

สารบัญ

การประยุกต์ใช้ IoT สำหรับบุคคลทั่วไป.....	3
เนื้อหาหลักสูตร.....	5
แนะนำ IoT.....	6
รู้จัก Arduino.....	21
รู้จัก ESP8266.....	28
รู้จัก NodeMCU.....	33
แนะนำ ESP32.....	40
ซอฟต์แวร์ที่ใช้พัฒนา.....	44
ESP32 โมดูล.....	46
บอร์ดพัฒนา ESP32.....	49
เปรียบเทียบระหว่างESP8266 และ ESP32.....	53
การติดตั้ง Arduino ide.....	56
การเสัต์ค่าบอร์ด ESP32 ใน Arduino ide.....	74
การใช้งาน arduino ide เบื้องต้น.....	76
EX1 WiFi Scan_ทดสอบการอัฟโหลด sketch.....	80
การใช้งานพอร์ตดิจิตอล และอนาล็อก GPIO.....	82
EX2 ไฟกระพริบ.....	85
EX3 อ่านค่าขาสวิตช์ดิจิตอล.....	87
EX4 สวิตช์ดิจิตอล กดติดปล่อยดับ.....	90
EX5 สวิตช์ดิจิตอล กดติด กดดับ.....	93
EX6 เซ็นเซอร์สัมผัส.....	97
รู้จักเซ็นเซอร์ประเภทต่างๆ.....	101
การเขียนโปรแกรมใช้งานพอร์ต อนาล็อก GPIO.....	110
EX7 วัดค่าความชื้นดิน.....	111
EX8 วัดค่าความชื้นดินและแสดงสถานะบนหลอด.....	114
EX9 วัดค่าความชื้นดินและแสดงสถานะบนหลอด RGB LED.....	116
EX10 วัดค่าความชื้นดินและแสดงสถานะบนหลอด RGB LED CALIBRATE.....	118
EX11 อ่านค่าอุณหภูมิและความชื้นสัมพัทธ์.....	120
EX12 อ่านค่าความเข้มแสงของ LDR Sensor.....	125
EX13 ควบคุมหลอด LED ตามความเข้มแสง.....	128
Blynk.....	130
EX14 สร้างโปรเจคควบคุมไฟ LED ออนไลน์.....	139

สารบัญ

EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์.....	145
EX15-1 Blynk Timer.....	155
EX15-2 Blynk Time Input.....	158
Blynk HTTP RESTful API.....	162
Blynk Add Device.....	163
Blynk Share Controller.....	164
Blynk Clone Controller.....	165
QR Smart Farm Project.....	168
EX16 ส่งการแจ้งเตือนไป LINE ด้วย LINE API.....	170
โหมดประหยัดพลังงาน.....	179
EX17 แจ้งเตือนอุณหภูมิความชื้นสัมพัทธ์ออนไลน์.....	184
Firebase iot realtime database.....	187
EX18_FIREBASE_COUNTER.....	198
EX19_FIREBASE_DHT11.....	200
EX20 ควบคุมหลอดไฟ LED บน Firebase.....	205
EX21 SMART FARM PROJECT.....	209

การอบรมเชิงปฏิบัติการ

การประยุกต์ใช้ IoT สำหรับบุคคลทั่วไป



หลักการและเหตุผล

เมื่ออุปกรณ์อิเล็กทรอนิกส์มีความฉลาดมากขึ้นจนสามารถเชื่อมต่อกับอินเทอร์เน็ตได้ สิ่งเหล่านี้เรียกว่า smart device มันจะเข้ามาช่วยอำนวยความสะดวกให้มนุษย์ได้อย่างมากมาย เราสามารถควบคุม ฝ้าระวัง ติดตาม สิ่งของจากที่ไหนก็ได้ ถ้าสามารถเชื่อมต่อกับอินเทอร์เน็ต ดังนั้นการส่งเสริมให้ความรู้ ทักษะเกี่ยวกับเรื่อง Internet of things (IoT) เพื่อกระตุ้นให้เกิดการพัฒนานวัตกรรมใหม่ๆ ทั้งในด้าน ผลิตภัณฑ์ การบริการ จึงเป็นเรื่องที่สำคัญ

วัตถุประสงค์โครงการ

- ผู้เข้าอบรมสามารถนำความรู้ที่ได้จากการอบรมไปสร้างผลิตภัณฑ์ในชีวิตประจำวันและต่อยอดทางธุรกิจได้ ต่อยอดแนวคิดเกิดเป็นนวัตกรรมใหม่ๆ

สิ่งที่ผู้เข้าอบรมจะได้รับ

- เข้าใจสถาปัตยกรรม IoT แบบต่างๆ
- เรียนรู้และใช้งาน Arduino IDE เพื่อเขียนโปรแกรมลงบนบอร์ด ESP32
- ประยุกต์ใช้ IoT กับเครื่องมือสื่อสารออนไลน์ จัดการ IoT ผ่านโมบาย
- ศึกษาเคสตัวอย่างการทำตัวอย่าง SmartFarm

เนื้อหาหลักสูตร

วันที่ 1

- แนะนำ IoT และบอร์ด ESP32
- การติดตั้ง ตั้งค่า เขียนโปรแกรมบน Arduino IDE
- การใช้งานพอร์ตดิจิทัล และอนาล็อก GPIO
- การเชื่อมต่อ ESP32 กับ WiFi Network
- การเขียนโปรแกรมจัดการ LED / LDR
- รู้จักเซ็นเซอร์ประเภทต่าง ๆ
- ทดสอบ touch sensor และการแสดงผล
- การสั่งการเปิดปิดไฟ วัดอุณหภูมิและความชื้นสัมพัทธ์ วัดความชื้นดิน ผ่านทางออนไลน์
- การใช้งาน dashboard และการพล็อตกราฟ

วันที่ 2

- การใช้งาน sleep mode แบบต่างๆ
- การเขียนโปรแกรมแบบ multi tasking
- การเชื่อมต่อ IoT Cloud Platform
- การเชื่อมต่อ ESP32 กับเครื่องมือสื่อสารสังคมออนไลน์ LINE
- สั่งการ ESP32 ผ่าน Blynk Cloud Platform
- การใช้งาน Blynk บนโมบายแอปพลิเคชัน Android / IOS
- การเชื่อมต่อฐานข้อมูลออนไลน์ Google Firebase

แนะนำ IoT



แนวคิด Internet of Things

1999!

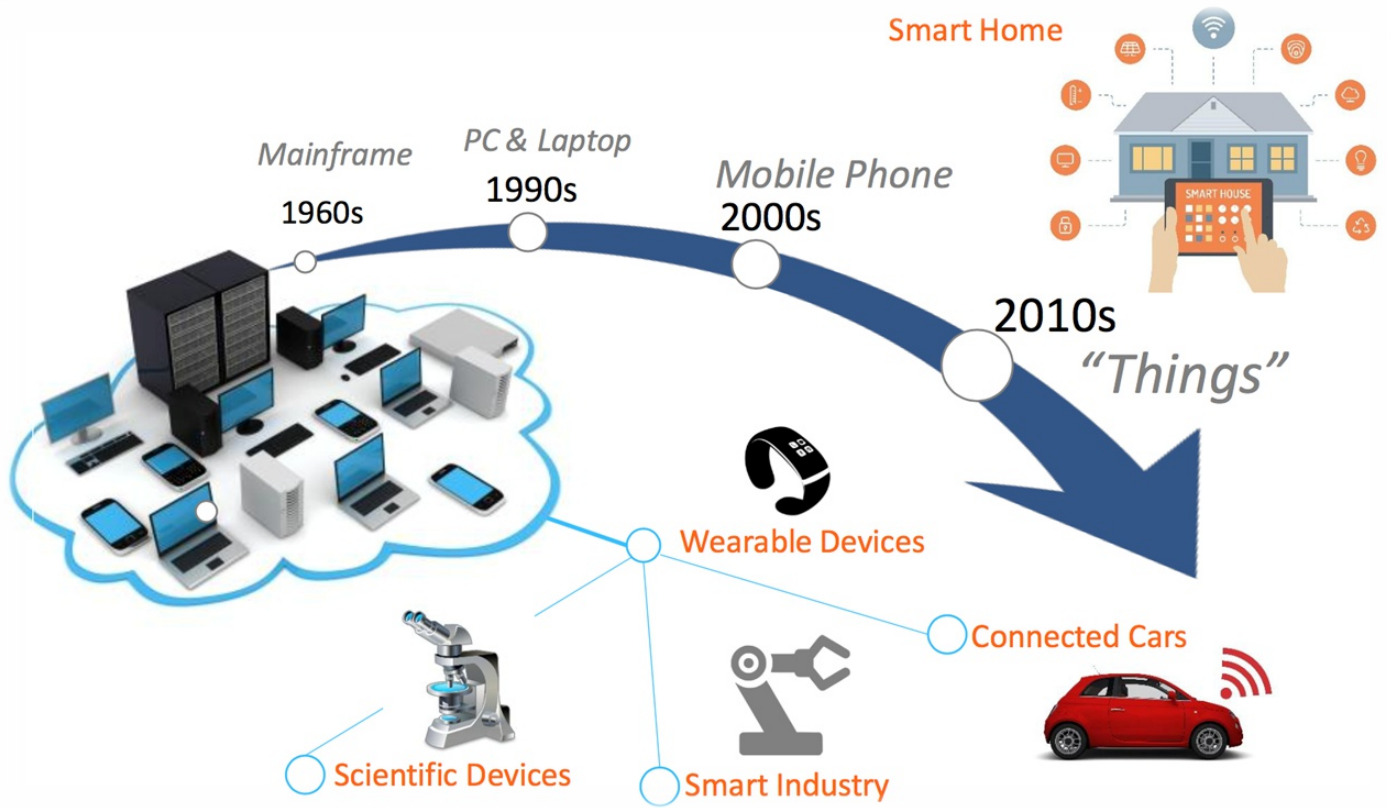
**“THE INTERNET OF THINGS IS
ABOUT EMPOWERING COMPUTERS
...SO THEY CAN SEE, HEAR
AND SMELL THE WORLD FOR
THEMSELVES”**

**KEVIN ASHTON
INVENTOR OF THE TERM
“INTERNET OF THINGS”**



แนวคิด Internet of Things ถูกคิดค้นขึ้นโดย Kevin Ashton ในปี 1999 ซึ่งเริ่มต้นจากโครงการ “Auto-ID Center” ในมหาวิทยาลัย Massachusetts Institute of Technology จากเทคโนโลยี RFID ย่อมาจากคำว่า Radio Frequency Identification เป็นระบบที่นำเอาคลื่นวิทยุมาใช้ในการสื่อสารข้อมูลระหว่างอุปกรณ์สองชนิด ซึ่งเป็นการสื่อสารแบบไร้สาย ต่อมาในยุคหลังปี 2000 เทคโนโลยีต่างๆ ได้รับการพัฒนาอย่างรวดเร็ว เริ่มมีอุปกรณ์อิเล็กทรอนิกส์ออกมาเป็นจำนวนมาก และยังมีการใช้คำว่า Smart เกิดขึ้นเช่น Smart grid, Smart home, Smart device, Smart network เป็นต้น สิ่งเหล่านี้สามารถเชื่อมต่อกับโลกอินเทอร์เน็ตได้ ทำให้อุปกรณ์ดังกล่าวสามารถสื่อสารแลกเปลี่ยนข้อมูลโดยอาศัยตัว Sensor ในการสื่อสารถึงกัน โดย Kevin ได้ให้นิยามไว้ว่า “Internet-like” ต่อมาจึงมีคำว่า “Things” เข้ามาแทนอุปกรณ์อิเล็กทรอนิกส์ต่างๆ

IoT and Internet

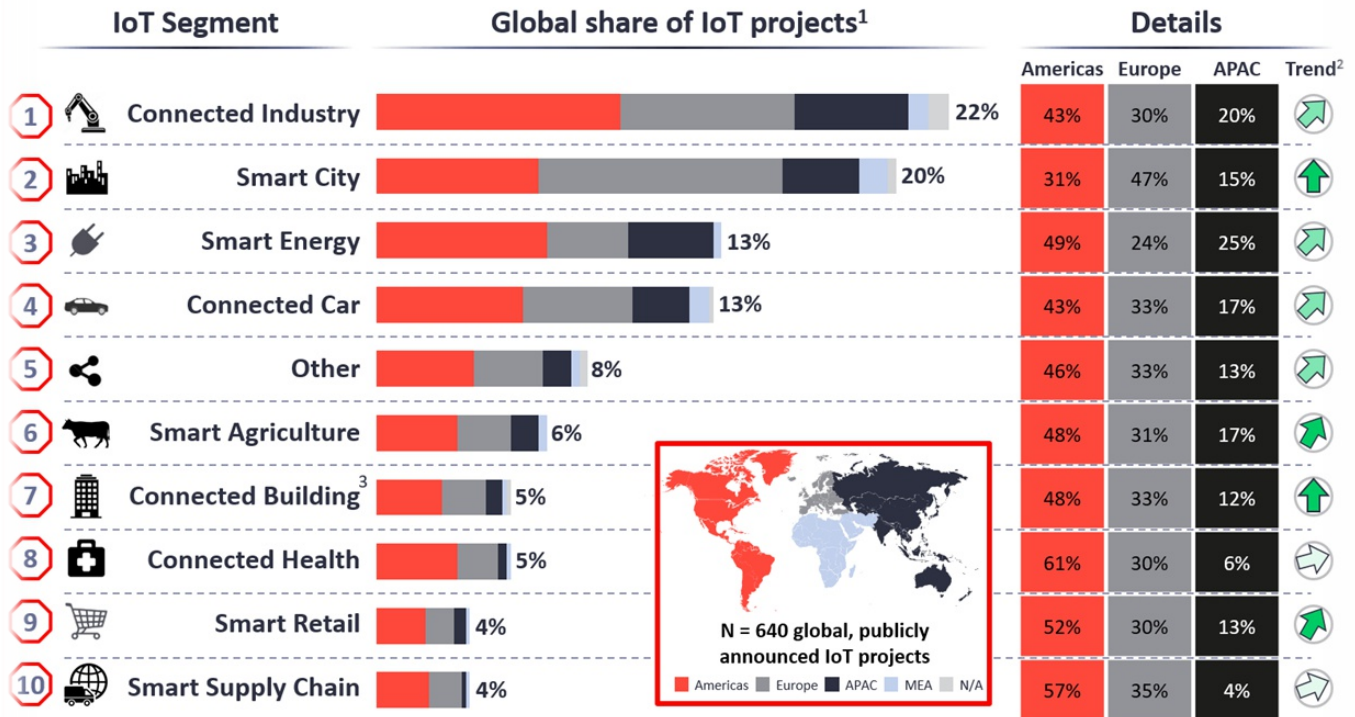


IoT Analytics

IoT ANALYTICS

Q3/2016

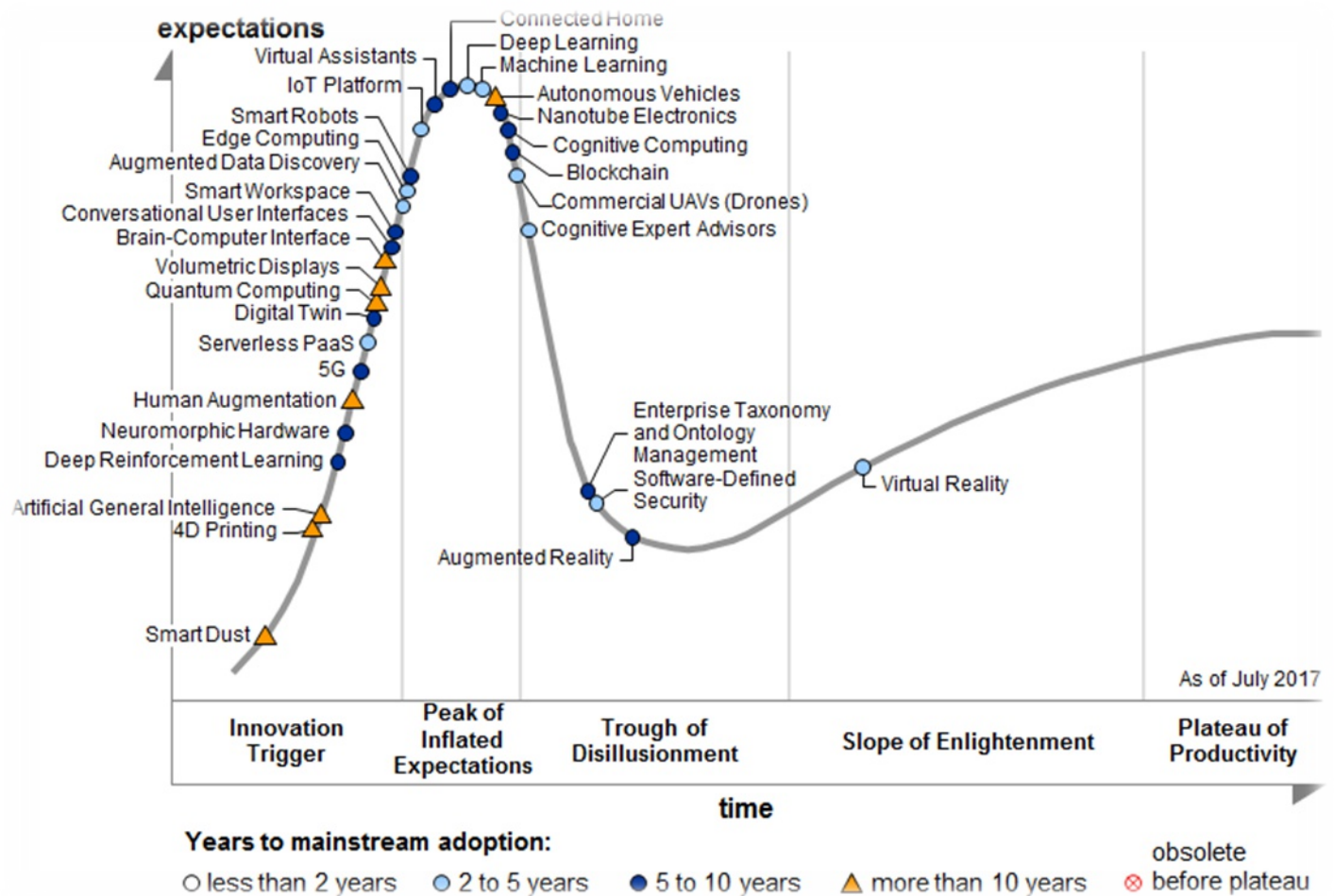
Insights that empower you to understand IoT markets



1. Based on 640+ publicly known enterprise IoT projects.(Not including consumer IoT projects e.g., Wearables, Smart Home) 2. Trend based on IoT Analytics's Q2/2016 IoT Employment Statistics Tracker 3. Not including Consumer Smart Home Solutions Source: IoT Analytics 2016 Global overview of 640 enterprise IoT use cases (August 2016)

IoT : Internet of Things (บางที่เรียก IoE : Internet of Everything) หรือ “อินเทอร์เน็ตในทุกสิ่ง” หมายถึง การที่สิ่งต่างๆ ถูกเชื่อมโยงทุกสิ่งทุกอย่างเข้าสู่โลกอินเทอร์เน็ต ทำให้มนุษย์สามารถ สั่งการควบคุมการใช้งาน อุปกรณ์ต่างๆ ผ่านทางเครือข่ายอินเทอร์เน็ต เช่น การเปิด-ปิด อุปกรณ์เครื่องใช้ไฟฟ้า รถยนต์ โทรศัพท์มือถือ เครื่องมือสื่อสาร เครื่องมือทางการแพทย์ อาคาร บ้านเรือน เครื่องใช้ในชีวิตรประจำวันต่างๆ ผ่านเครือข่ายอินเทอร์เน็ต เป็นต้น

