path = "/ledState";
  String stateStr = ledState == HIGH ? "ON" : "OFF";
  Serial.printf("Set LED state to %s... %s\n", stateStr.c_str(), Firebase.setString(fbdo, path, stateStr) ? "ok" : fbdo.errorReason().c_str());
}

void loop()
{
  Blynk.run();
}
```

- YourNetworkName ให้กรอกชื่อเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม แนะนำให้ใช้ Hotspot ของโทรศัพท์มือถือ
- YourPassword ให้กรอกรหัสผ่านของเครือข่ายอินเทอร์เน็ตที่จะทำการเชื่อม
- YourAuthToken ให้นำ Token จากการสร้างแอปพลิเคชันใน Blynk มากรอก
- YourTemplateID ให้นำ Template ID จากการสร้างแอปพลิเคชันใน Blynk มากรอก
- DatabaseURL ให้นำ Database URL ที่คัดลอกไว้มาใส่ในรูปแบบ

`<databaseName>.<region>.firebasedatabase.app`
- DatabaseSecret ให้นำ Secret Key ของ Firebase ที่คัดลอกไว้มาใส่

- สร้างแอปพลิเคชัน Blynk เช่นเดียวกับแล็บที่ 9.2 และนำค่า AuthToken กับ TemplateID ไปใส่ในโค้ด
- ไปที่แท็บ Datastreams ทำการสร้าง Virtual Pin สำหรับ Switch ดังแสดงในรูปที่ 10.20

Virtual Pin Datastream

General Expose to Automations

NAME	ALIAS	
 Switch	Switch 	
PIN	DATA TYPE	
V2	Integer	
UNITS		
None		
MIN	MAX	DEFAULT VALUE
0	1	0

Cancel **Save**

(ก)

Virtual Pin Datastream

General Expose to Automations

AUTOMATION TYPE

Switch

Condition
Allow users to use this datastream as an Automation condition

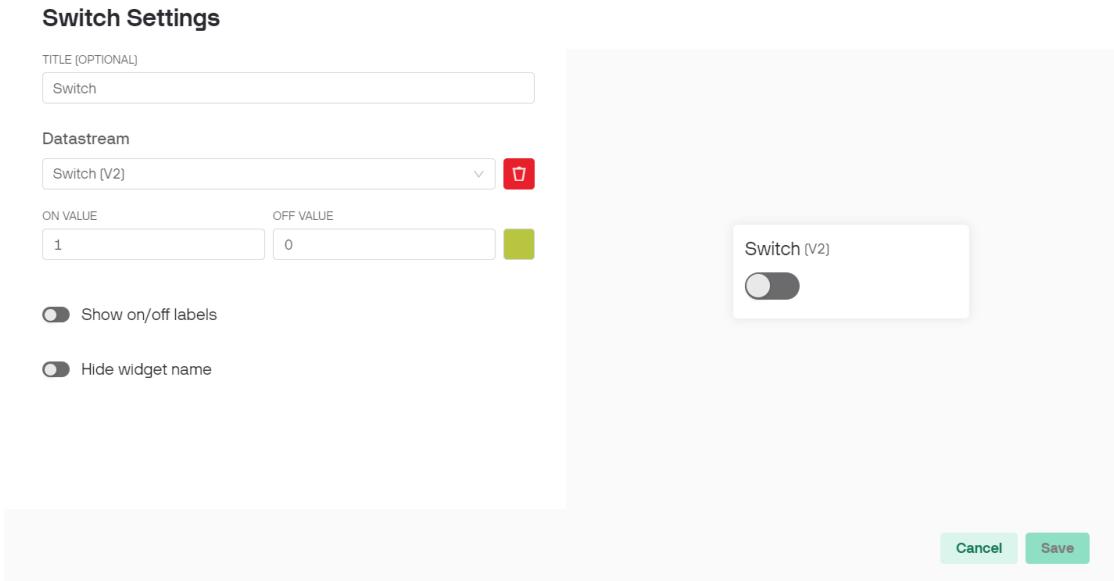
Action
Allow users to change this datastream's state using Automations

Cancel **Save**

(ข)

รูปที่ 10.20 การกำหนด Virtual Pin (ก) แท็บ General (ข) แท็บ Expose to Automations

- ที่ Web Dashboard ทำการกำหนดหน้า Dashboard เป็น Switch โดยทำการแก้ไขให้เข้มกับ Datastream V2 จากนั้นกด Save And Apply



รูปที่ 10.21 การตั้งค่าสวิทช์

- ทำการ Verify โค้ด และอัปโหลดโค้ดลงบอร์ด
- เปิด Serial Monitor
- เปิด Firebase
- ทดลองเปิดปิดสวิตช์บน Blynk

ผลการทำงานหลังจากอัปโหลดโค้ดลงบอร์ด

Reference

- [1] NodeMCU Team, NodeMCU Documentation, Retrieved from: <https://nodemcu.readthedocs.io/en/release/>, 21 Sep. 2024, (online).
- [2] Make-It.ca, NodeMCU Specification, Retrieved from: <https://www.make-it.ca/nodemcu-details-specifications/>, 21 Spe. 2024, (online).
- [3] ภาสกร พาเจริญ, พัฒนา IoT บนแพลตฟอร์ม Arduino ด้วย NodeMCU, กรุงเทพฯ ประเทศไทย, 2562. 272 หน้า.
- [4] Programming Arduino UNO clone with built-in WiFi module UNO R3 + WiFi ATmega328P + ESP8266, Retrieved from: <https://blog.devgenius.io/programming-arduino-uno-clone-with-built-in-wifi-module-uno-r3-wifi-atmega328p-esp8266-9494d9a90cfa>, 23 Sep. 2024, (online)