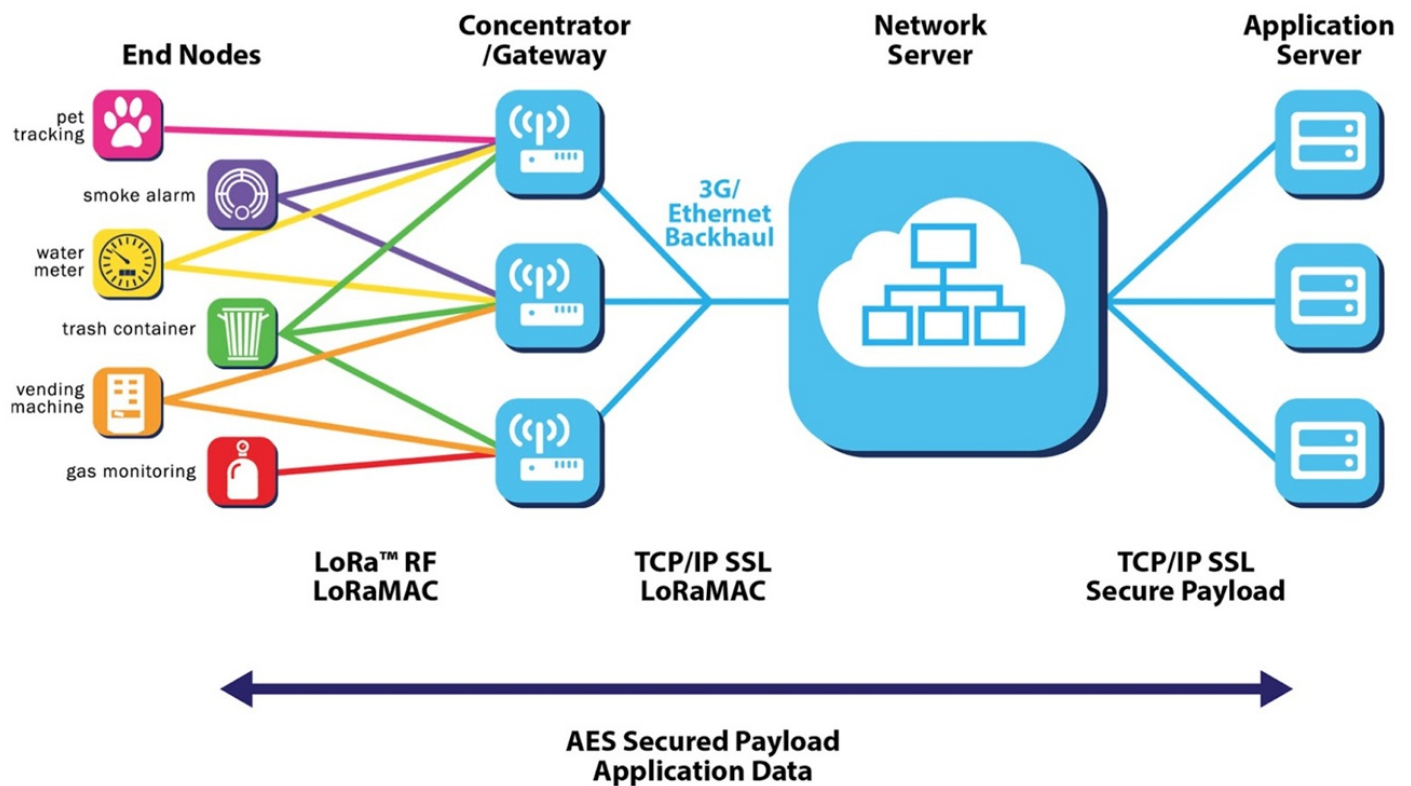
Gartner Hype Cycle 2017



IOT Communications Standard



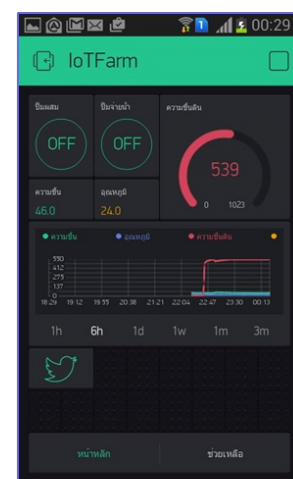
IOT NETWORK



การประยุกต์นำ IoT ไปใช้งาน



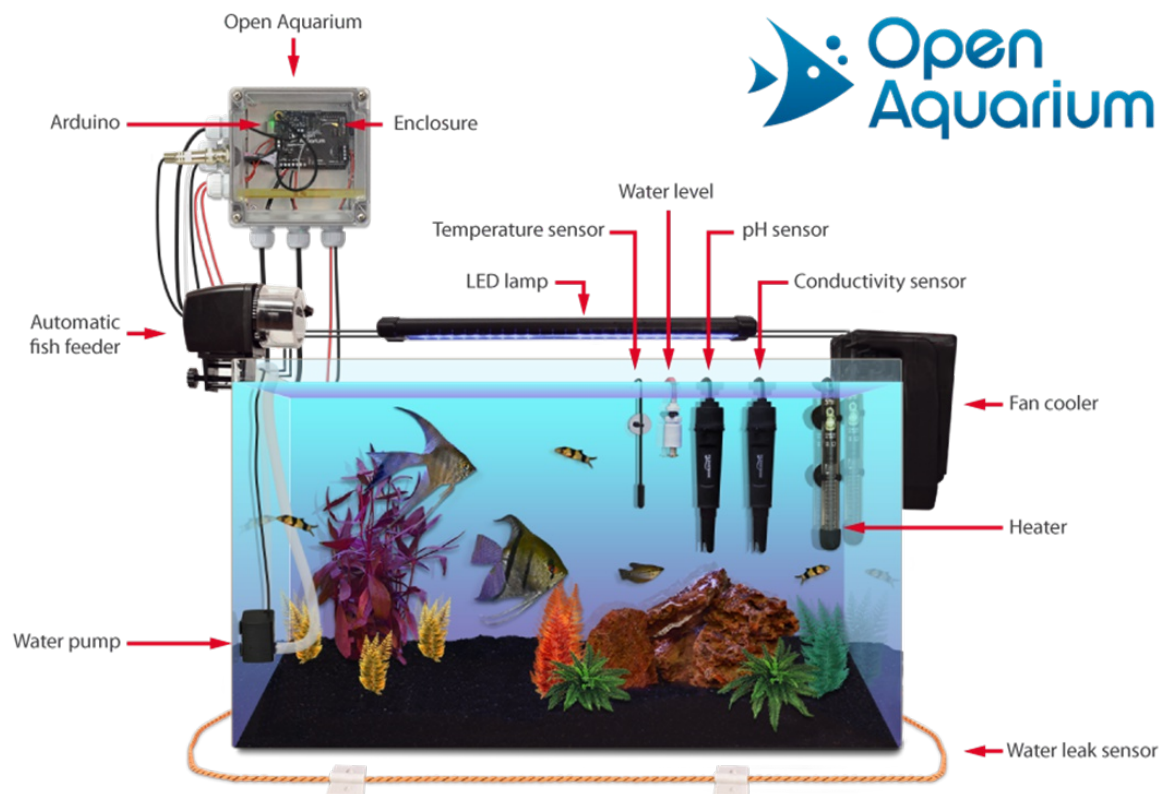
SMART FARM



SMART HOME



SMART Aquarium



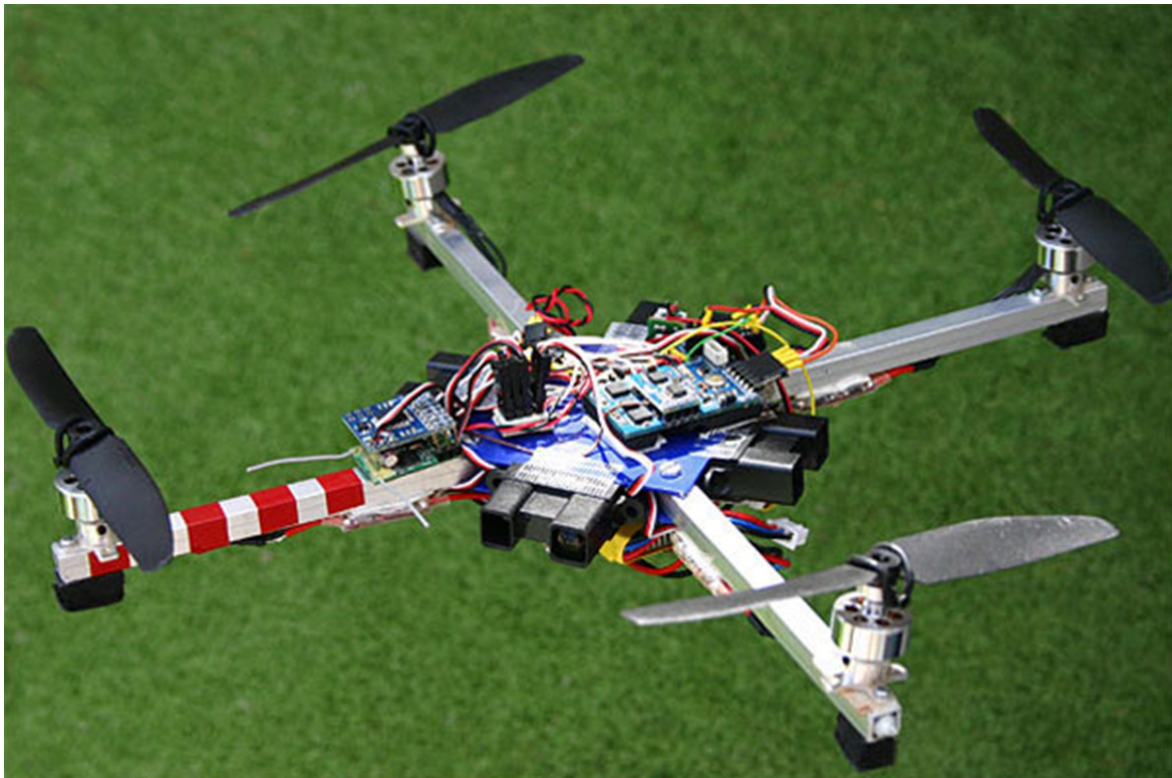
eHealth



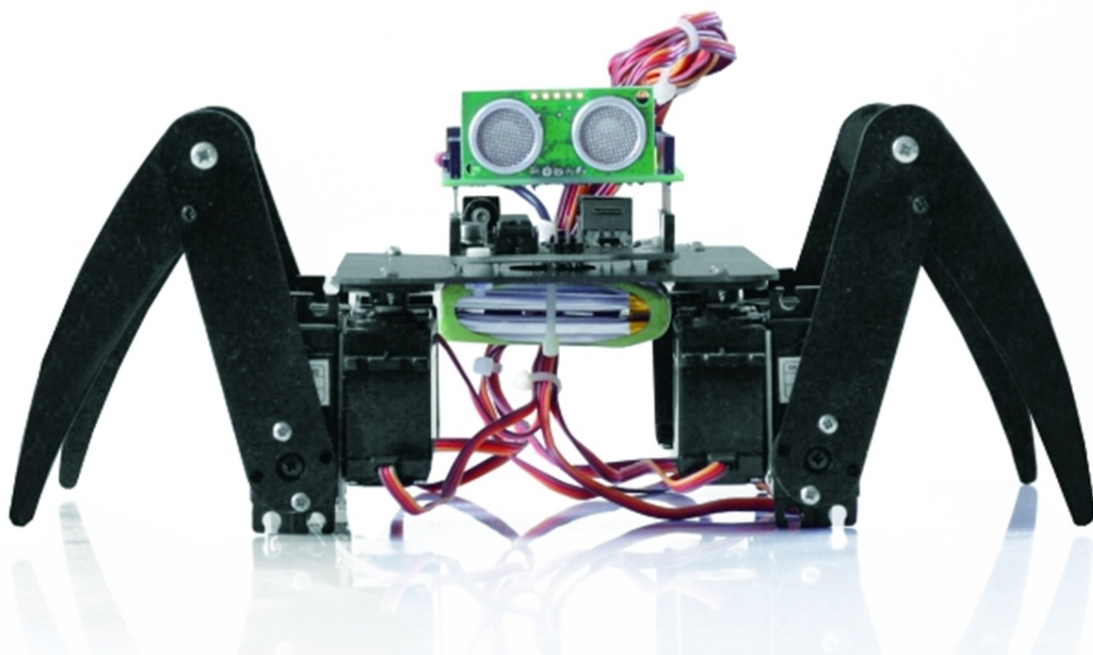
Hydroponics & Garden Plants



DRONES



ROBOTS

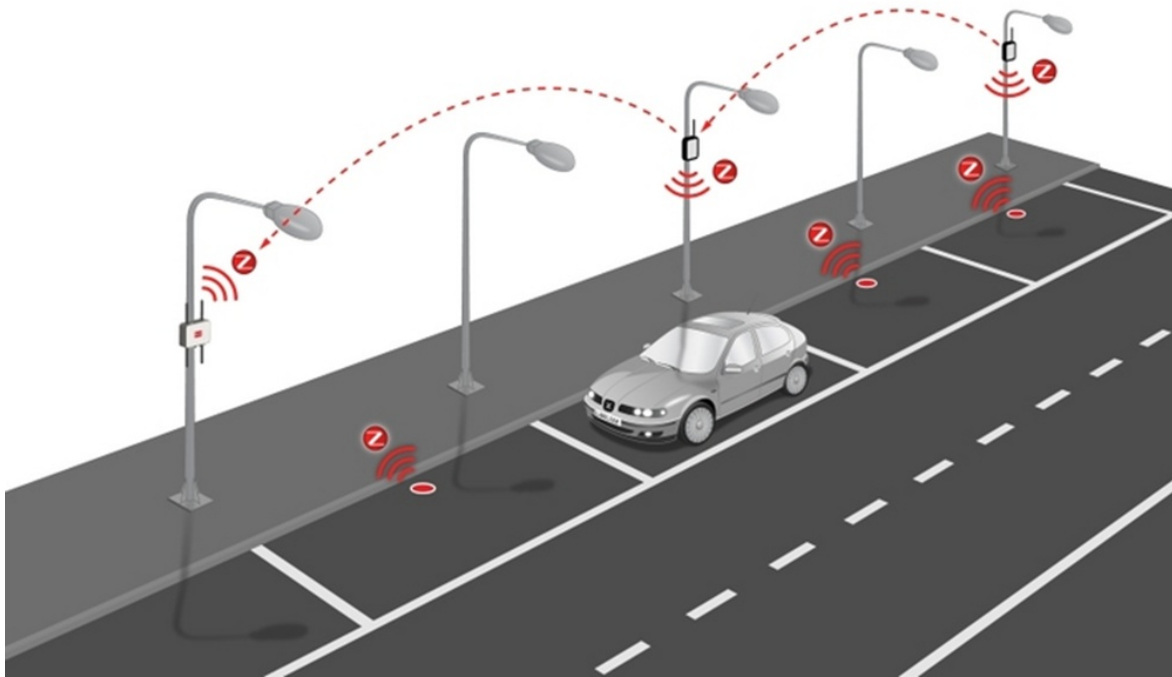


WEARABLE





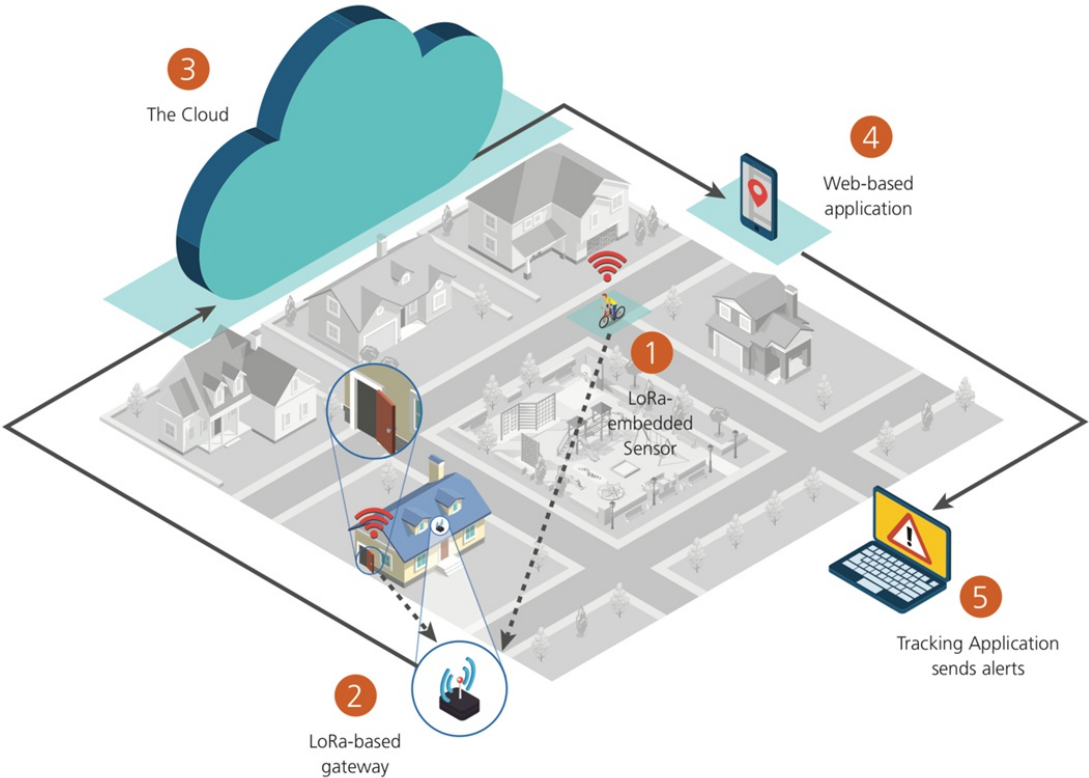
Wide area communication between remote sensors and gateways connected to the network.





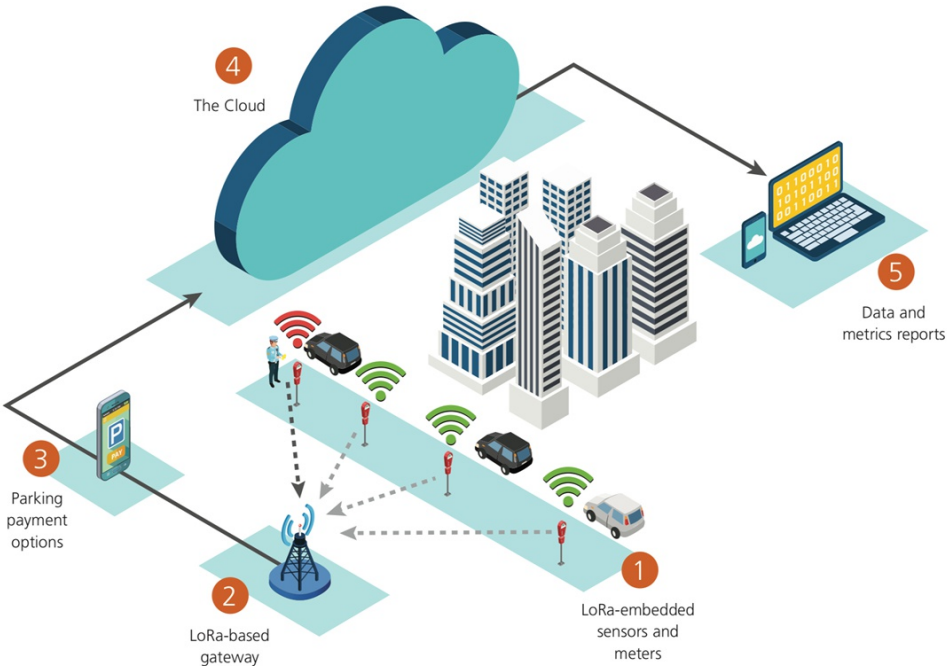
LoRa Applications

Smart Home Asset Tracking



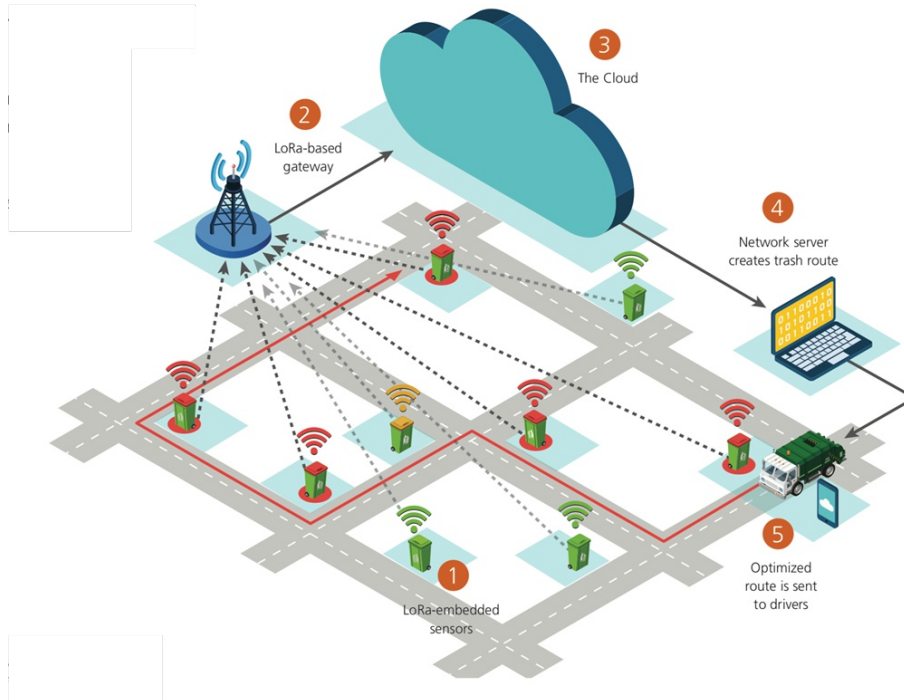
LoRa Applications

Smart Parking

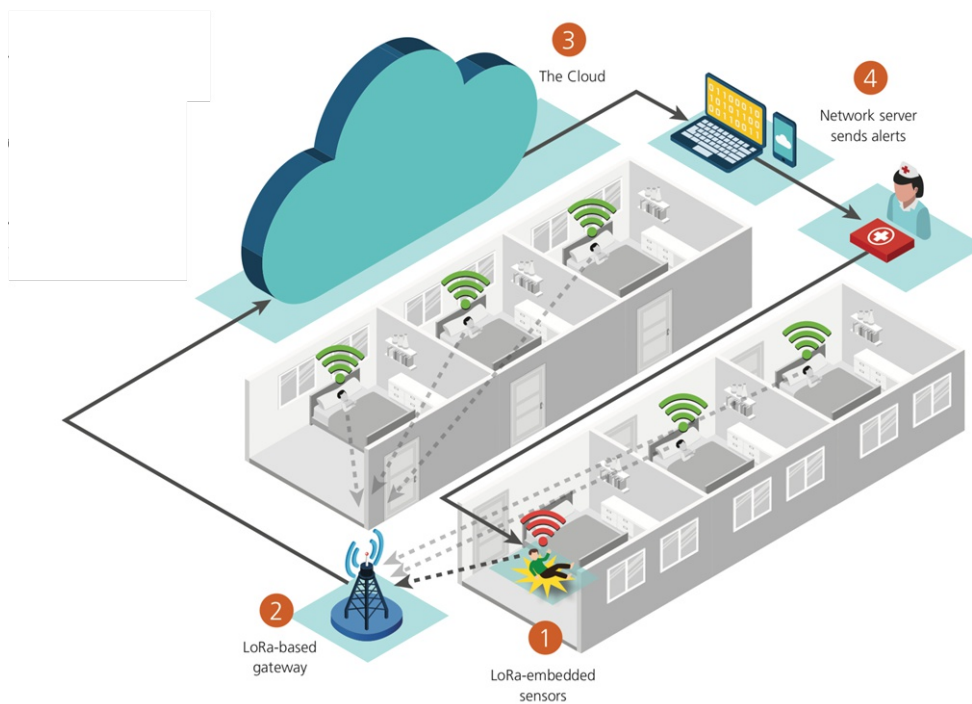




LoRa Applications Waste Management



LoRa Applications Fall Detection



LoRa Applications



Asset Tracking

[Download Application Brief »](#)



Cattle Tracking

[Download Application Brief »](#)



Fleet Management

[Download Application Brief »](#)



Smart Home Asset Tracking

[Download Application Brief »](#)



Smart Parking

[Download Application Brief »](#)



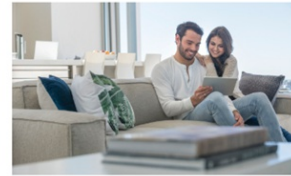
Air Pollution Monitoring

[Download Application Brief »](#)



Agriculture Processing

[Download Application Brief »](#)



Energy Management & Sustainability

[Download Application Brief »](#)



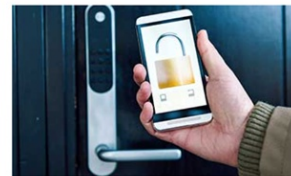
Fall Detection



Fire Detection



Fleet Tracking



Home Security

<http://www.semtech.com/wireless-rf/internet-of-things/lora-applications/briefs#all>



Indoor Air Quality

[Download Application Brief »](#)



Industrial Temp

[Download Application Brief »](#)



Liquid Presence Detection

[Download Application Brief »](#)



Locating Stolen Vehicles & Cargo

[Download Application Brief »](#)



Medical Fridge

[Download Application Brief »](#)



Precision Farming

[Download Application Brief »](#)



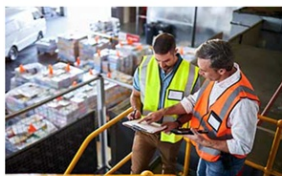
Predictive Maintenance

[Download Application Brief »](#)



Radiation Leak Detection

[Download Application Brief »](#)



Shipment Quality



Smart Lighting



Tank Flow Monitoring



Waste Management

<http://www.semtech.com/wireless-rf/internet-of-things/lora-applications/briefs#all>

รู้จัก Arduino

โครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูล AVR

แบบ Open Source



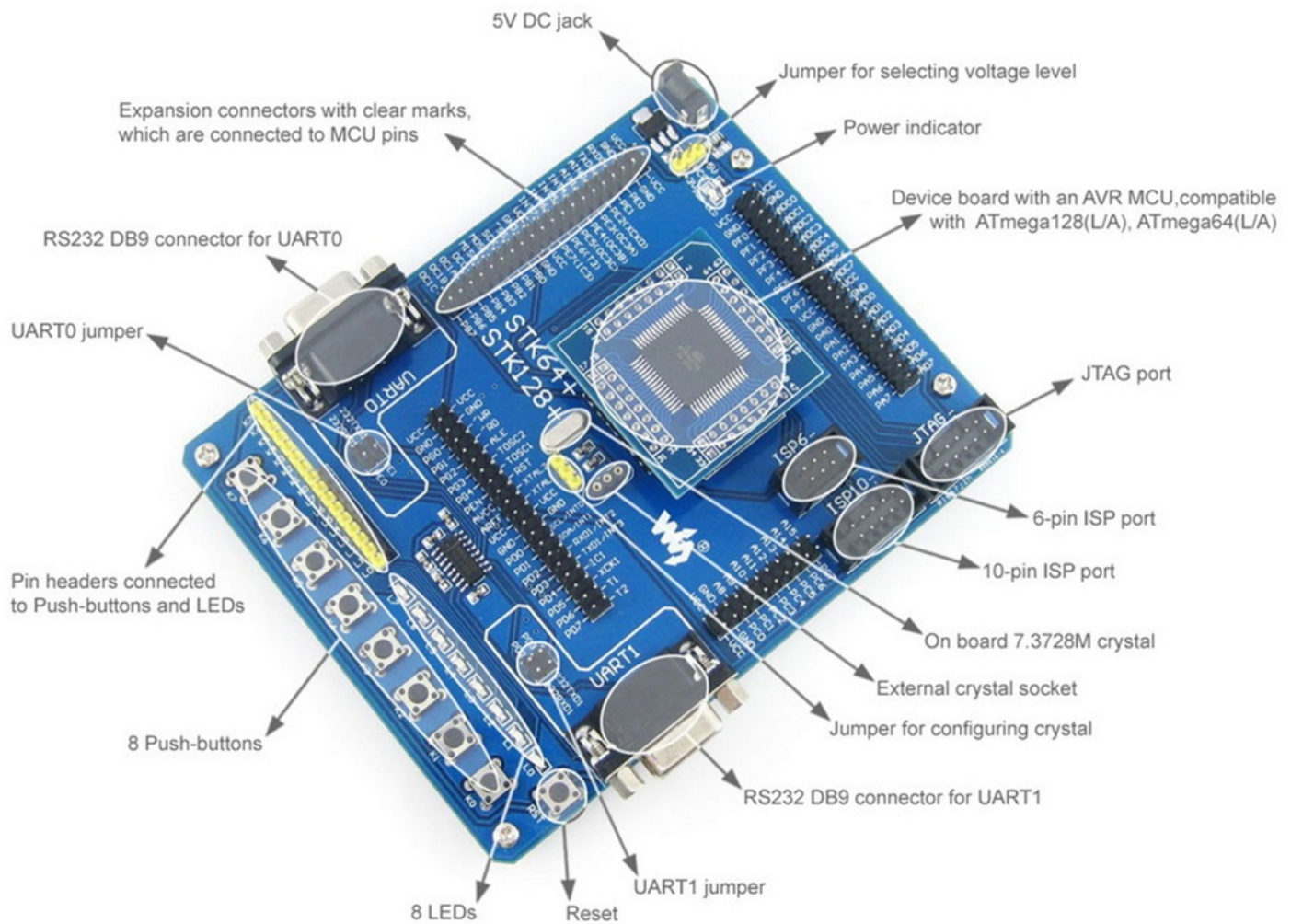
What is arduino ?



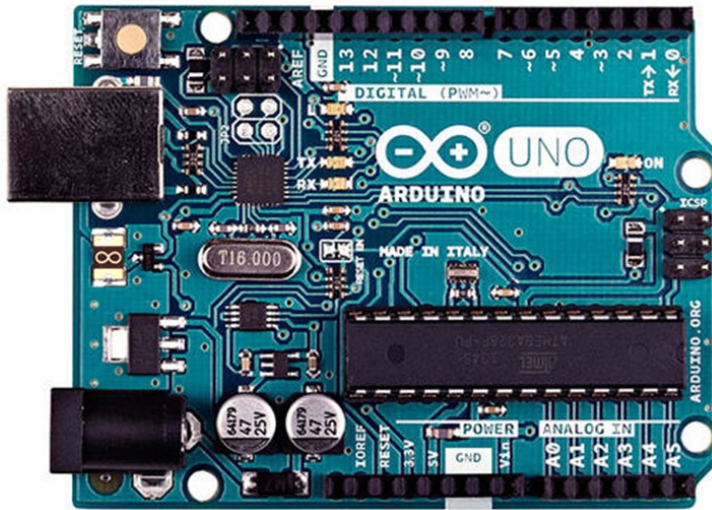
"Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Learn how to write code and upload it to your board in minutes."

“Wiring “ โครงการพัฒนา Open Souceใช้ AVR

เบอร์ ATmega 128



Arduino uno



ARDUINO MICROCONTROLLER

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog I/O Pins	6
EEPROM	1 KB
DC Current per I/O Pins	40 mA on I/O Pins; 50 mA on 3,3 V Pin



GENERAL

Input Voltage	7-12 V
Digital I/O Pins	20 (of which 6 provide PWM output)
PWM Output	6
PCB Size	53.4 x 68.6 mm
Weight	25 g
Product Code	A000066 (TH); A000073 (SMD)

ที่มา: <http://www.arduino.org/products/boards/arduino-uno>

จุดเด่นของ arduino ?

- ราคาถูก
- โปรแกรมพัฒนาใช้ได้กับ Windows, Mac, Linux
- คำนึงง่ายต่อการใช้
- ใช้งานได้จริง
- พัฒนาต่อยอดได้ ไม่มีลิขสิทธิ์

Arduino BOARD

Arduino INDUSTRIAL 101



Arduino YUN



Arduino UNO



Arduino YUN MINI



Arduino LEONARDO ETH



Arduino M0



Arduino BOARD

Arduino M0 PRO



Arduino MEGA 2560



Arduino LEONARDO



Arduino DUE



Arduino MICRO



Arduino ETHERNET



Arduino ROBOT



Arduino NANO



Arduino MEGA ADK



Arduino MINI 05



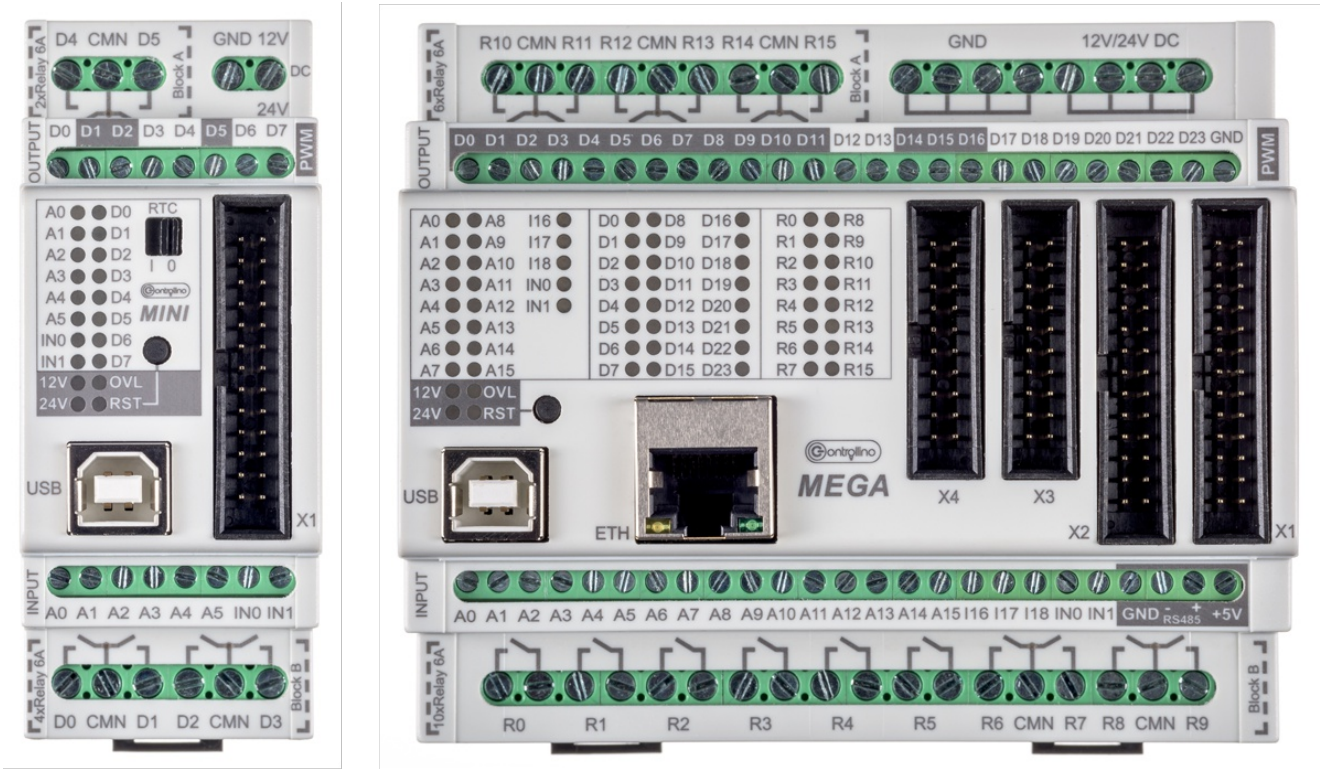
Arduino ESPLORA



SIMATIC IOT2020



CONTROLLINO



รู้จัก ESP8266

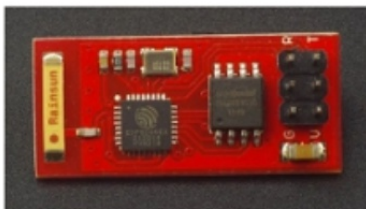
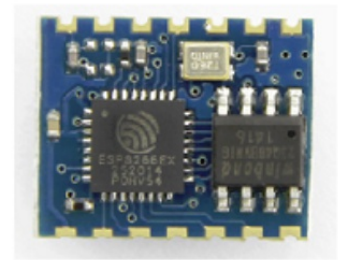
ชิปวายฟายและไมโครคอนโทรลเลอร์ราคาถูก



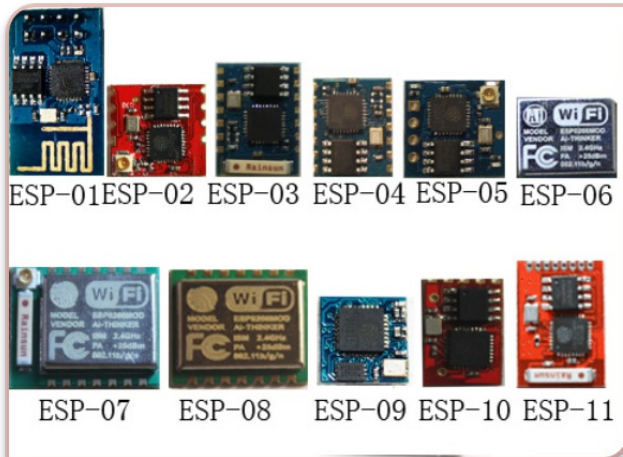
What is ESP8266 ?

“The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (microcontroller unit) capability.”

รู้จัก ESP8266

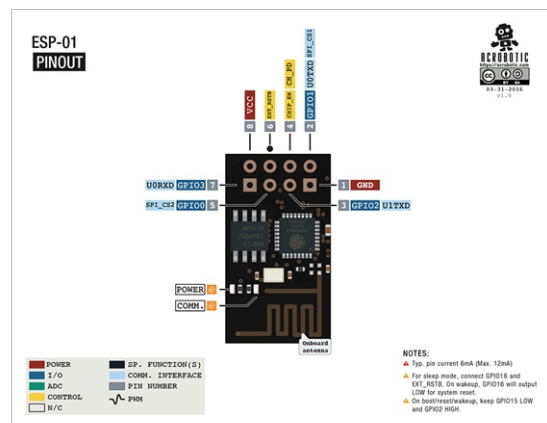
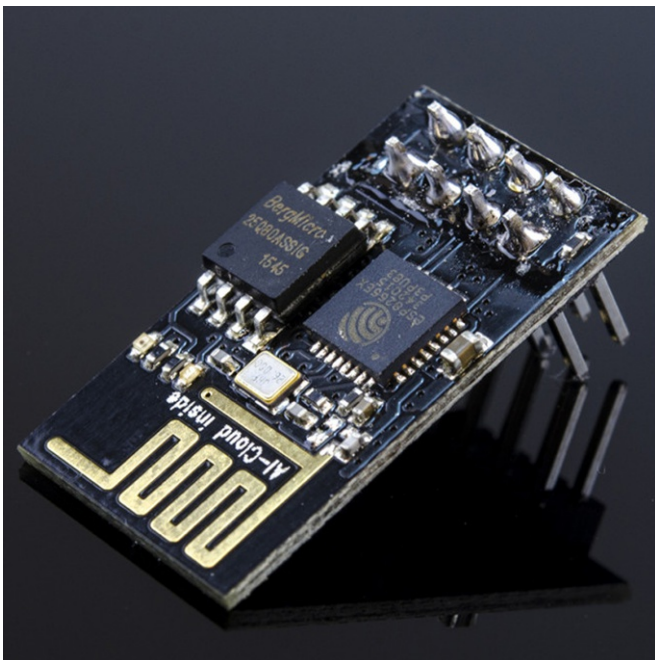


รู้จัก ESP8266

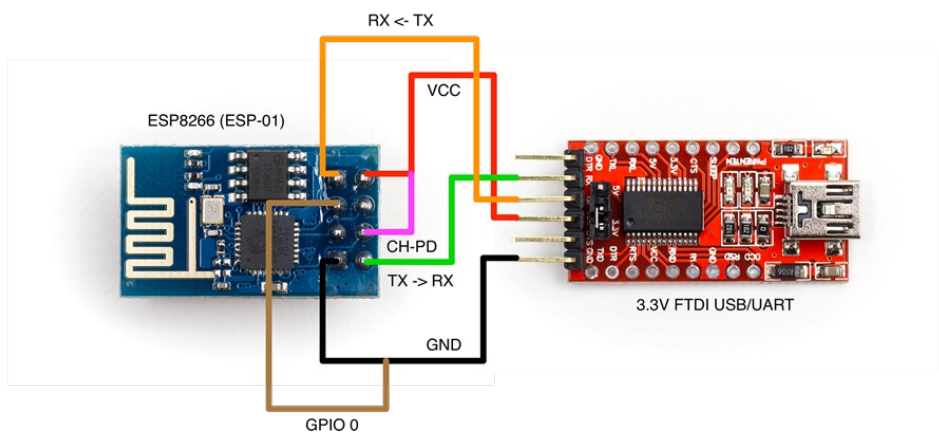
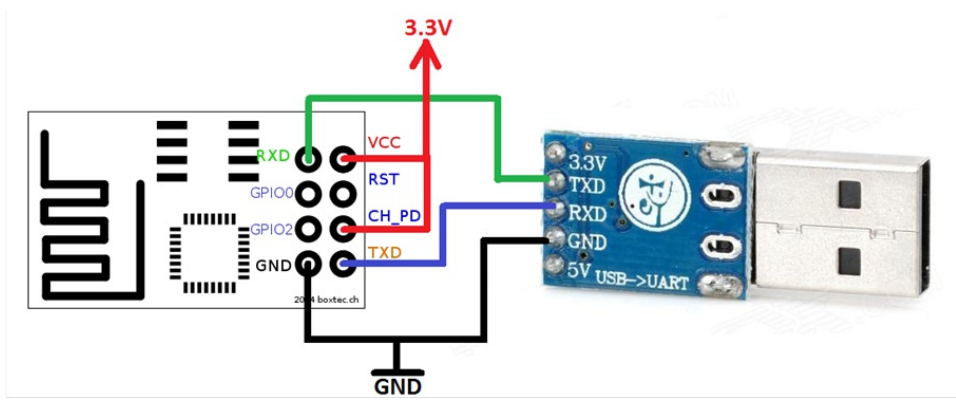


- ESP-01 through the use of professional equipment matching PCB antenna, the radiation effect is excellent, 8PIN-pin serial port and upgrade leads to the mouth, and a reset
- ESP-02 external antenna can penetrate the metal shield, better external antenna, the proposed standard with 50-ohm antenna
- ESP-03 leads all available IO ports, and using high-gain antennas shrink ceramics, suitable for development with the SDK friend
- ESP-04 leads all the IO ports, antenna customers can design their own freedom, more flexible
- ESP-05 interface only leads to ease of use UART and VCC
- ESP-06: bottom mount technology, leads all the IO ports, with metal shielding shell
- ESP-07: Semi-hole chip technology, all the IO leads, with metal shielding shell
- ESP-08: same as ESP-07, except that the antenna is in the form of customers can define their own
- ESP-09: Ultra-small size package, only 10 * 10 mm, four-layer board technology 1M bytes!
- ESP-10: SMD interface, narrow-body design, 10 mm wide, suitable for light with controller
- ESP-11: SMD interface, ceramic antenna, small size

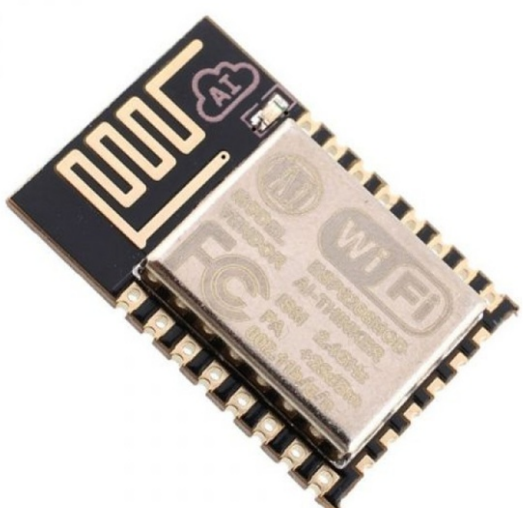
โมดูล ESP-01



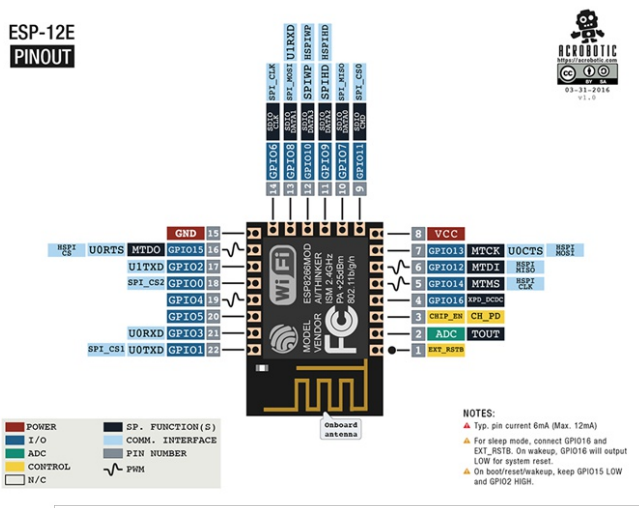
การเชื่อมต่อโมดูล ESP-01 กับ USB



โมดูล ESP-12E ที่ได้รับความนิยม



ESP-12E
PINOUT



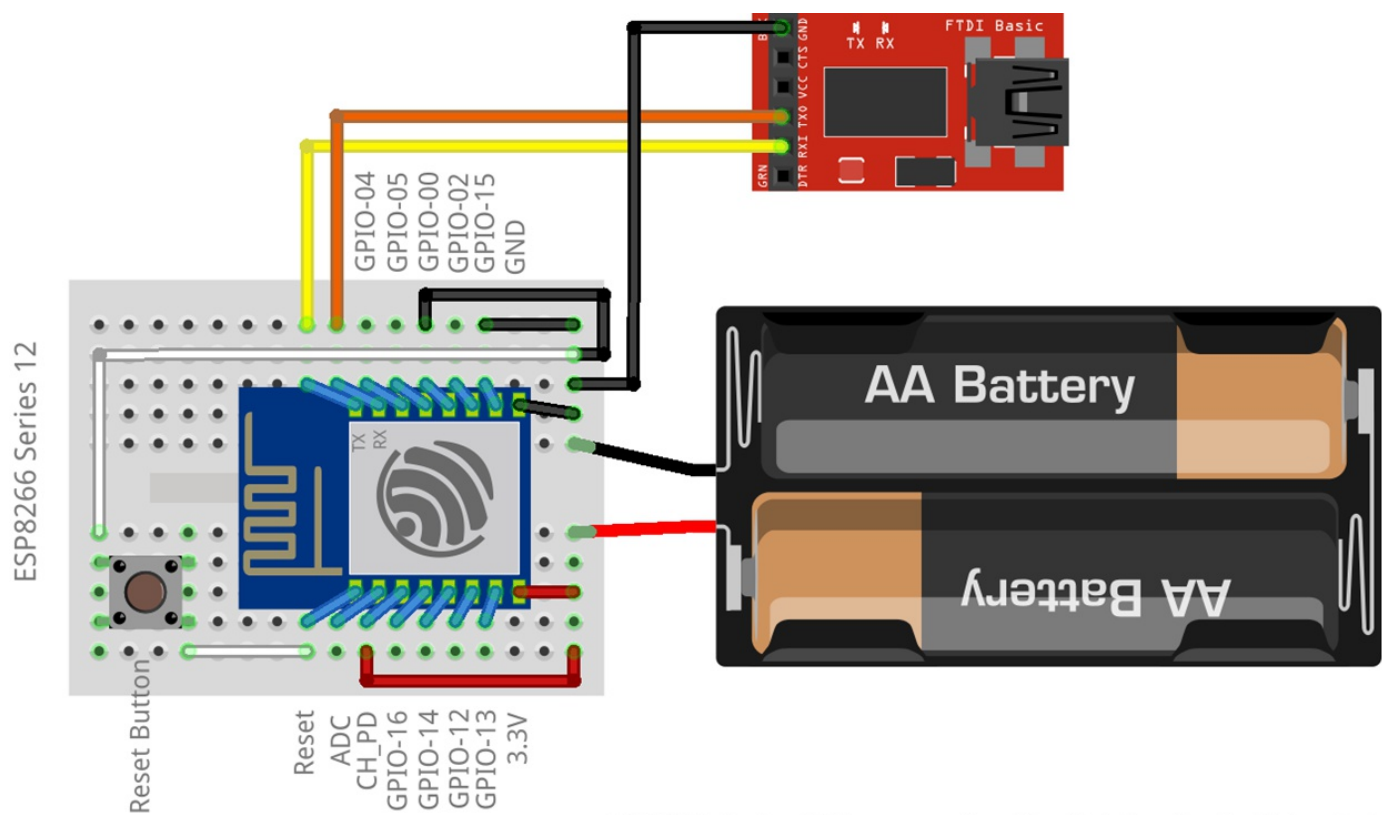
ESP-12E PINOUT diagram showing pin numbers 1-26 and their functions. The diagram includes a legend for POWER (red), I/O (blue), ADC (green), CONTROL (yellow), and N/C (white). It also includes a legend for GP_ FUNCTION (S) (black), COM. INTERFACE (light blue), PIN NUMBER (dark blue), and PIN (checkmark). The diagram shows the following connections:

- 1: GND
- 2: VCC
- 3: TXD
- 4: RXD
- 5: GND
- 6: TXD
- 7: RXD
- 8: GND
- 9: TXD
- 10: RXD
- 11: GND
- 12: TXD
- 13: RXD
- 14: GND
- 15: TXD
- 16: RXD
- 17: GND
- 18: TXD
- 19: RXD
- 20: GND
- 21: TXD
- 22: RXD
- 23: GND
- 24: TXD
- 25: RXD
- 26: GND

NOTES:

- Typ. pin current 6mA (Max. 12mA)
- For sleep mode, connect GPIO16 and EXT_RSTB. On wakeup, GPIO16 will output LOW for system reset.
- On boot/reset/lookup, keep GPIO15 LOW and GPIO2 HIGH.

การเชื่อมต่อโมดูล ESP-12E กับ USB



ESP8266 Series 12 Programming Circuit (Gareth aka Chiprobot)

รู้จัก NodeMCU

แพลตฟอร์มสำหรับพัฒนา Internet of Things (IoT)



What is NODEMCU ?



Connect Things EASY

“An open-source firmware and development kit that helps you to prototype your IOT product within a few Lua script lines.”

NodeMCU



NodeMCU คืออะไร ?

NodeMCU คือ แพลตฟอร์มที่ใช้ในการพัฒนา Internet of Things (IoT) ตัวบอร์ดได้รวมโมดูล ESP8266, USB-to-TTL และ พอร์ต micro USB เข้าด้วยกัน สามารถเขียนโปรแกรมด้วยภาษา Lau และภาษา C/C++ บน Arduino IDE ได้

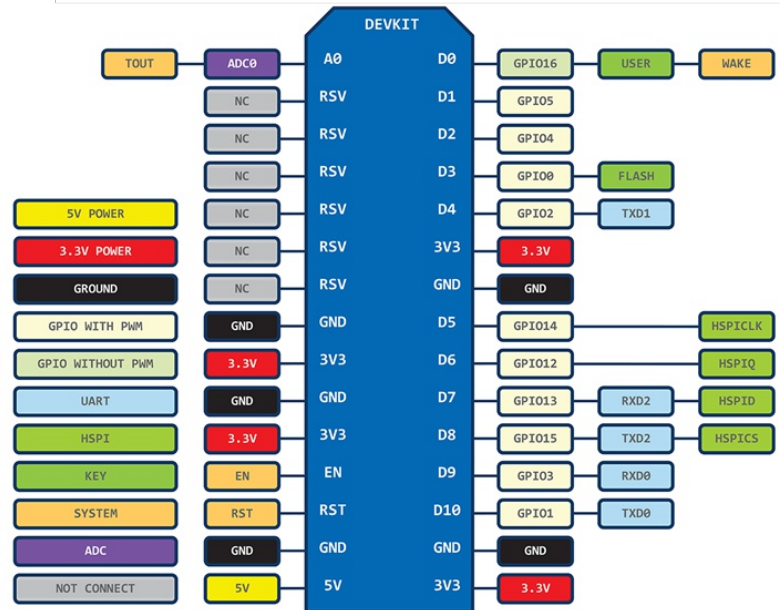
NodeMCU คือ แพลตฟอร์มหนึ่งที่ใช้ช่วยในการสร้างโปรเจกต์ Internet of Things (IoT) ที่ประกอบไปด้วย Development Kit (ตัวบอร์ด) และ Firmware (Software บนบอร์ด) ที่เป็น open source สามารถเขียนโปรแกรมด้วยภาษา Lau ได้ ทำให้ใช้งานได้ง่ายขึ้น มาพร้อมกับโมดูล WiFi (ESP8266) ซึ่งเป็นหัวใจสำคัญในการใช้เชื่อมต่อกับอินเทอร์เน็ตนั่นเอง ตัวโมดูลESP8266นั้นมีอยู่ด้วยกันหลายรุ่นมาก ตั้งแต่เวอร์ชันแรกที่เป็น ESP-01 ไล่ไปเรื่อยๆจนปัจจุบันมีถึงESP-12 แล้ว และที่ฝังอยู่ใน NodeMCU version แรกนั้นก็จะเป็น ESP-12 แต่ใน version2 นั้นจะใช้เป็น ESP-12E แทน ซึ่งการใช้งานโดยรวมก็ไม่แตกต่างกันมากนัก NodeMCU นั้นมีลักษณะคล้ายกับ Arduino ตรงที่มีพอร์ต Input Output build ในตัว สามารถเขียนโปรแกรมคอนโทรลอุปกรณ์ I/O ได้โดยไม่ต้องผ่านอุปกรณ์อื่นๆ และเมื่อไม่นานมานี้ก็มีนักพัฒนาที่สามารถทำให้ Arduino IDE ใช้งานร่วมกับ Node MCU ได้ จึงทำให้ใช้ภาษา C/C++ ในการเขียนโปรแกรมได้ ทำให้เราสามารถใช้งานมันได้หลากหลายมากยิ่งขึ้น NodeMCUตัวนี้สามารถทำอะไรได้หลายอย่างมากโดยเฉพาะเรื่องที่เกี่ยวข้องกับ IoT ไม่ว่าจะเป็นการทำ Web Server ขนาดเล็ก การควบคุมการเปิดปิดไฟผ่านWiFi และอื่นๆอีกมากมาย ซึ่งจะสอนในบทความต่อไป

NodeMCU

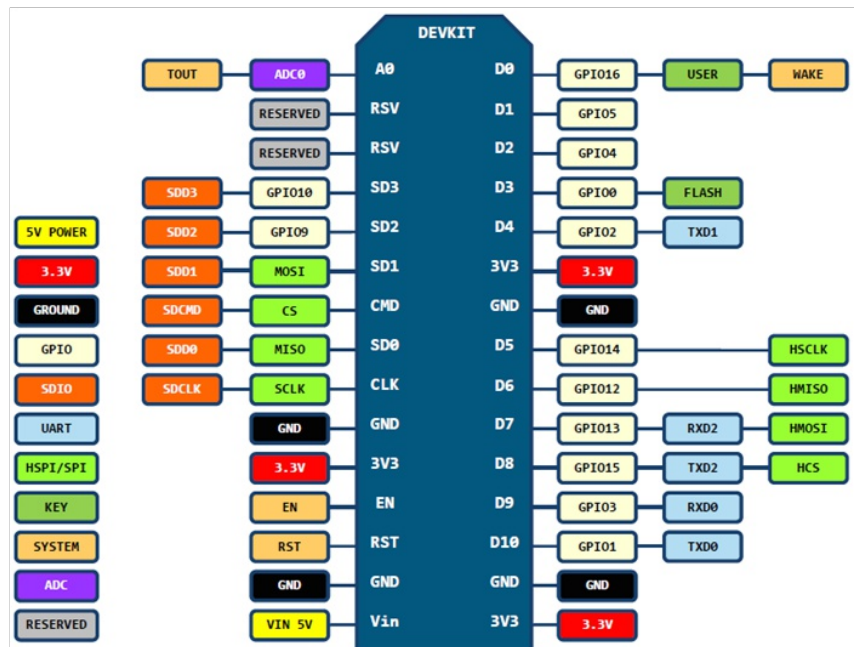
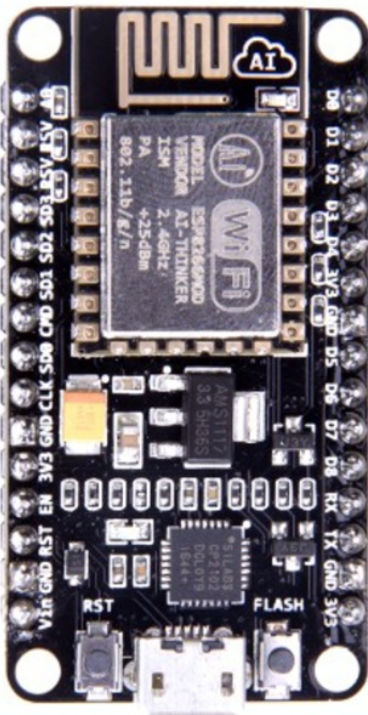


Model	NodeMCU Version 3
Type	Microcontroller
CPU	@ 80 MHz (default) or 160 MHz
Memory	64 KiB instruction, 96 KiB data
Input	16 GPIO pins
Power	3.3 VDC

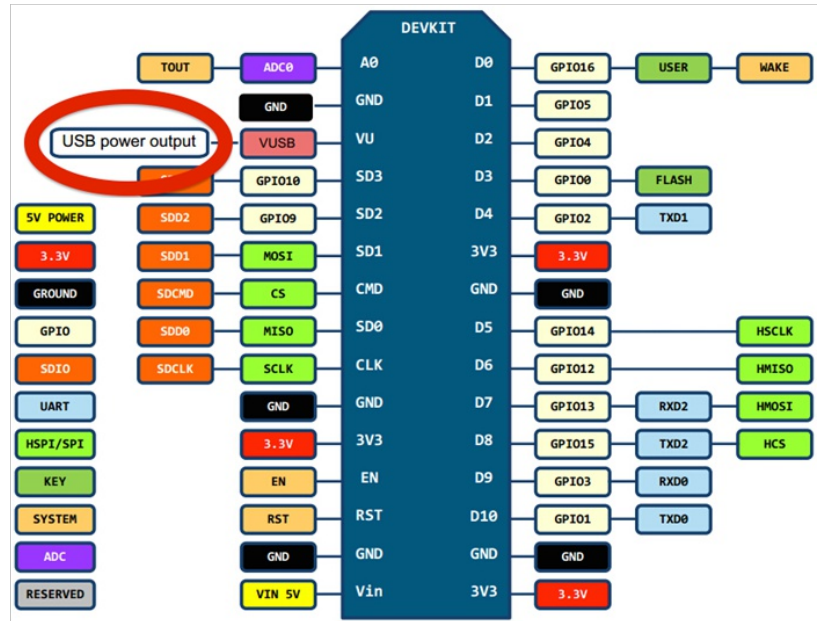
NodeMCU Version 1



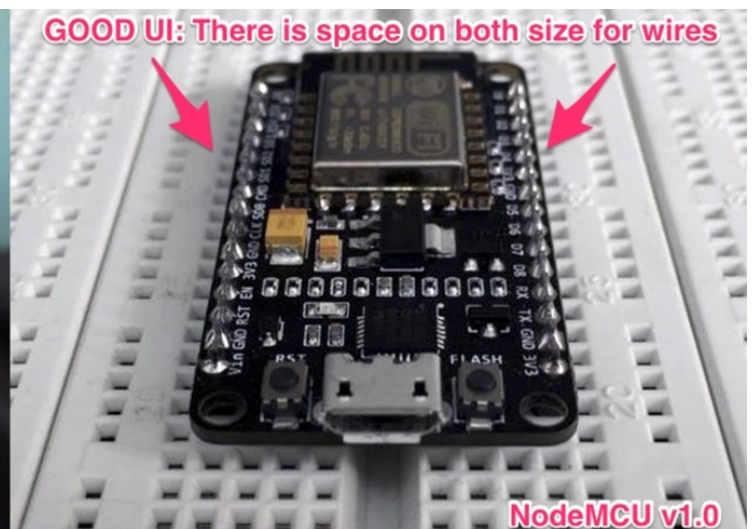
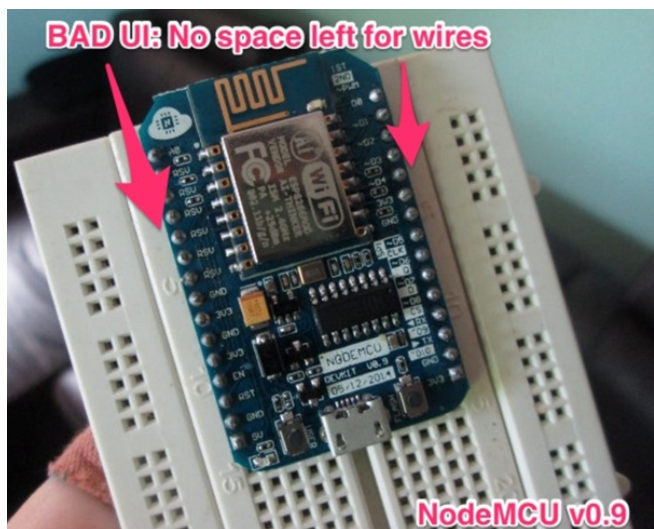
NodeMCU Version 2



NodeMCU Version 3

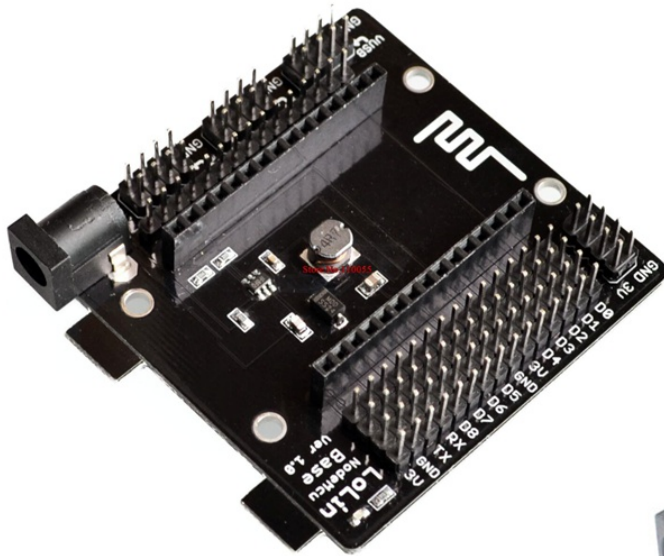


Compare NodeMCU size

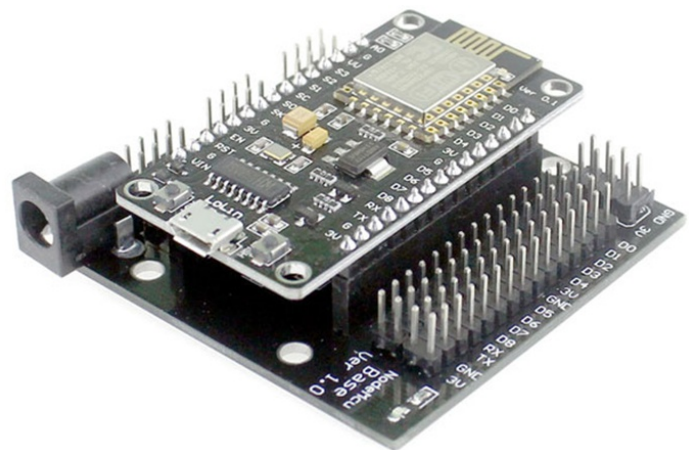


Version 1 และ Version 3 จะไม่มีพื้นที่เหลือด้านข้างบนบอร์ดทดลอง

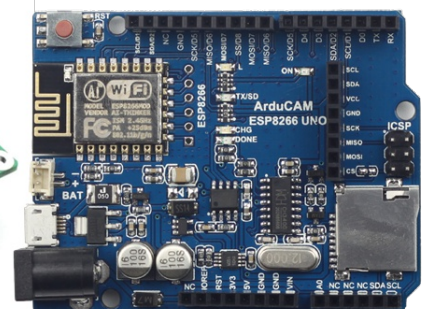
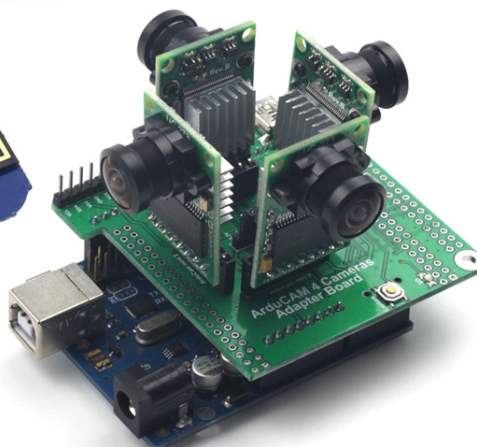
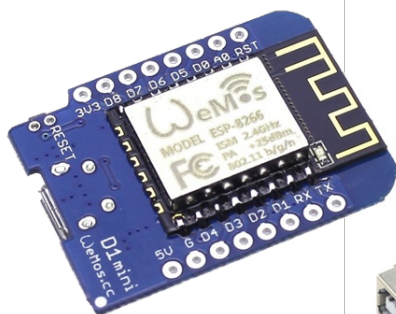
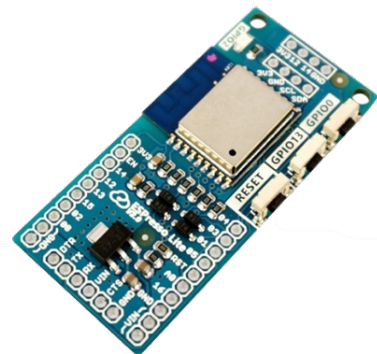
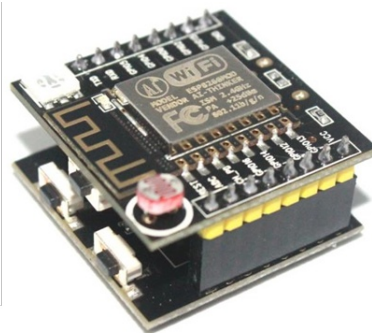
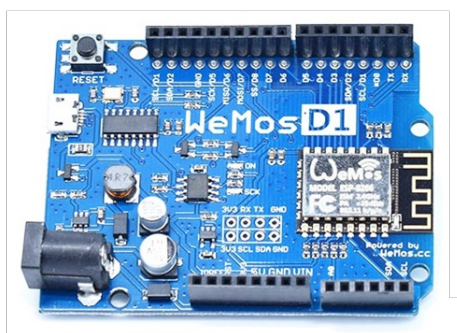
Nodemcu v3 module



ขยายขาให้เชื่อมต่อได้ง่ายขึ้น และรองรับการจ่ายไฟ 6-24 VDC ได้



ESP8266 BOARD



แนะนำ ESP32

รู้จัก ESP32 และเปรียบเทียบกับ ESP8266



ESP32

- แนะนำ ESP32
- ซอฟต์แวร์ที่ใช้พัฒนา
- ESP32 โมดูล
- บอร์ดพัฒนา ESP32
- เปรียบเทียบระหว่าง ESP8266 และ ESP32
- การติดตั้ง ESP32 ใน Arduino IDE
- การเสียบบอร์ด ESP32 ใน Arduino ide
- ตัวอย่างการเขียนโค้ด
- ทดสอบการอัปโหลด sketch

แนะนำ ESP32



ESP32 คือ

ESP32 เป็นชื่อของไอซีไมโครคอนโทรลเลอร์ที่รองรับการเชื่อมต่อ WiFi มีความสามารถในการเชื่อมต่อ Bluetooth Low-Energy (BLE, BT4.0, Bluetooth Smart) ผลิตโดยบริษัท Espressif จากประเทศจีน

ESP32 ได้แก้ไขจุดด้อยต่างๆของ esp8266 ไปจนหมด ไม่ว่าจะเป็นเรื่องของ I/O และ Analog input ที่มีไม่เพียงพอกับการใช้งาน และ ปรับสเปคของ hardware ให้สูงขึ้น มีความเสถียรภาพสูง

การพัฒนา ESP32

หลังจากที่บริษัท Espressif ได้ออกไอซี ESP8266 และได้รับความนิยมสูงสุด ก็ได้ออกไอซีรุ่นใหม่ชื่อว่า ESP31B มีการพัฒนาชุดซอฟต์แวร์ ESP32_RTOS_SDK ไปพร้อมกัน แต่หลังจากนั้นไม่นาน บริษัท Espressif ได้ยกเลิกการใช้ชุดซอฟต์แวร์พัฒนาดังกล่าว แล้วไปสร้างชุดพัฒนาใหม่ชื่อ ESP-IDF แล้วได้ออกไอซี ESP32 เป็นครั้งแรก

บริษัท Espressif ได้ผลิตโมดูล ESP-WROOM-32 ออกมา จากนั้นบริษัท Ai-Thinker และ Seeedstudio ก็ได้ผลิตโมดูล ESP3212 ขึ้นมา แต่ด้วยปัญหาด้านการออกแบบจึงได้ยกเลิกการผลิตแล้วหันไปผลิต ESP32S แทน โดยมีลายวงจรเหมือนกับ ESP-WROOM-32 ทุกอย่าง

การพัฒนา ESP32

บริษัท Espressif ได้ทีมผู้พัฒนา Arduino core for ESP8266 WiFi chip และทีมผู้พัฒนา Arduino core for ESP31B WiFi chip มาร่วมงานช่วยพัฒนาชุดไลบรารีและคอมไพเลอร์สำหรับ Arduino ทำให้การพัฒนาเป็นไปได้อย่างรวดเร็ว

ESP-IDF ไม่มีโปรแกรม Editor แต่เป็นแกนพัฒนาหลัก เมื่อมีการเพิ่มฟีเจอร์ใหม่ๆ ถึงจะเพิ่มใน Arduino core for ESP32 WiFi chip ปัจจุบันเราสามารถใช้ Arduino IDE พัฒนา ESP32 ได้

Key Features

- 240 MHz dual core Tensilica LX6 microcontroller with 600 DMIPS
- Integrated 520 KB SRAM
- Integrated 802.11 b/g/n HT40 Wi-Fi transceiver, baseband, stack and LwIP
- Integrated dual mode Bluetooth (classic and BLE)
- 16 MB flash, memory-mapped to the CPU code space
- 2.3V to 3.6V operating voltage
- -40°C to +125°C operating temperature
- On-board PCB antenna / IPEX connector for external antenna

Security Related

- WEP, WPA/WPA2 PSK/Enterprise
- Hardware-accelerated encryption: AES/SHA2/Elliptical
- Curve Cryptography/RSA-4096

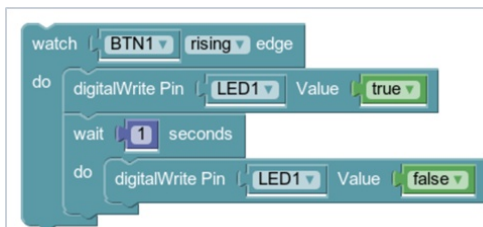
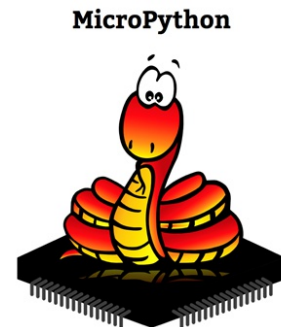
Performance

- Max data rate of 150 Mbps@11n HT40, 72 Mbps@11n HT20, 54
- Maximum transmit power of 19.5 dBm@11b, 16.5 dBm@11g, 15.5 dBm@11n
- Minimum receiver sensitivity of -97 dBm
- 5 μ A power consumption in Deep-sleep

ซอฟต์แวร์ที่ใช้พัฒนา



Software Development



The Graphical Editor lets you use Espruino even if you can't code!

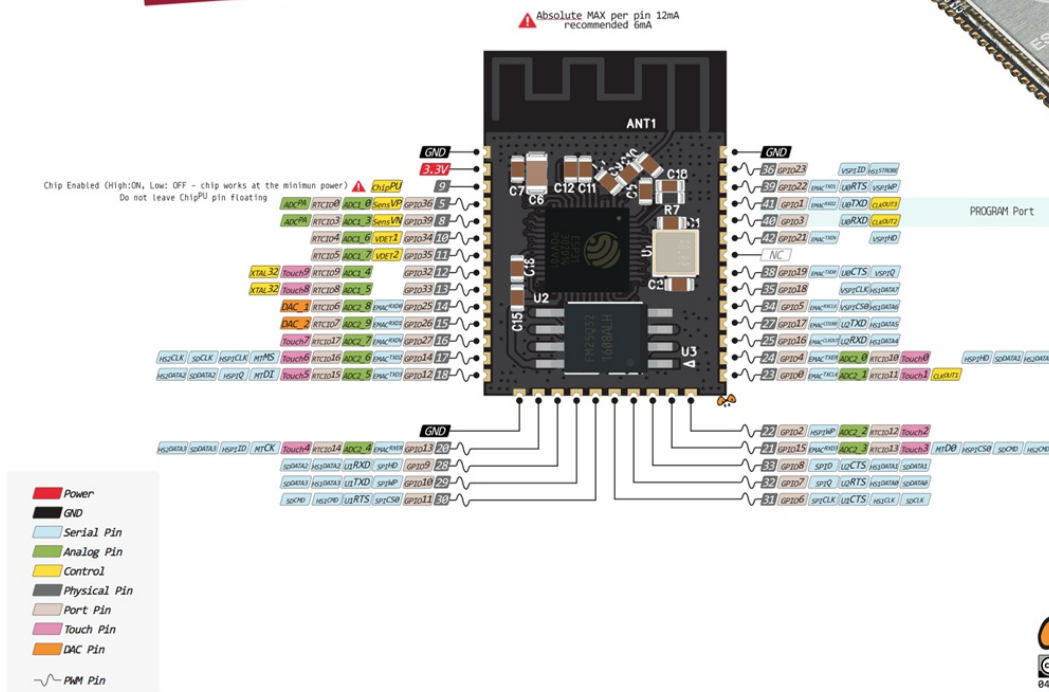
The optional graphical editor makes it quick and easy to create simple devices, even if you've never programmed before.

MicroPython-ESP32 รองรับการใช้งานพื้นฐานภาษา Python ส่วนใหญ่ได้ รองรับการจัดการ WiFi การเชื่อมต่อต่างๆ ของ ESP32 ส่วน Espruino on ESP32 ใช้ภาษา JavaScript ในการสั่งงานและรองรับการเขียนโปรแกรมแบบ Text และภาษابل็อก (Block) และ LuaNod ก็รองรับคำสั่งที่ใช้บน Lua จริง ๆ แทบทุกคำสั่ง และรองรับการควบคุม WiFi ได้เต็มรูปแบบ

ESP32 โมดูล

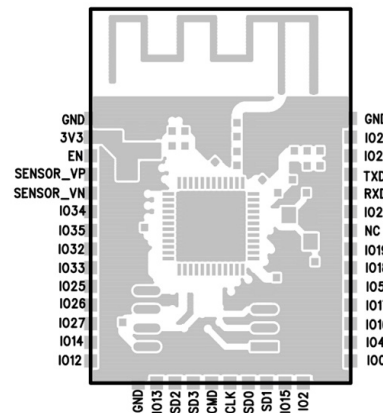
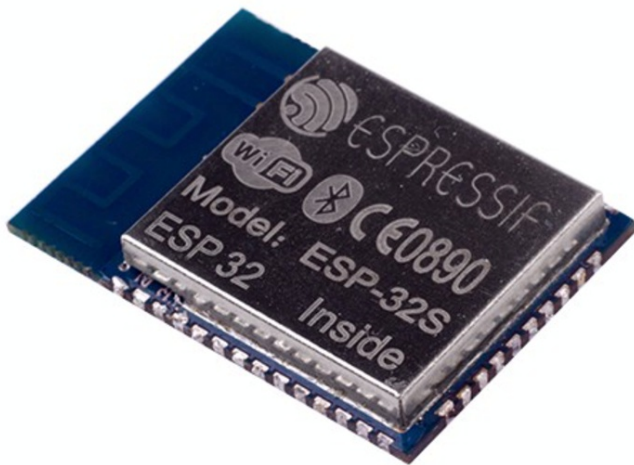
ESP32 Module – WROOM-32

WROOM32
PINOUT



ผลิตโดยบริษัท Espressif (บริษัทผู้ผลิตชิปไอซี ESP32) มีอยู่ด้วยกัน 2 เวอร์ชัน เวอร์ชันแรกจะแสดงเฉพาะโลโก้ของบริษัท Espressif และมีคำว่า ESP-WROOM-32 ส่วนเวอร์ชันที่สอง เกิดจากการที่ชิปได้ผ่านการทดสอบได้ใบรับรองจากมาตรฐานต่าง จึงได้ออกรุ่นที่แสดงหมายเลขใบรับรองบนชิปออกมา และได้ใช้จนมาถึงรุ่นปัจจุบัน ตัวโมดูลมาพร้อมกับเสาอากาศแบบ PCB บนตัว มีรอบคุ่มอุปกรณ์ทั้งหมดบนโมดูลเพื่อลดสัญญาณรบกวนลง ซึ่งจะทำให้การทำงานโดยรวมมีเสถียรภาพมากยิ่งขึ้น ชิป ESP-WROOM-32 นี้ได้รับความนิยมเป็นอย่างมากในการใช้งานในด้านต่างๆ เนื่องจากสามารถผลิตได้ทันความต้องการและมีราคาถูกกว่าโมดูลที่ผลิตโดยผู้ผลิตอื่น ๆ มาก นอกจากนี้ ตำแหน่งขาต่างๆ ของ ESP-WROOM-32 ยังถือเป็นมาตรฐานที่โมดูลจากผู้ผลิตอื่นต้องทำตามอีกด้วย

ESP32 Module – ESP-32S



ESP-32S คือบอร์ดโมดูล (ESP32 chip + PCB board + cover box) ของบริษัท Ai-Thinker โดยมีขนาดใช้งานและขนาดของตัวถัง ลายวงจรต่างๆ เหมือนกับ ESP-WROOM-32 ของ Espressif และสามารถใช้ทดแทนกันได้ แต่ทั้งนี้ราคาของ ESP32S แพงกว่าโมดูล ESP-WROOM-32 และ ESP32-Bit ทำให้ไม่ค่อย

EESP32 Module – ESP32-Bit



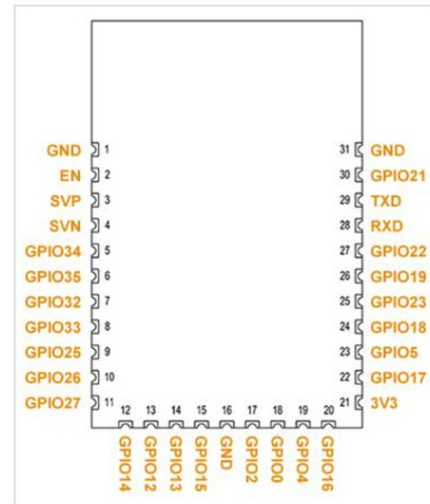
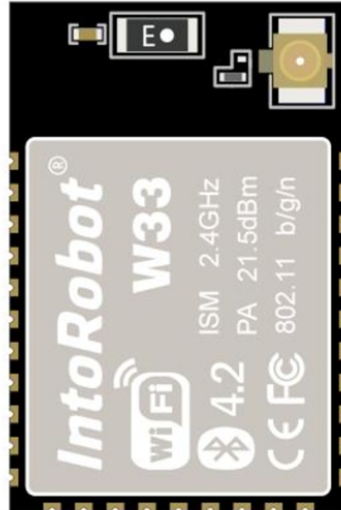
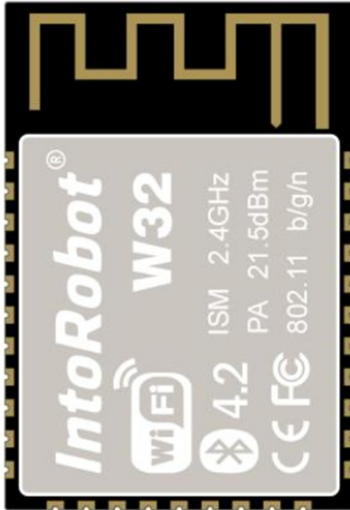
ESP32-Bit		
1	GND	VCC3V3
2	Sens_VP	GND
3	Sens_VN	GPIO21
4	CHIP_UP	TXD0
5	GPIO14	RXD0
6	GPIO15	GPIO22
7	GPIO32	GPIO19
8	GPIO33	GPIO23
9	GPIO25	GPIO18
10	GPIO26	GPIO5
11	GPIO27	GPIO17
12	GPIO14	GPIO16
13	GPIO12	GPIO4
14	GPIO13	GPIO0/DL
15	GPIO15	GPIO2
30		
29		
28		
27		
26		
25		
24		
23		
22		
21		
20		
19		
18		
17		
16		

ESP32-Bit

บริษัทผลิตบอร์ดไมโครคอมพิวเตอร์ในจีน eBox และ Widora ได้พัฒนาโมดูล ESP32-Bit ข้อดีคือมีขนาดเล็กกว่า ESP-WROOM-32 ทำให้ช่วยลดขนาดของบอร์ดพัฒนาลง สามารถเลือกใช้เสาอากาศภายนอกโดยใช้คอนเนคเตอร์แบบ IPEX หรือเลือกใช้เสาอากาศแบบเซรามิคบนตัวโมดูล ส่วนข้อเสียคือมีการตัดขาใช้งานบางส่วนออกไป ทำให้มีขาใช้งานน้อยลงโมดูล



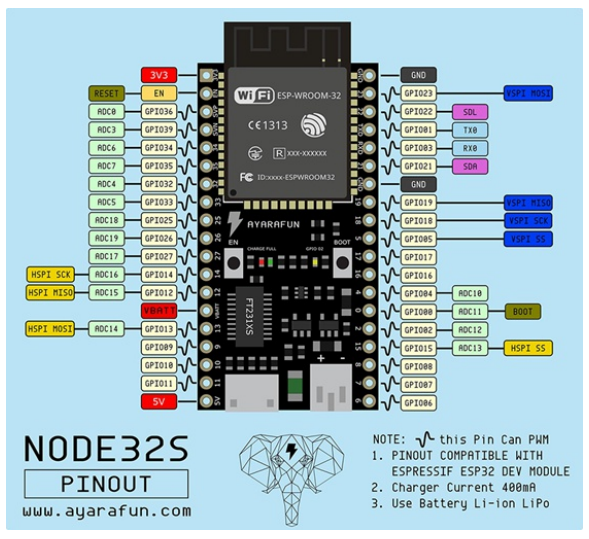
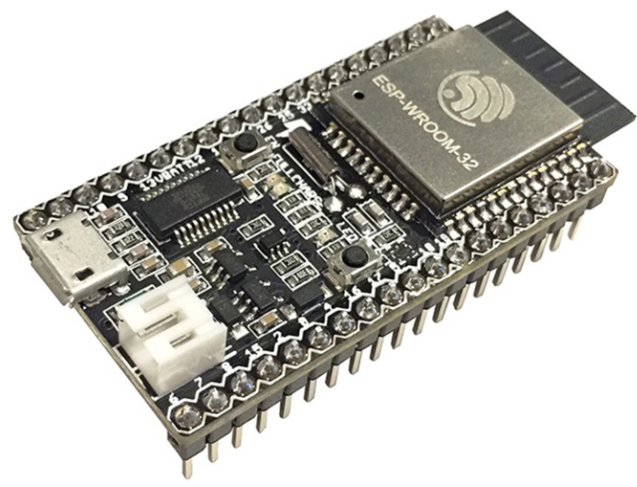
ESP32 Module – IntoRobot-W32/W33



IntoRobot

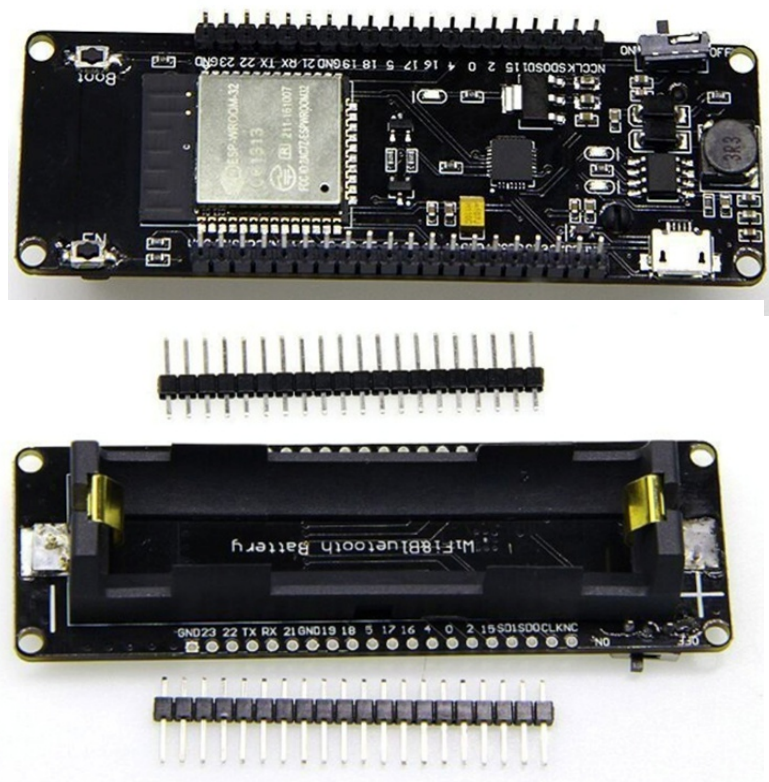
IntoRobot-W32/W33 ผลิตโดยบริษัท IntoRobot มีอยู่ด้วยกัน 2 รุ่น คือรุ่น W32 ที่ใช้เสาอากาศแบบ PCB ที่อยู่บนบอร์ด และรุ่น W33 ที่มีเสาแบบเซรามิกเป็นเสาหลักในการรับสัญญาณและสามารถเปลี่ยนไปใช้คอนเนคเตอร์แบบ IPEX สำหรับต่อเสาอากาศภายนอกได้ ตำแหน่งขาและขนาดของโมดูลจะมีขาต่อใช้งานทั้งหมด 31 ขา น้อยกว่า ESP-WROOM-32 / ESP32S แต่มีขามากกว่าโมดูล ESP32-Bit มีขนาดเล็กกว่า ESP-WROOM-32/ESP32S แต่มีขนาดใหญ่กว่าโมดูล ESP32-Bit

บอร์ดพัฒนา ESP32



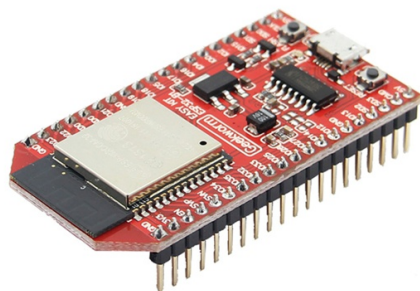
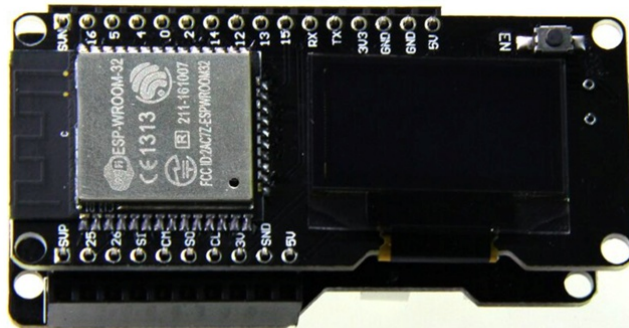
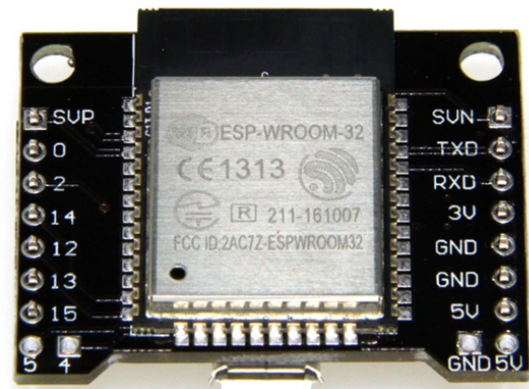
Ayarafun/LamLoei Node32S

"ภูมิใจไทยออกแบบ ภูมิใจไทยผลิต สนับสนุนผลงานไทย ซื้อได้อย่างภาคภูมิใจ"

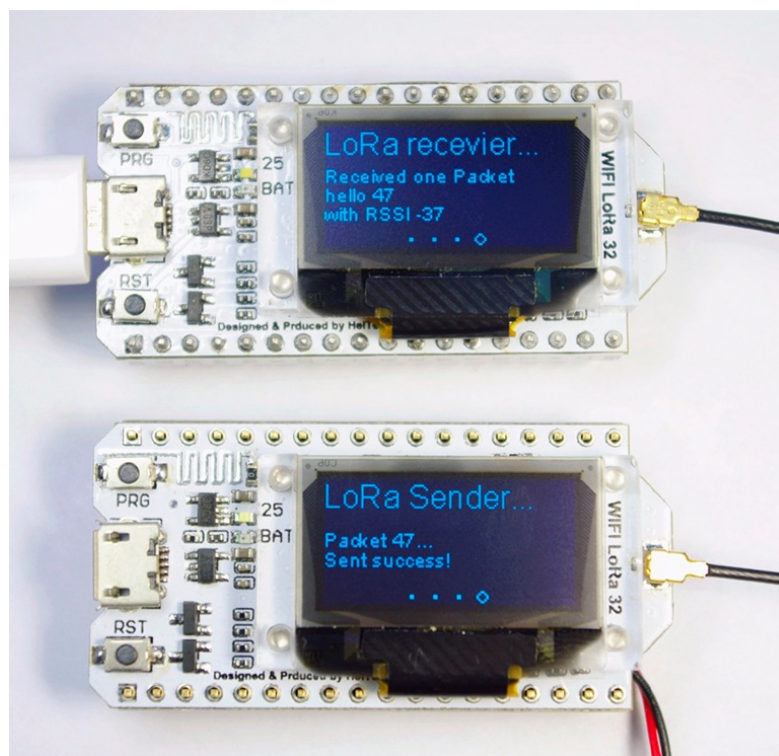
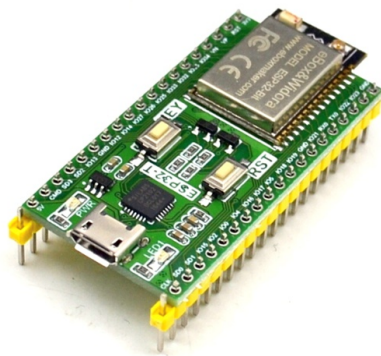
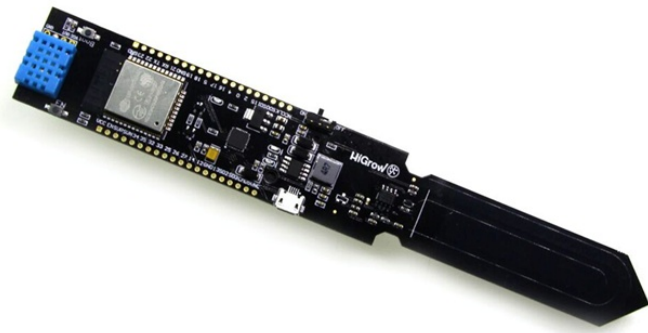


WeMos® WiFi + Bluetooth Battery ESP32 Development Tool

บอร์ดพัฒนา ESP32

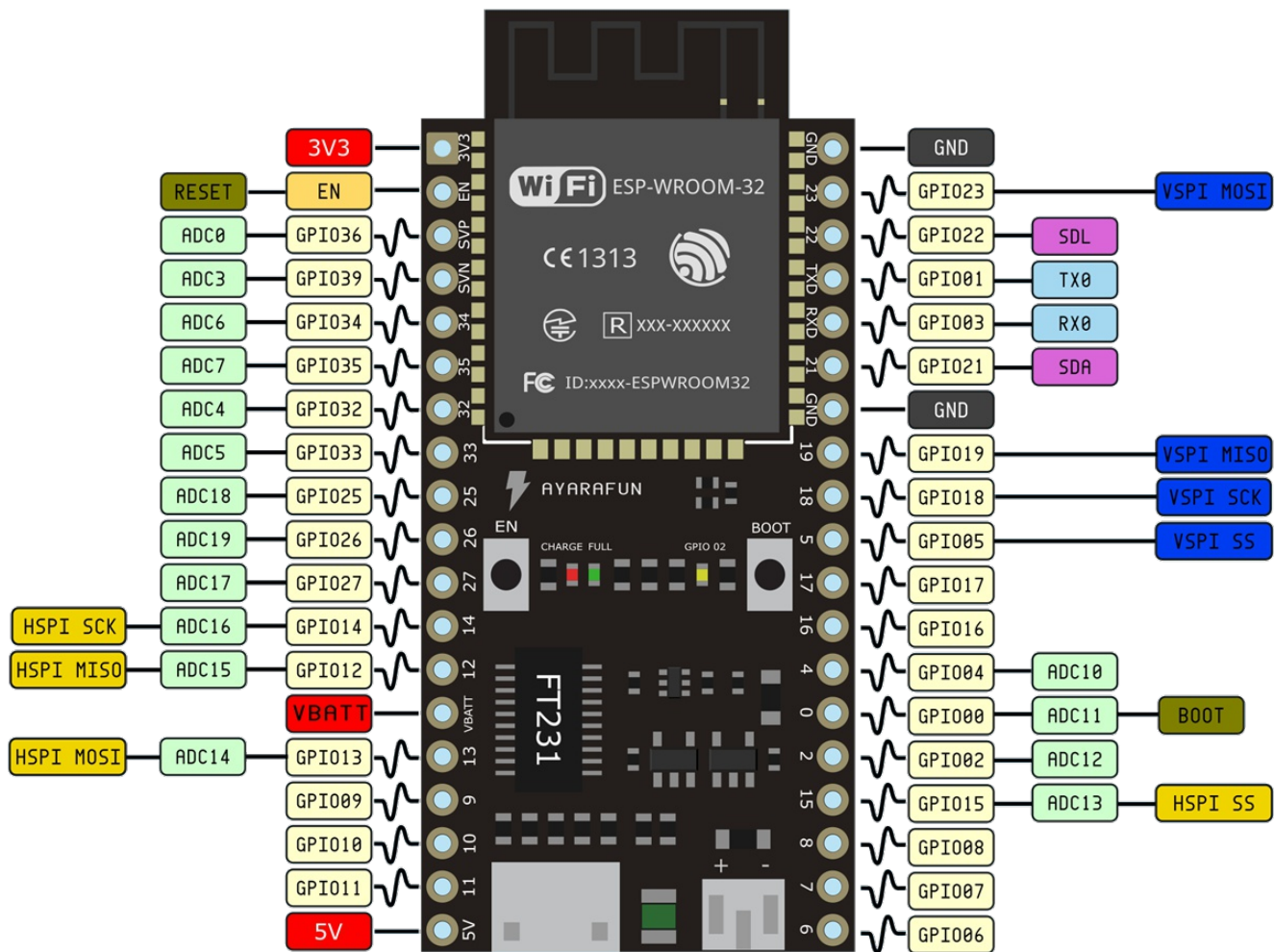


บอร์ดพัฒนา ESP32



WiFi LoRa 32

บอร์ดพัฒนา ESP32



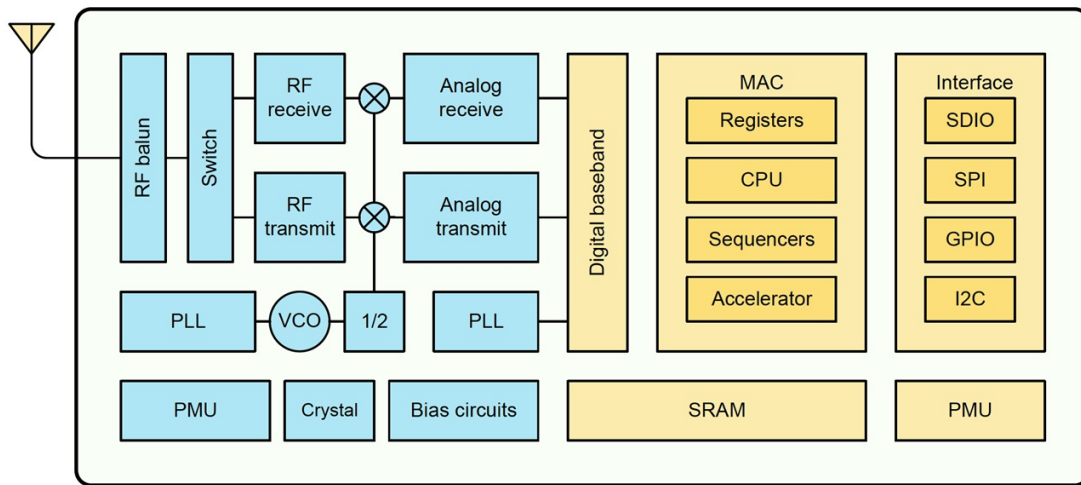
เปรียบเทียบระหว่าง ESP8266 และ ESP32

	ESP8266	ESP32
	Tensilica L106 32-bit single core micro controller	Xtensa® Single-/Dual-core 32-bit LX6 microprocessor(s)
ROM	none	448 KB
RAM	160kB	520 KB
RTC RAM	768 Bytes	8kB slow + 8kB fast
QSPI flash/SRAM,	up to 1 x 16 MB	up to 4 x 16 MB
GPIOs	17	36
Wi-Fi	802.11 b/g/n/e/i	802.11 b/g/n/e/i
Bluetooth	no	yes
PWM Pins	8 Software	1 Hardware/16 software
ADC	10 bit	12 bit
DAC	no	2 x 8 bit
Touch sensor	no	10
Temperature Sensor	no	yes
Cryptographic engine	no	yes
• Power supply:	2.5V to 3.6V	2.3V to 3.6V

IEEE 802.11e เป็นมาตรฐานที่ออกแบบมาสำหรับการใช้งาน แอปพลิเคชันทางด้านมัลติมีเดียอย่าง VoIP (Voice over IP)

IEEE 802.11i เป็นมาตรฐานในด้านการรักษาความปลอดภัย ของผลิตภัณฑ์เครือข่ายไร้สาย

ESP8266 Block Diagram



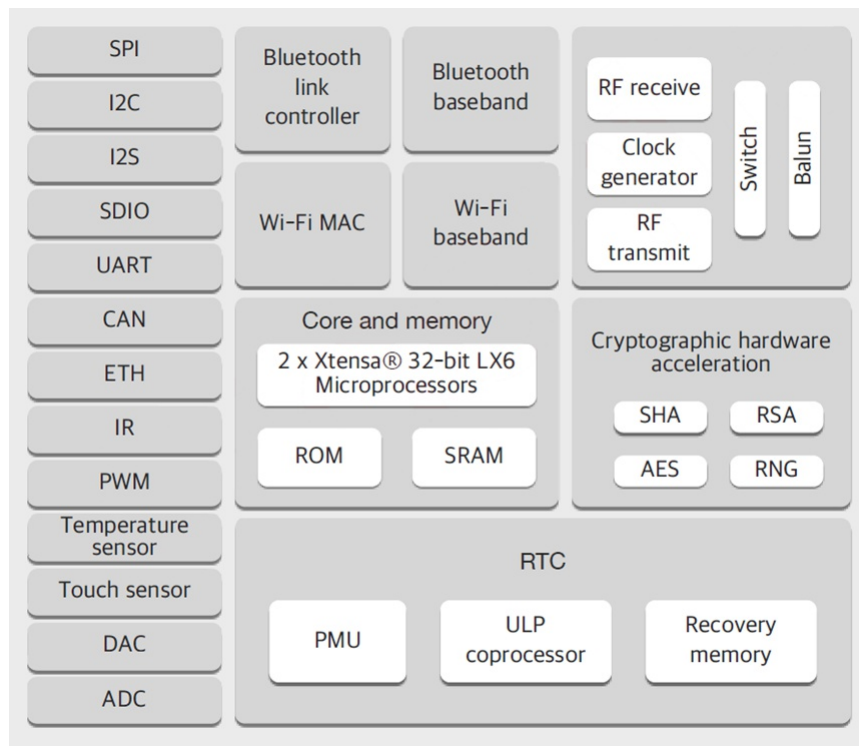
แบ่งได้เป็น 2 ส่วน คือฝั่งด้านซ้าย Radio frequency (RF) part มีหน้าที่เชื่อมต่อและจัดการกับ WiFi และฝั่งด้านขวาคือ Logical RF ที่ประกอบด้วย MCU, SRAM, Interface เชื่อมต่อแบบต่างๆ สถาปัตยกรรมนี้มีขาใช้งานจำนวน 32 ขา

ข้อจำกัด ESP8266

- MCU รักษาการเชื่อมต่อ WiFi และรันคำสั่งใน Sketch ในเวลาเดียวกัน
- wdt reset / “yield” command
- เหลือขาให้ใช้งานไม่มาก
- มีหลายขาที่มีการเรียกใช้ตอนบูต (GPIO0, GPIO2, GPIO16)
- ปัญหาระหว่างการบูต / มีขาใช้งานไม่เพียงพอ
- โหมดประหยัดพลังงาน
- รีเซ็ต signal เพื่อ wake-up
- สูญเสียการเชื่อมต่อ WiFi ใน
- ต้องเชื่อมต่อ WiFi ใหม่หลังจาก deep-sleep

วิธีรีเซ็ต signal เพื่อ wake-up ทำได้สองอย่างคือ สร้าง RTC ให้ ESP ขา GPIO16 หรือปลุกจาก signal ภายนอก

ESP32 Block Diagram



สถาปัตยกรรมนี้มี 48 pin ใช้งาน. ULP = ultra low power processor สามารถ deep sleep 2 core processor แต่ยังคงเชื่อมต่อ wifi ได้และปลุกตื่นได้ ใช้กระแส 150 micro ampre. รองรับ Bluetooth

Advanced Peripheral Interfaces

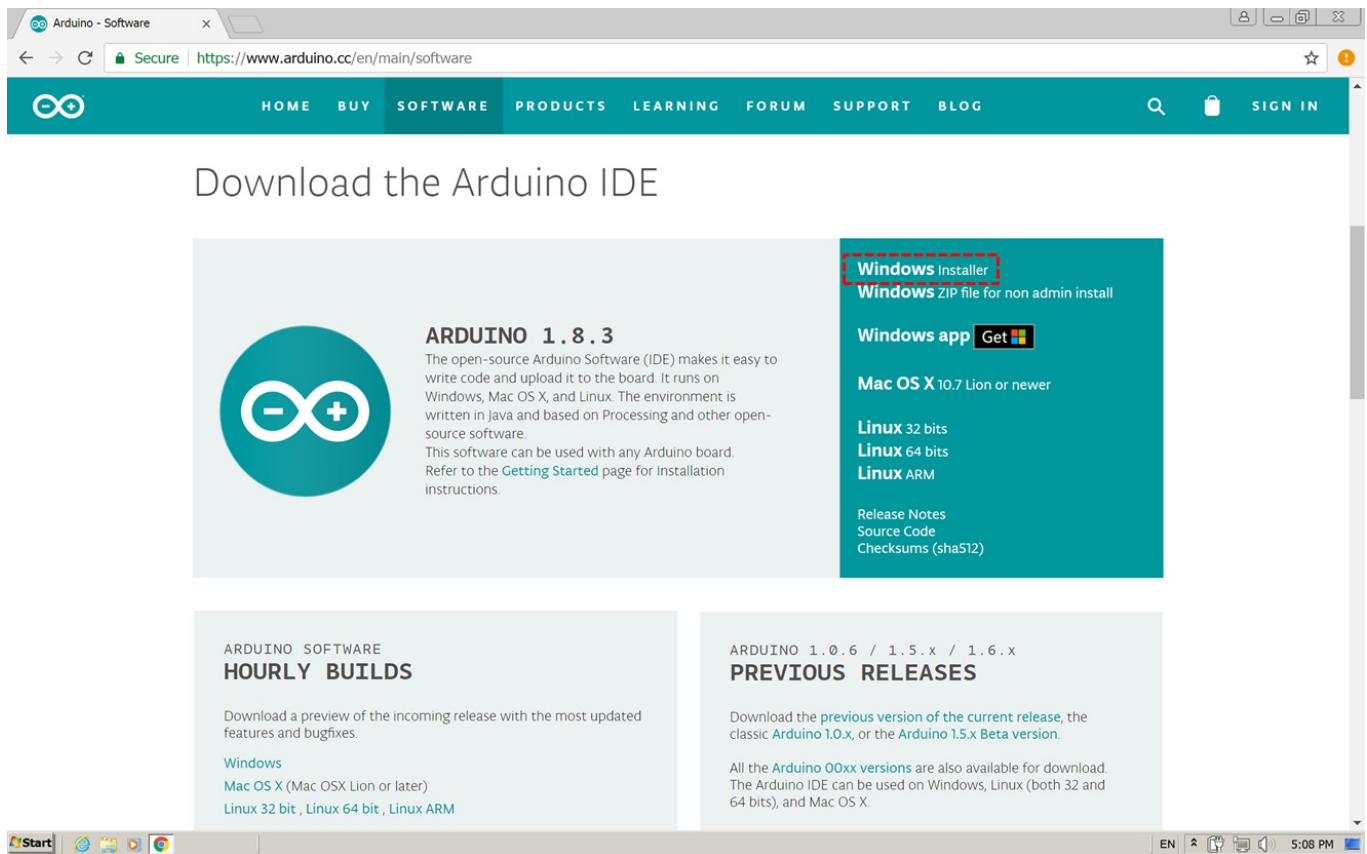
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit D/A converters / 10 × touch sensors / Temperature sensor
- 4 × SPI / 2 × I2S / 2 × I2C / 3×UART
- 1 host (SD/eMMC/SDIO) / 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN 2.0
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor
- Ultra low power analog pre-amplifier

การติดตั้ง Arduino ide

เครื่องมือตัวหลักสำหรับการเขียนโปรแกรม



<https://www.arduino.cc/en/main/software>



Download the Arduino IDE

ARDUINO 1.8.3
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows installer
Windows ZIP file for non admin install

Windows app [Get](#)

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

ARDUINO SOFTWARE HOURLY BUILDS
Download a preview of the incoming release with the most updated features and bugfixes.
[Windows](#)
[Mac OS X \(Mac OSX Lion or later\)](#)
[Linux 32 bit, Linux 64 bit, Linux ARM](#)

ARDUINO 1.0.6 / 1.5.x / 1.6.x PREVIOUS RELEASES
Download the previous version of the current release, the classic Arduino 1.0.x, or the Arduino 1.5.x Beta version.
All the Arduino 00xx versions are also available for download. The Arduino IDE can be used on Windows, Linux (both 32 and 64 bits), and Mac OS X.



Support the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). [Learn more](#) on how your contribution will be used.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED **16,349,348** TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

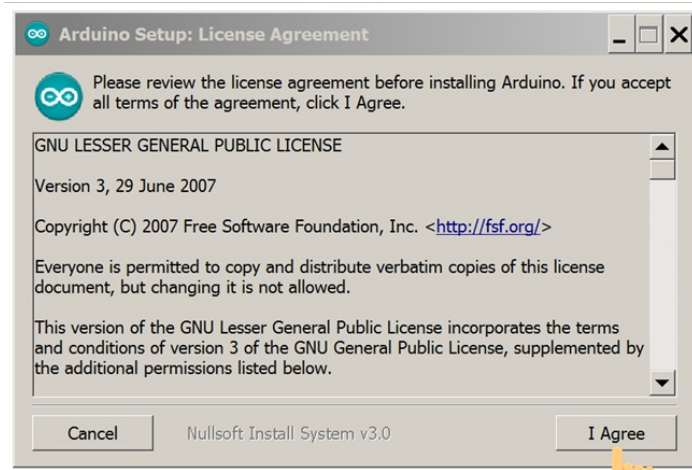
\$3 **\$5** **\$10** **\$25** **\$50** **OTHER**

[JUST DOWNLOAD](#) [CONTRIBUTE & DOWNLOAD](#)

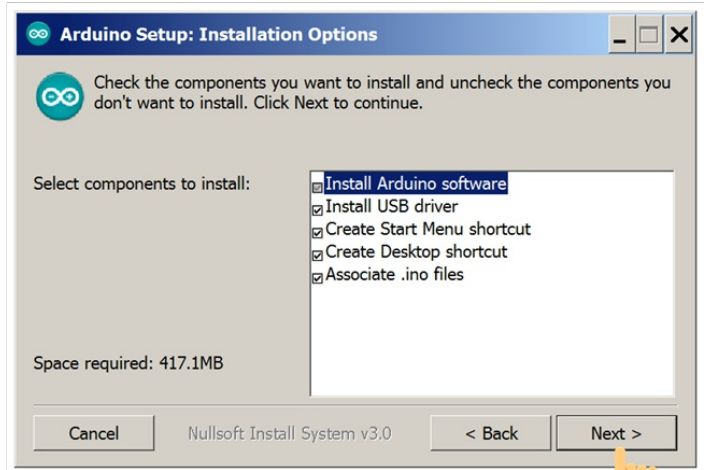
https://www.arduino.cc/download_handler.php

การติดตั้ง Arduino ide

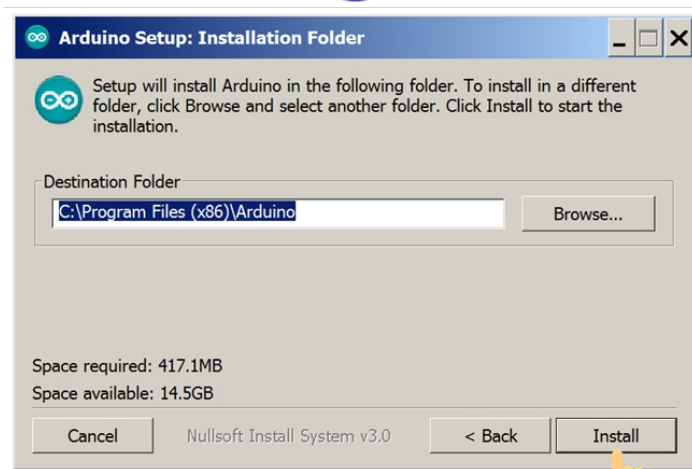
1



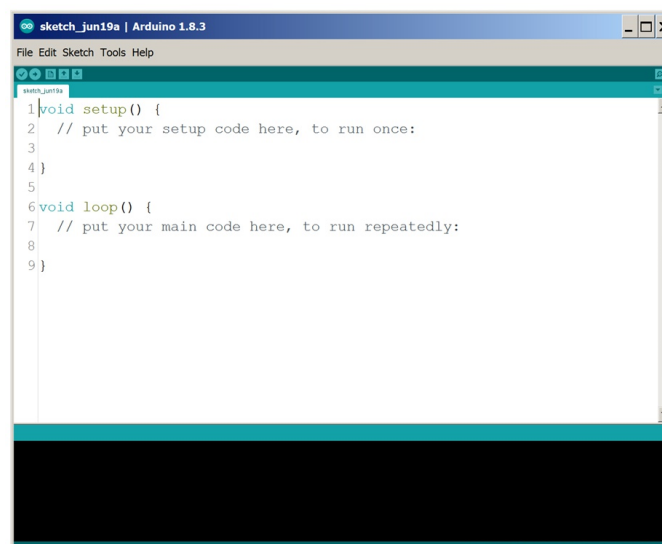
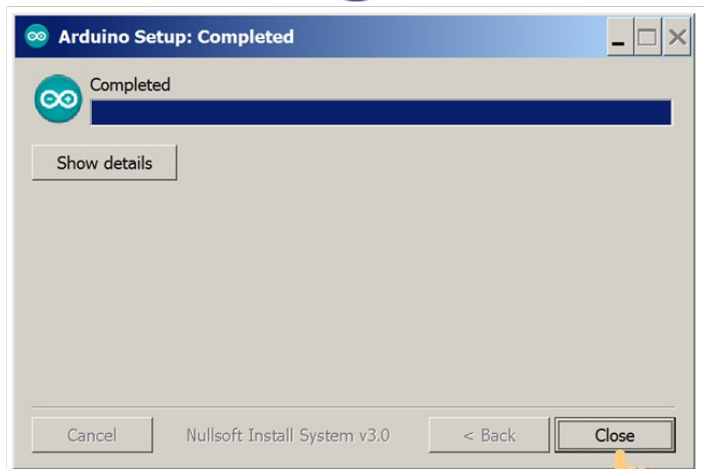
2



3

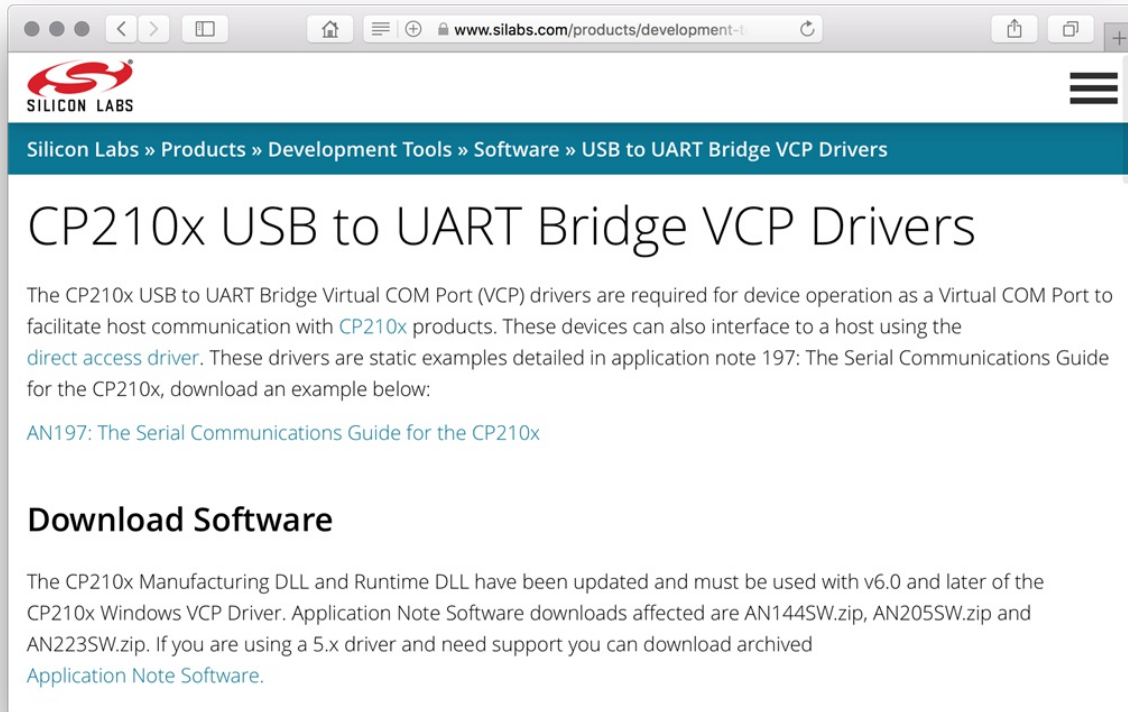


4



การติดตั้ง usb driver บน windows

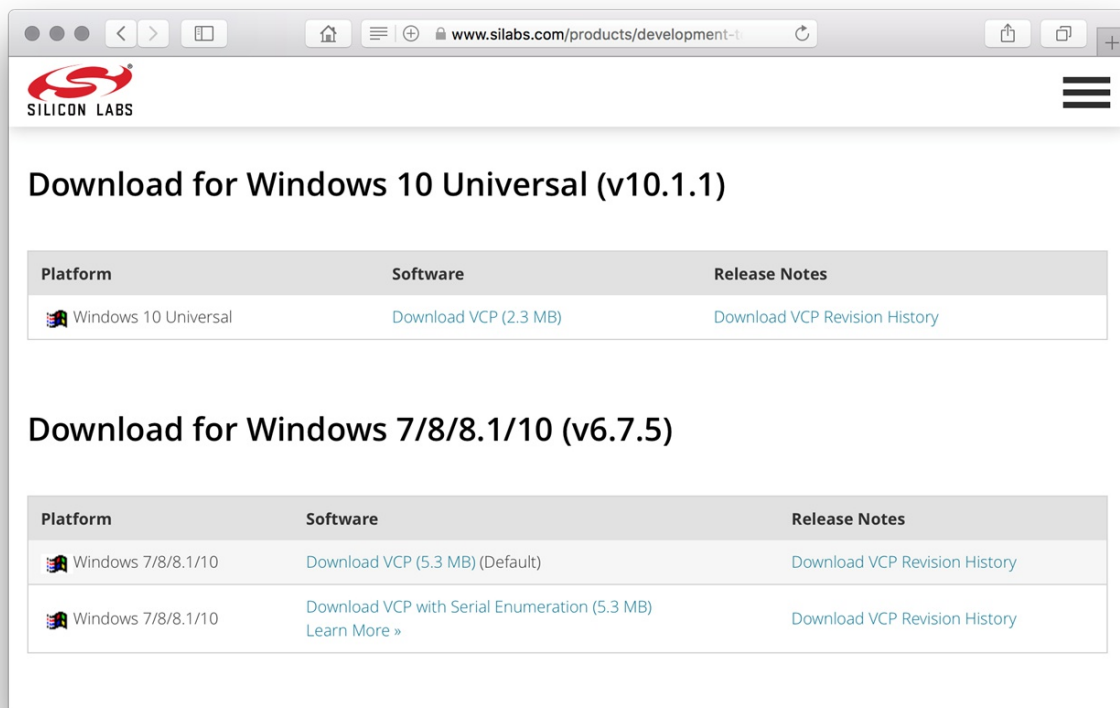
CP210x USB to UART Bridge VCP Driversสำหรับ ESP32



The screenshot shows the Silicon Labs website page for CP210x USB to UART Bridge VCP Drivers. The page includes the Silicon Labs logo, a breadcrumb trail (Silicon Labs » Products » Development Tools » Software » USB to UART Bridge VCP Drivers), and the main heading 'CP210x USB to UART Bridge VCP Drivers'. The text explains that these drivers are required for device operation as a Virtual COM Port to facilitate host communication with CP210x products. It also mentions that these devices can interface to a host using the direct access driver. A link to 'AN197: The Serial Communications Guide for the CP210x' is provided. Below this, there is a 'Download Software' section which states that the CP210x Manufacturing DLL and Runtime DLL have been updated and must be used with v6.0 and later of the CP210x Windows VCP Driver. It lists affected application note software downloads: AN144SW.zip, AN205SW.zip, and AN223SW.zip. A link to 'Application Note Software' is also present.

ลิงค์ดาวน์โหลด

<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

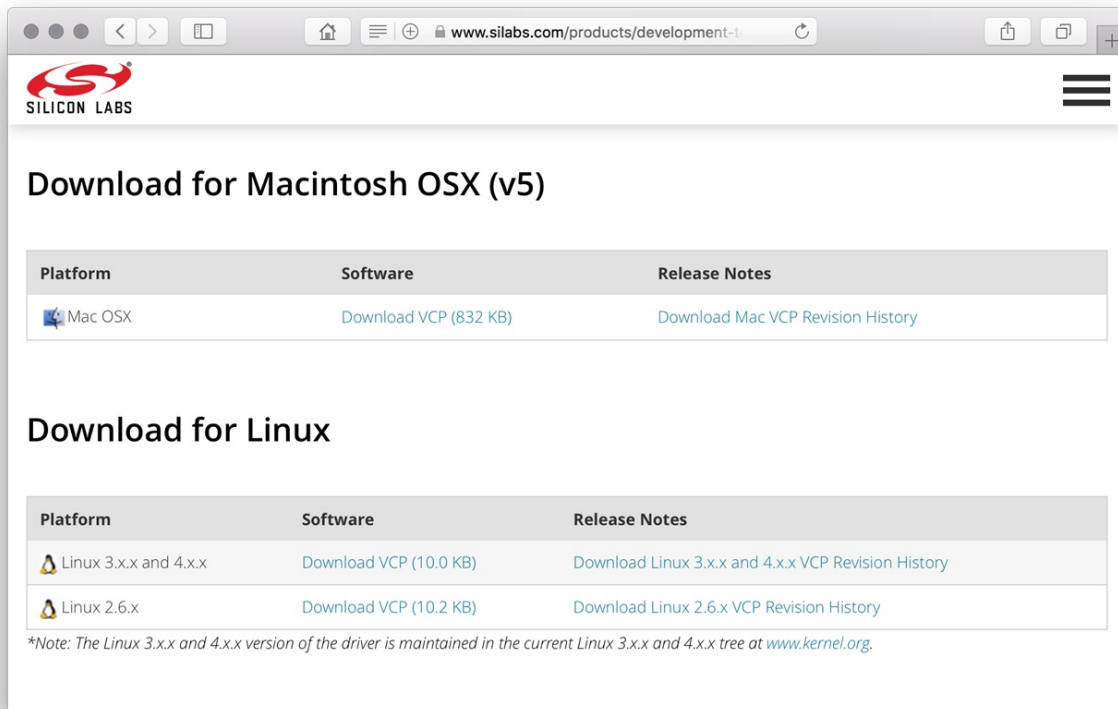


The screenshot shows the download section of the Silicon Labs website. It features two main sections: 'Download for Windows 10 Universal (v10.1.1)' and 'Download for Windows 7/8/8.1/10 (v6.7.5)'. Each section contains a table with columns for Platform, Software, and Release Notes.

Platform	Software	Release Notes
Windows 10 Universal	Download VCP (2.3 MB)	Download VCP Revision History

Platform	Software	Release Notes
Windows 7/8/8.1/10	Download VCP (5.3 MB) (Default)	Download VCP Revision History
Windows 7/8/8.1/10	Download VCP with Serial Enumeration (5.3 MB) Learn More »	Download VCP Revision History

การติดตั้ง usb driver CP210x USB to UART Bridge VCP Drivers



The screenshot shows the Silicon Labs website with the URL www.silabs.com/products/development-tools. The page is titled "Download for Macintosh OSX (v5)" and "Download for Linux". It contains two tables with download links for VCP drivers and revision history.

Download for Macintosh OSX (v5)

Platform	Software	Release Notes
 Mac OSX	Download VCP (832 KB)	Download Mac VCP Revision History

Download for Linux

Platform	Software	Release Notes
 Linux 3.x.x and 4.x.x	Download VCP (10.0 KB)	Download Linux 3.x.x and 4.x.x VCP Revision History
 Linux 2.6.x	Download VCP (10.2 KB)	Download Linux 2.6.x VCP Revision History

**Note: The Linux 3.x.x and 4.x.x version of the driver is maintained in the current Linux 3.x.x and 4.x.x tree at www.kernel.org.*

การติดตั้ง ESP32 ใน Arduino IDE

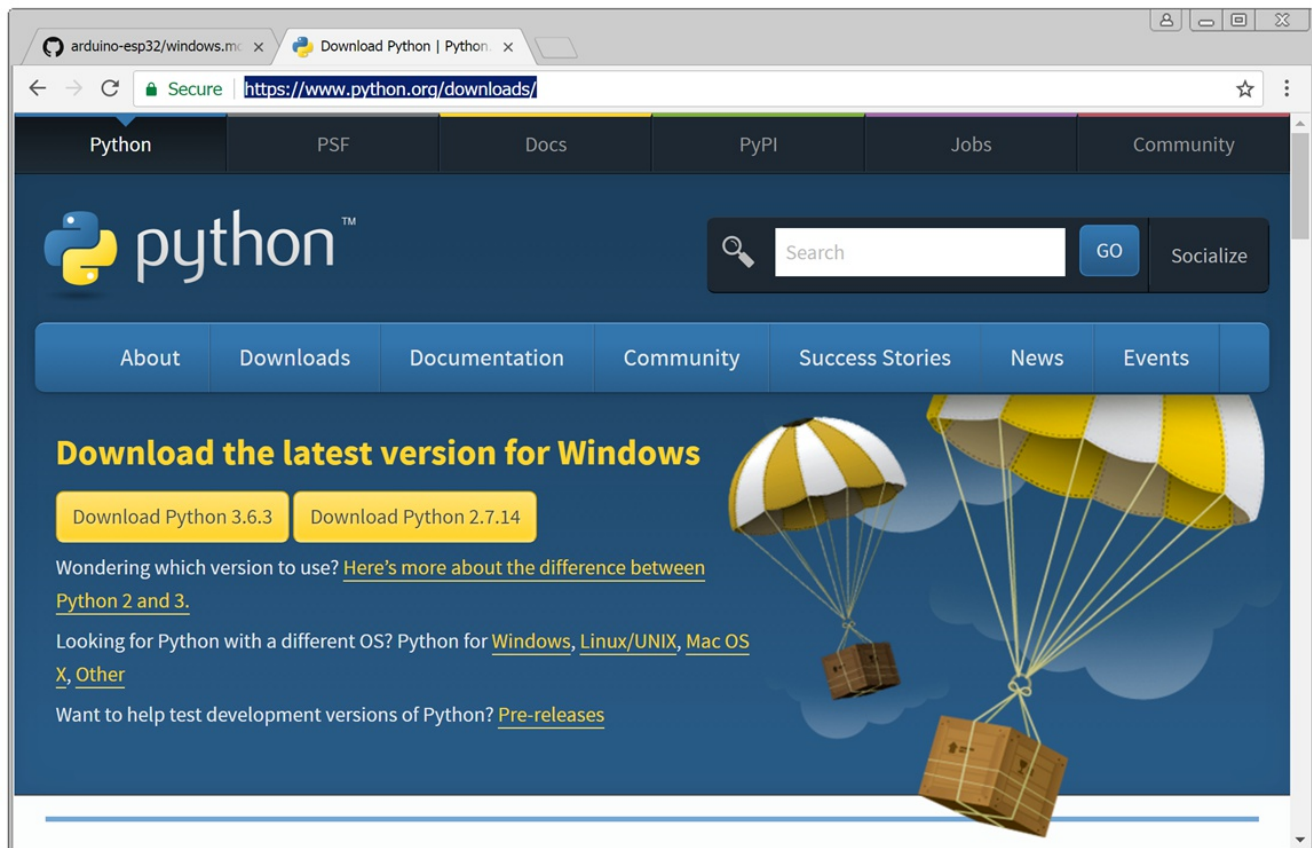
ขั้นตอนในการติดตั้ง ESP32

1. ติดตั้ง Arduino IDE เวอร์ชันล่าสุด
2. ติดตั้ง python 2.7x
3. ติดตั้ง git
4. โคลน existing repository
5. รัน git.exe
6. ติดตั้ง USB driver (ถ้าติดตั้งแล้วให้ข้ามไป)

การติดตั้ง ESP32 ใน Arduino IDE

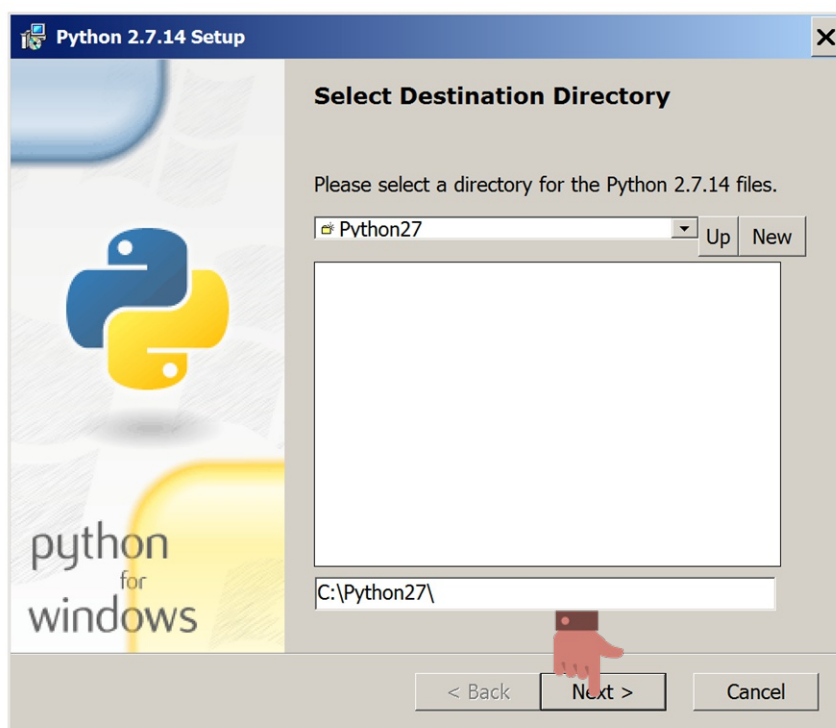
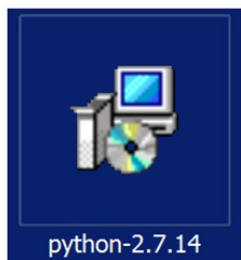
ขั้นตอนที่ 1. ดาวน์โหลด python บน windows

<https://www.python.org/downloads/>



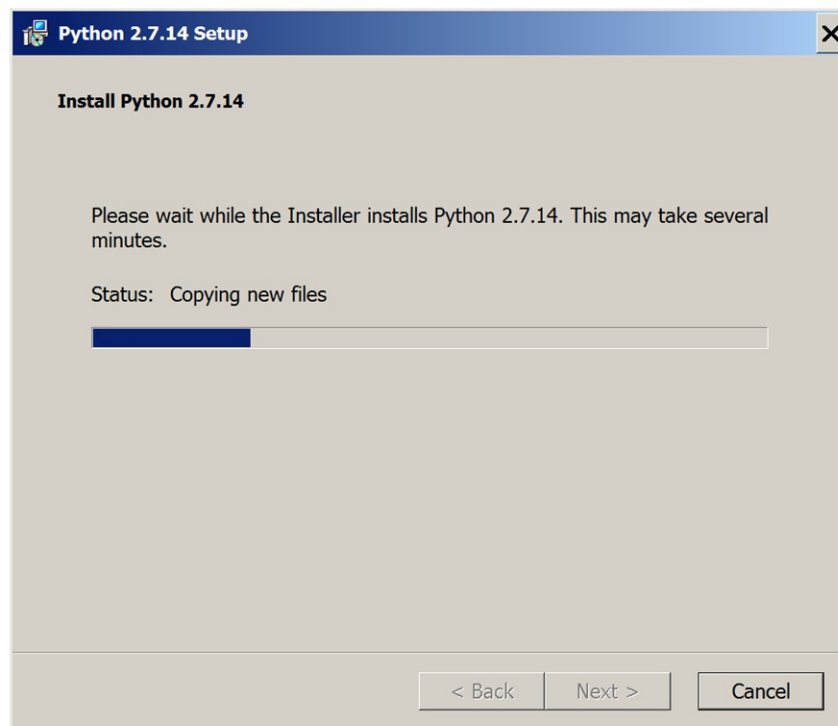
การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 2. ติดตั้ง python บน windows



การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 2. ติดตั้ง python บน windows



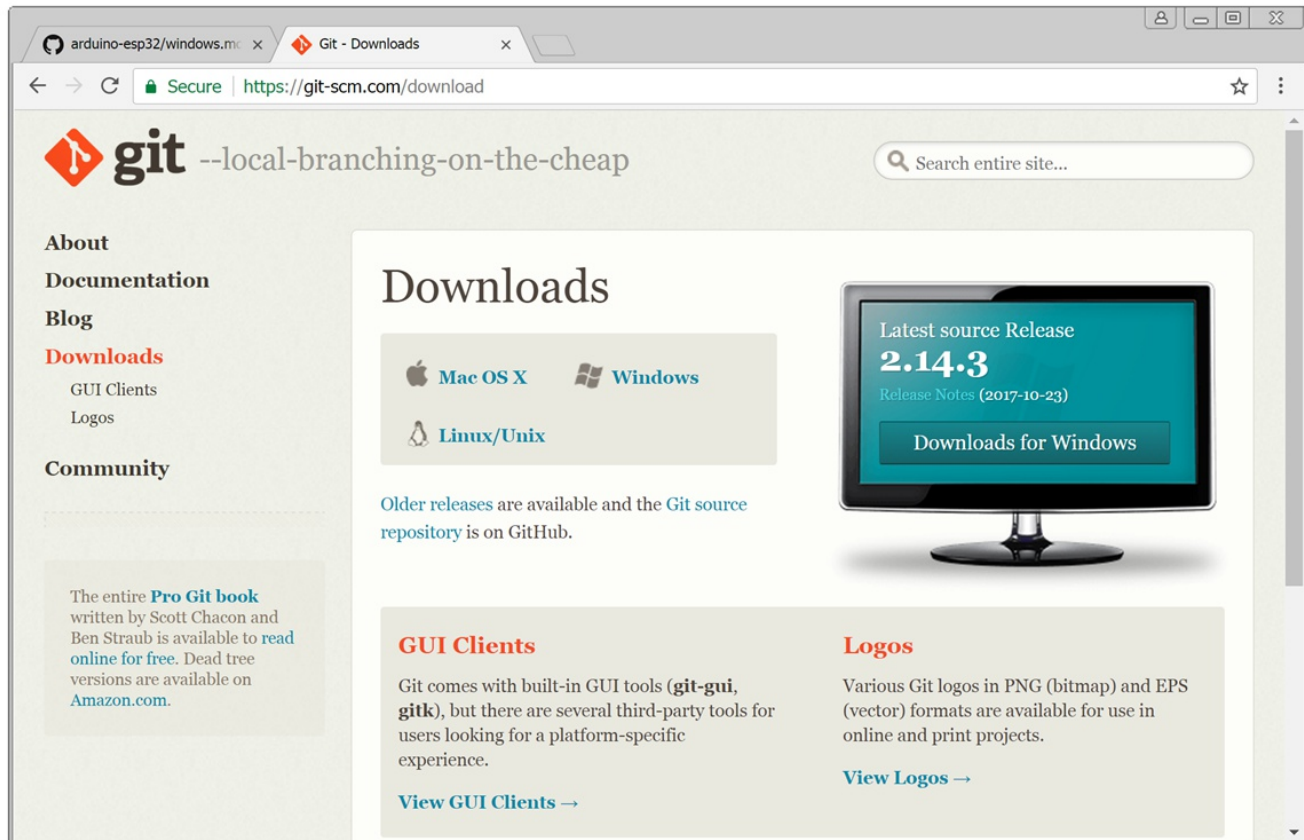
การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 2. ติดตั้ง python บน windows



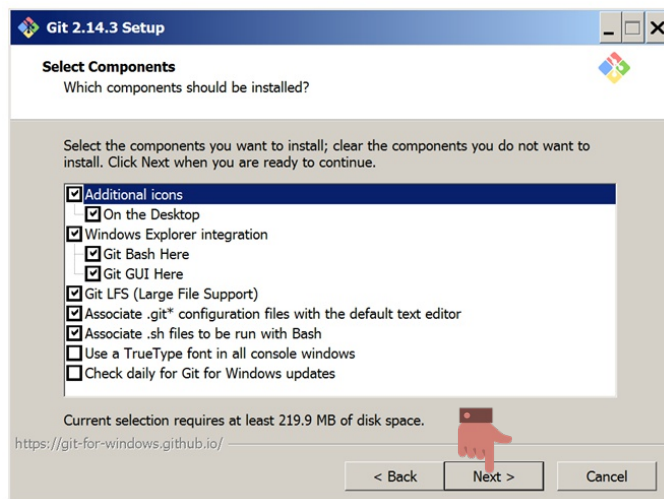
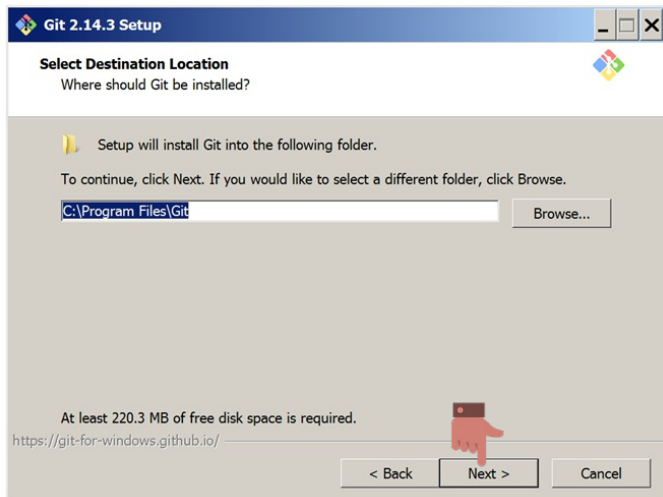
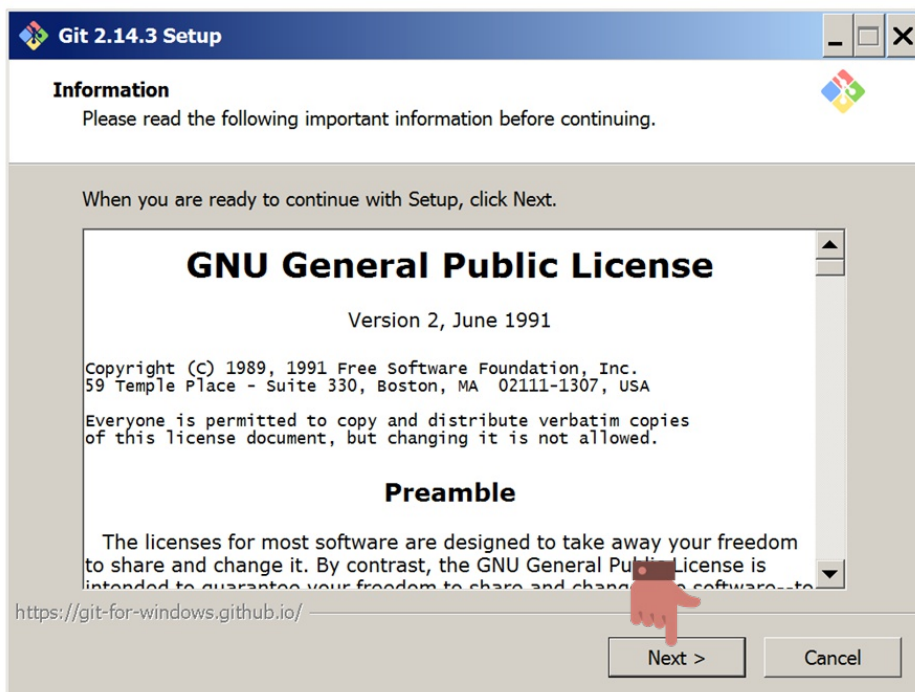
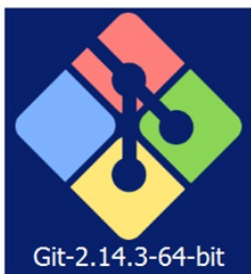
ขั้นตอนที่ 3. ดาวน์โหลด git บน windows

<https://git-scm.com/download>



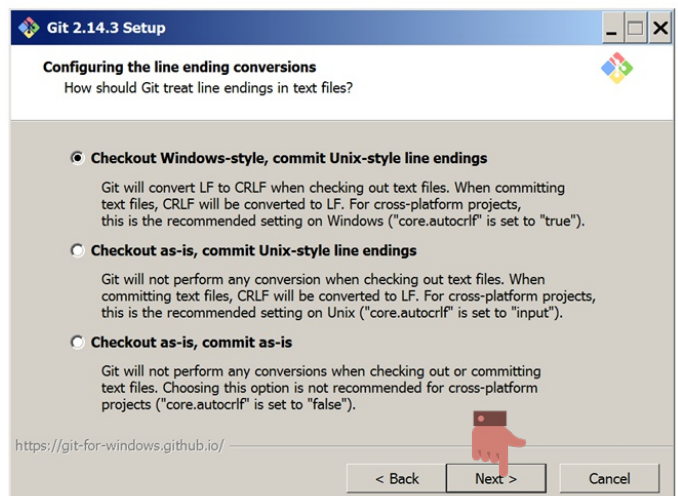
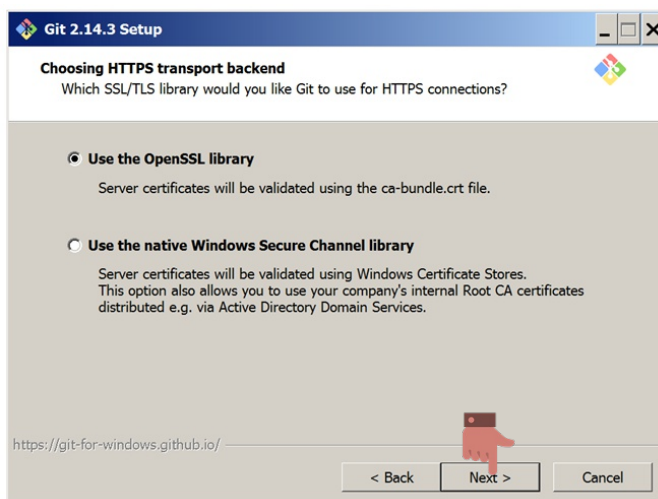
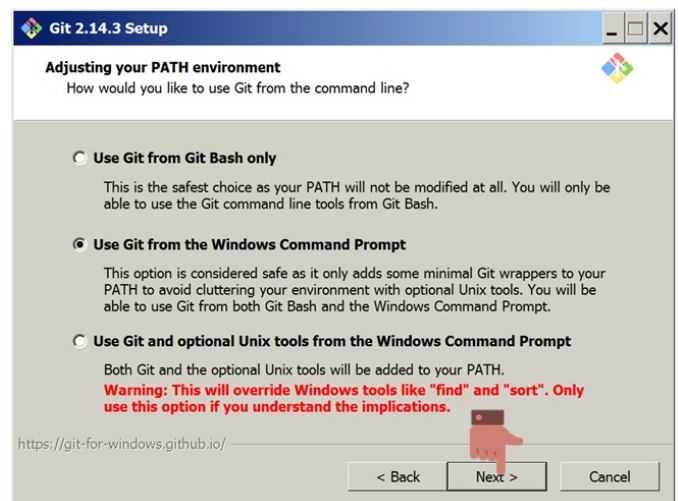
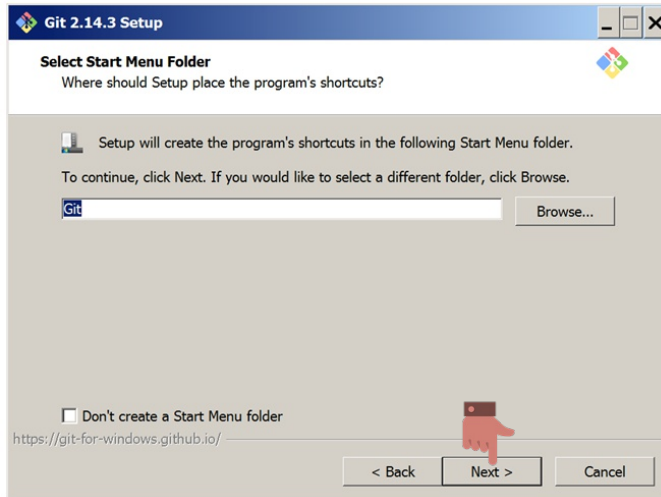
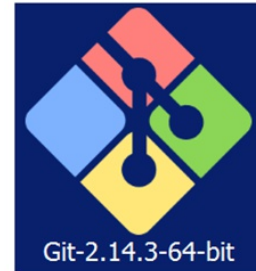
การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 4. ติดตั้ง git บน windows



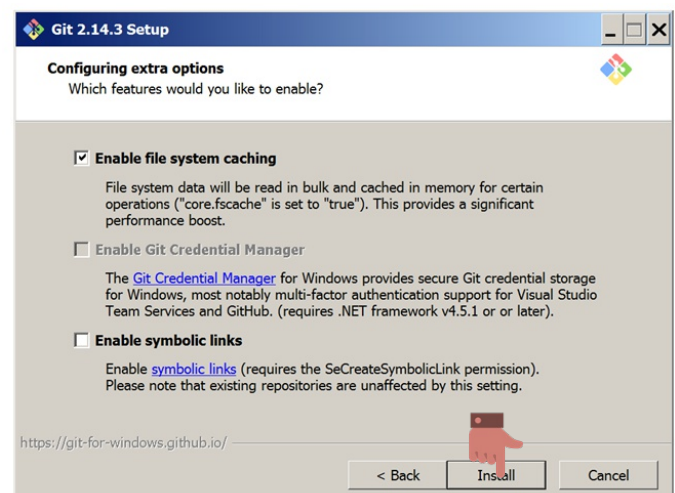
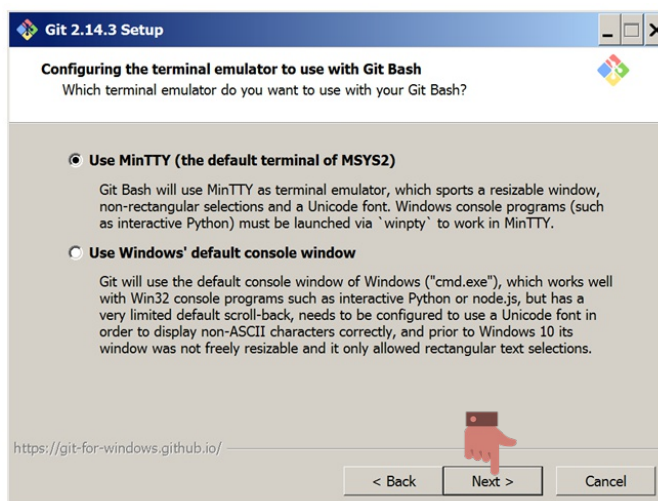
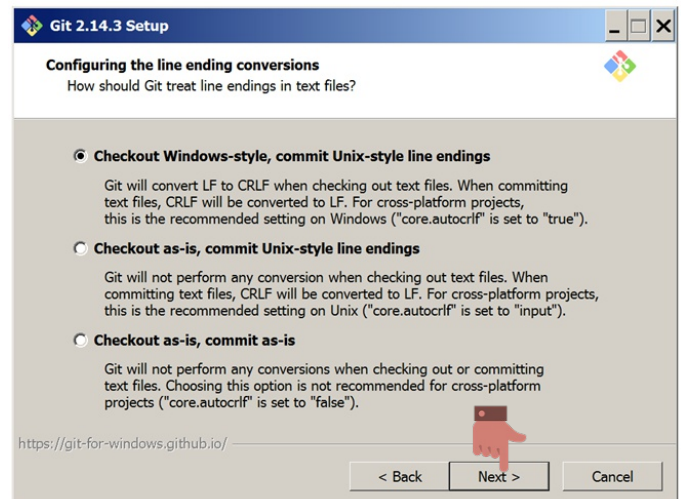
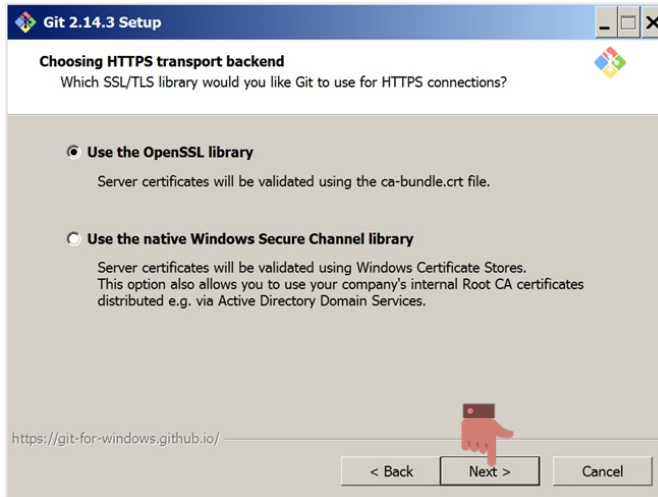
การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 4. ติดตั้ง git บน windows



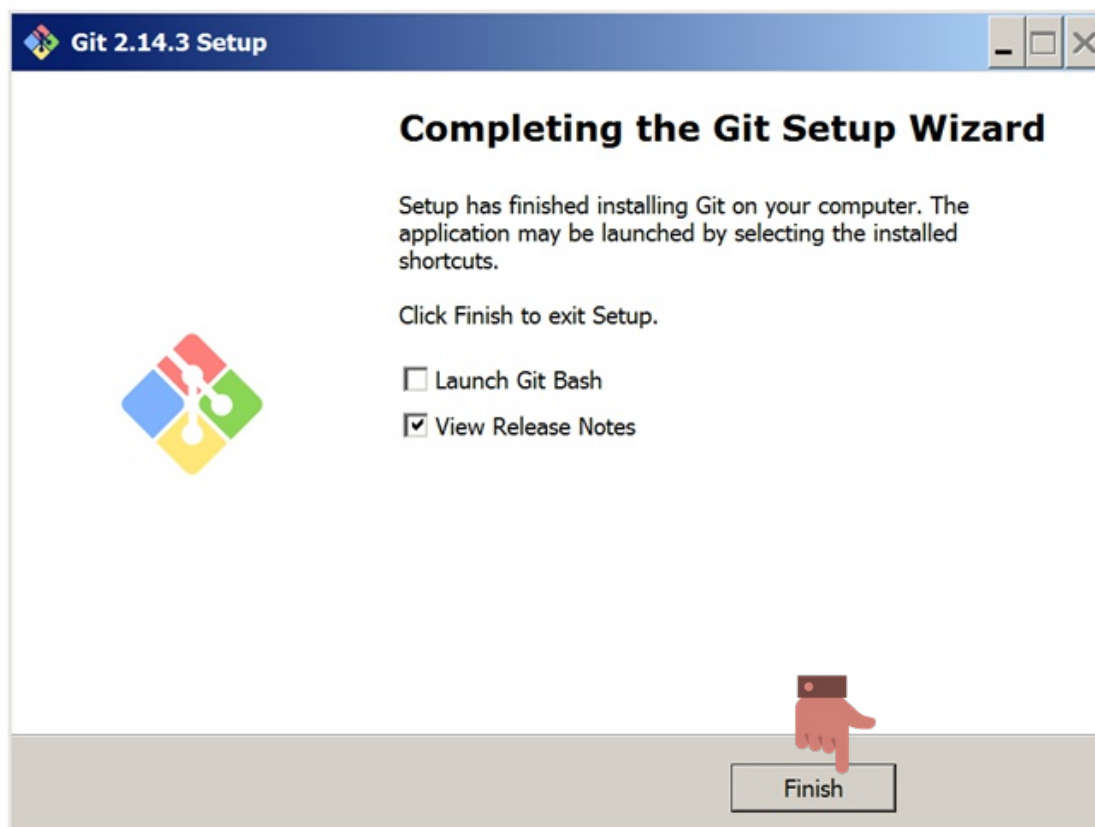
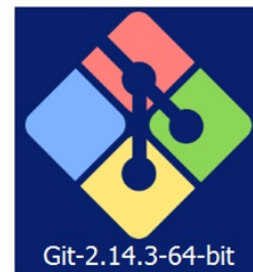
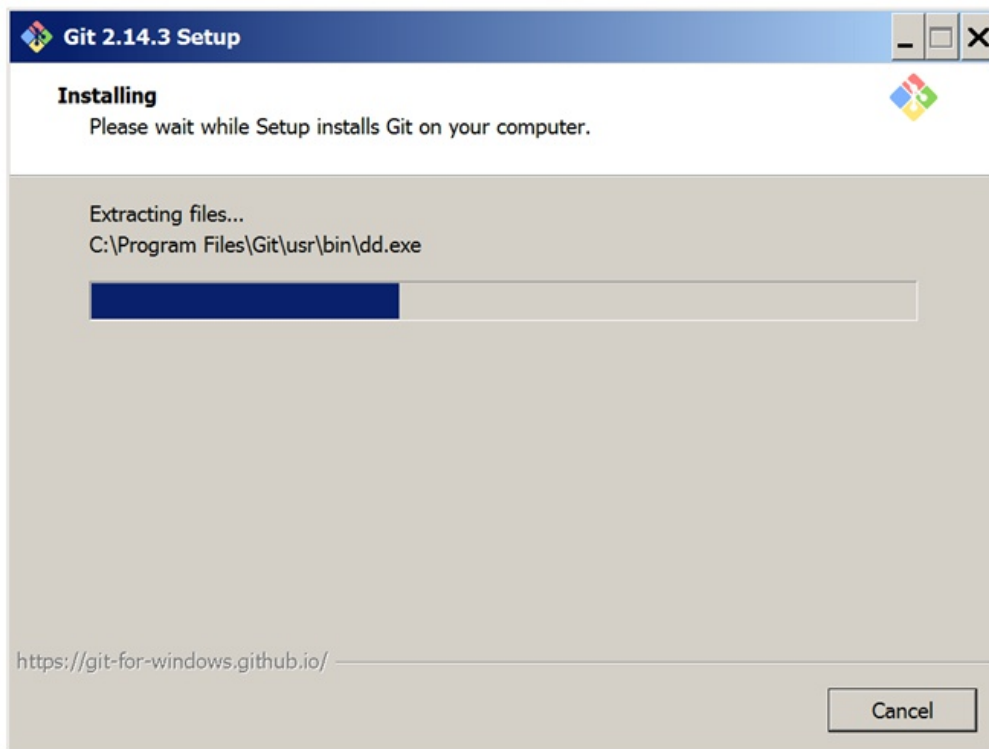
การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 4. ติดตั้ง git บน windows



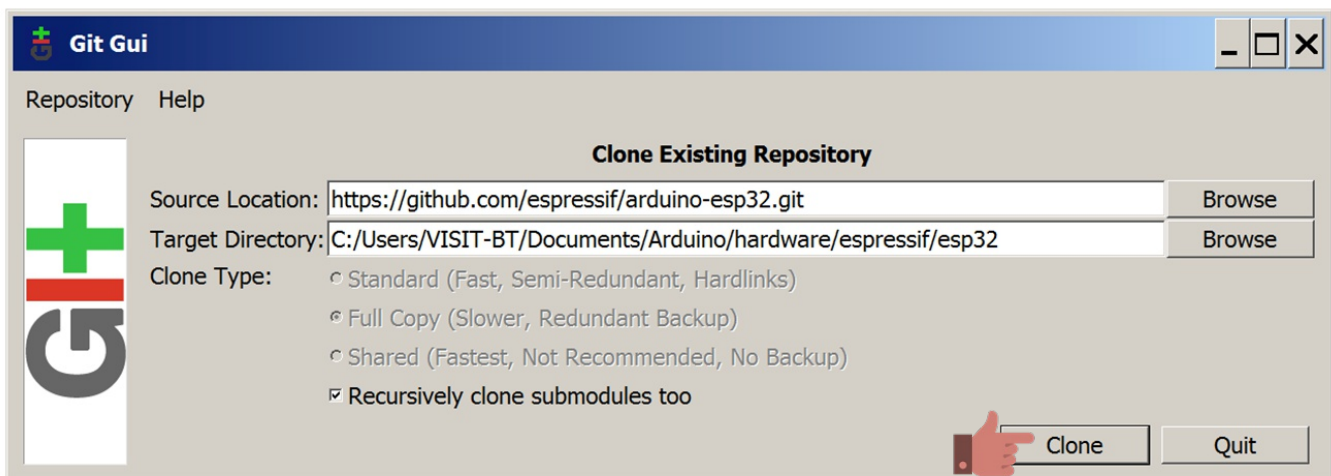
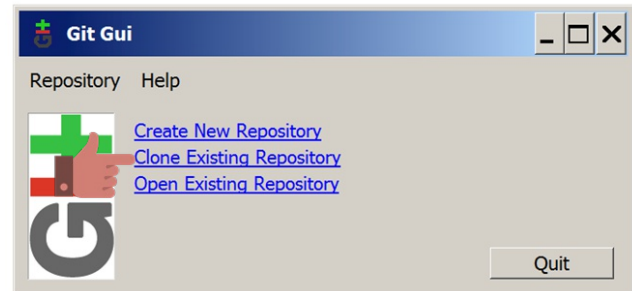
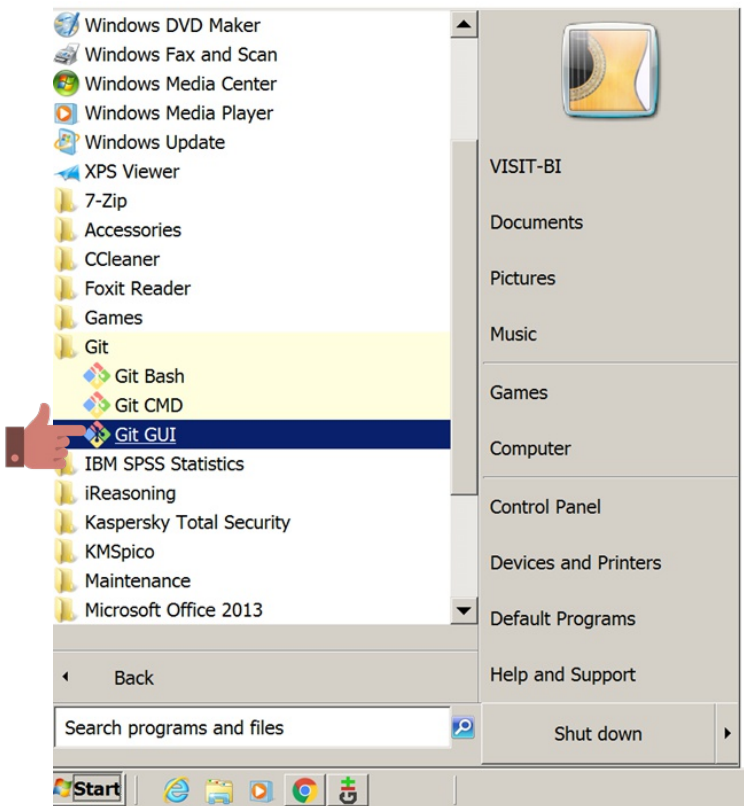
การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 4. ติดตั้ง git บน windows



การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 5. เพิ่มบอร์ด ESP32 ด้วย Git GUI



Source Location:

`https://github.com/espressif/arduino-esp32.git`

Target Directory:

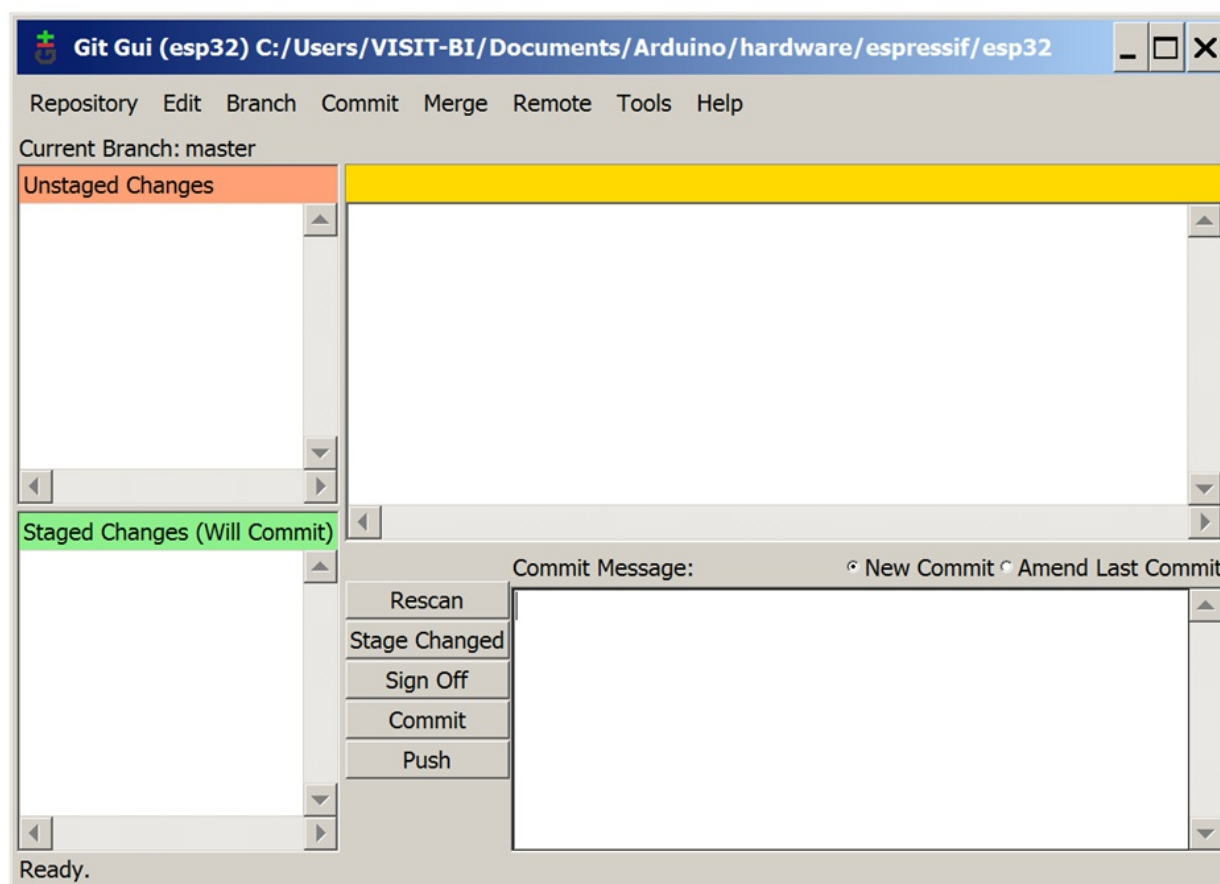
`C:/Users/YOUR_USER_NAME/Documents/Arduino/hardware/espressif/esp32`

การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 5. เพิ่มบอร์ด ESP32 ด้วย Git GUI



รอสักครู่ Git กำลังดาวน์โหลดไฟล์



โหลดเสร็จแล้วให้ปิดหน้าต่างนี้

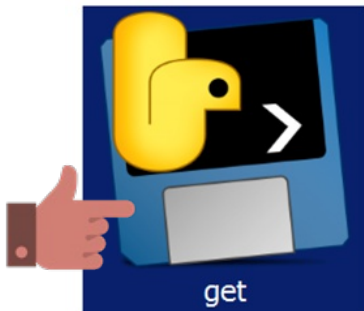
การติดตั้ง ESP32 ใน Arduino IDE

ขั้นตอนที่ 5. เพิ่มบอร์ด ESP32 ด้วย Git GUI

(1) เข้าไปที่

C:/Users/YOUR_USER_NAME/Documents/Arduino/hardware/espressif/esp32/tools

(2) แล้ว double-click get.exe จะแสดงหน้าต่างออกมา ถ้าเสร็จแล้วหน้าต่างจะปิดเอง



```

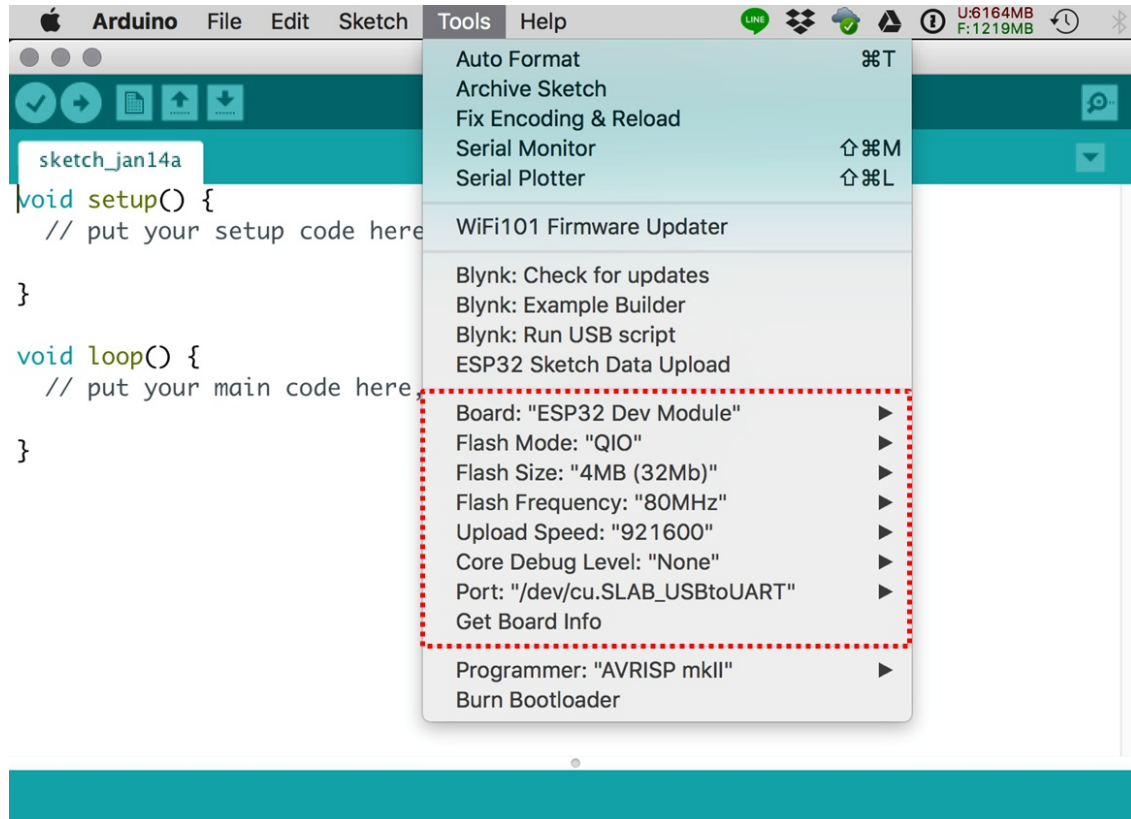
C:\Users\VISIT-BI\Documents\Arduino\hardware\espres...
System: Windows, Info: Windows-7-6.1.7601-SP1
Platform: i686-mingw32
Tool xtensa-esp32-elf-win32-1.22.0-73-ge28a011-5.2.0.zip already downloaded
Extracting xtensa-esp32-elf-win32-1.22.0-73-ge28a011-5.2.0.zip
  
```

หลังจากหน้าต่างปิดแล้วจะมีโฟลเดอร์ xtensa-esp32-elf เพิ่มเข้ามา

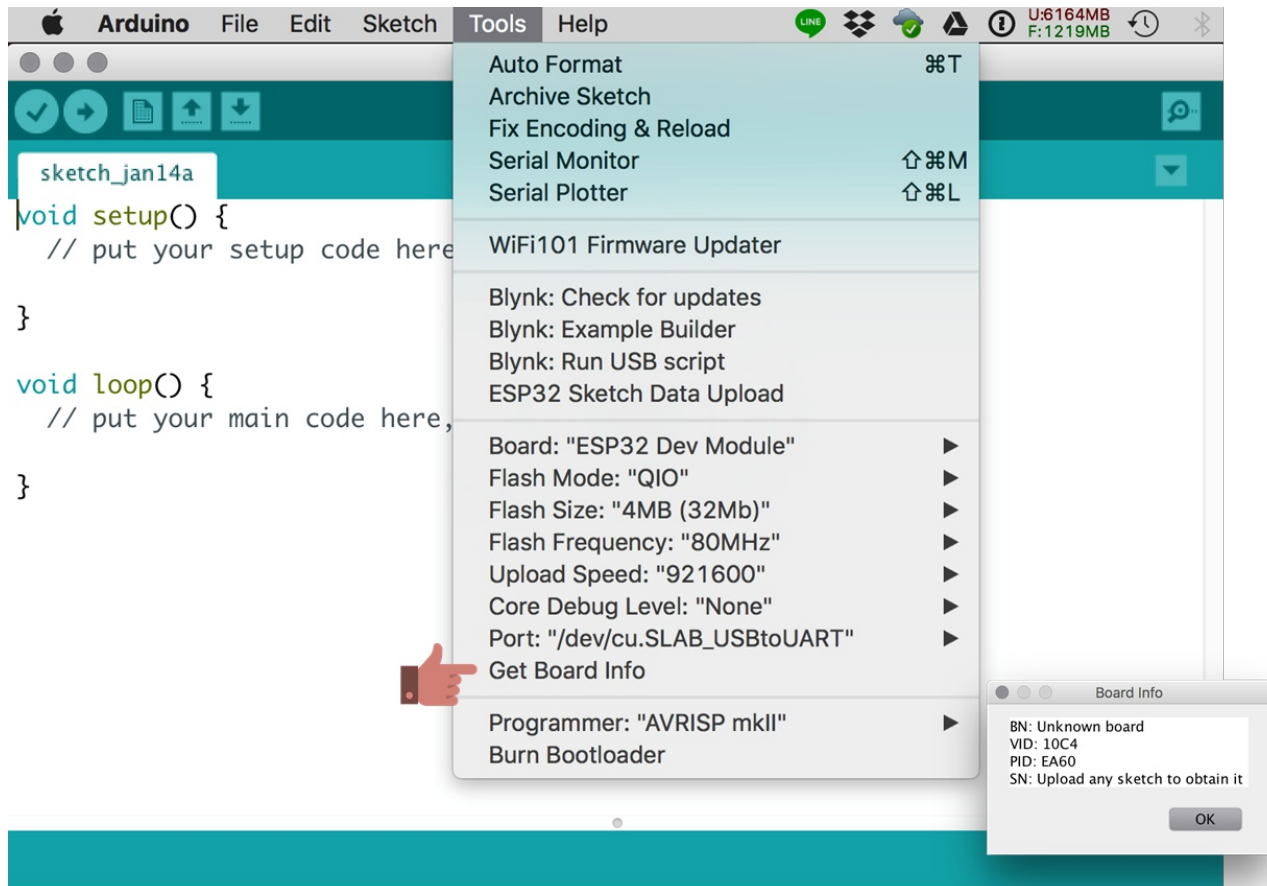
Name ^	Date modified	Type	Size
dist	27-Oct-17 7:34 PM	File folder	
mkspiffs	27-Oct-17 7:34 PM	File folder	
partitions	27-Oct-17 7:31 PM	File folder	
sdk	27-Oct-17 7:31 PM	File folder	
xtensa-esp32-elf	27-Oct-17 7:33 PM	File folder	
build	27-Oct-17 7:31 PM	PY File	5 KB
common	27-Oct-17 7:31 PM	SH File	3 KB
espota	27-Oct-17 7:31 PM	Application	3,936 KB
espota	27-Oct-17 7:31 PM	PY File	10 KB
esptool	27-Oct-17 7:34 PM	Application	3,382 KB
esptool	27-Oct-17 7:31 PM	PY File	113 KB
gen_esp32part	27-Oct-17 7:31 PM	Application	3,260 KB
gen_esp32part	27-Oct-17 7:31 PM	PY File	15 KB
get	27-Oct-17 7:31 PM	Application	5,090 KB
get	27-Oct-17 7:31 PM	PY File	5 KB
platformio-build	27-Oct-17 7:31 PM	PY File	7 KB

การเซตค่าบอร์ด ESP32 ใน Arduino ide

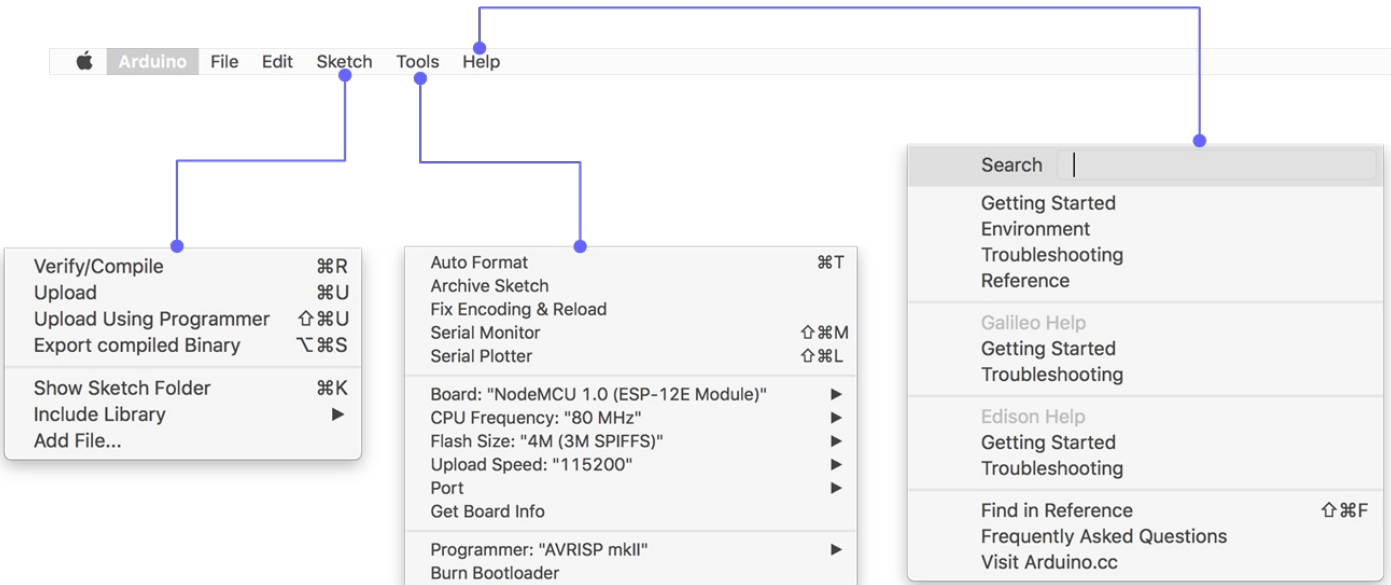
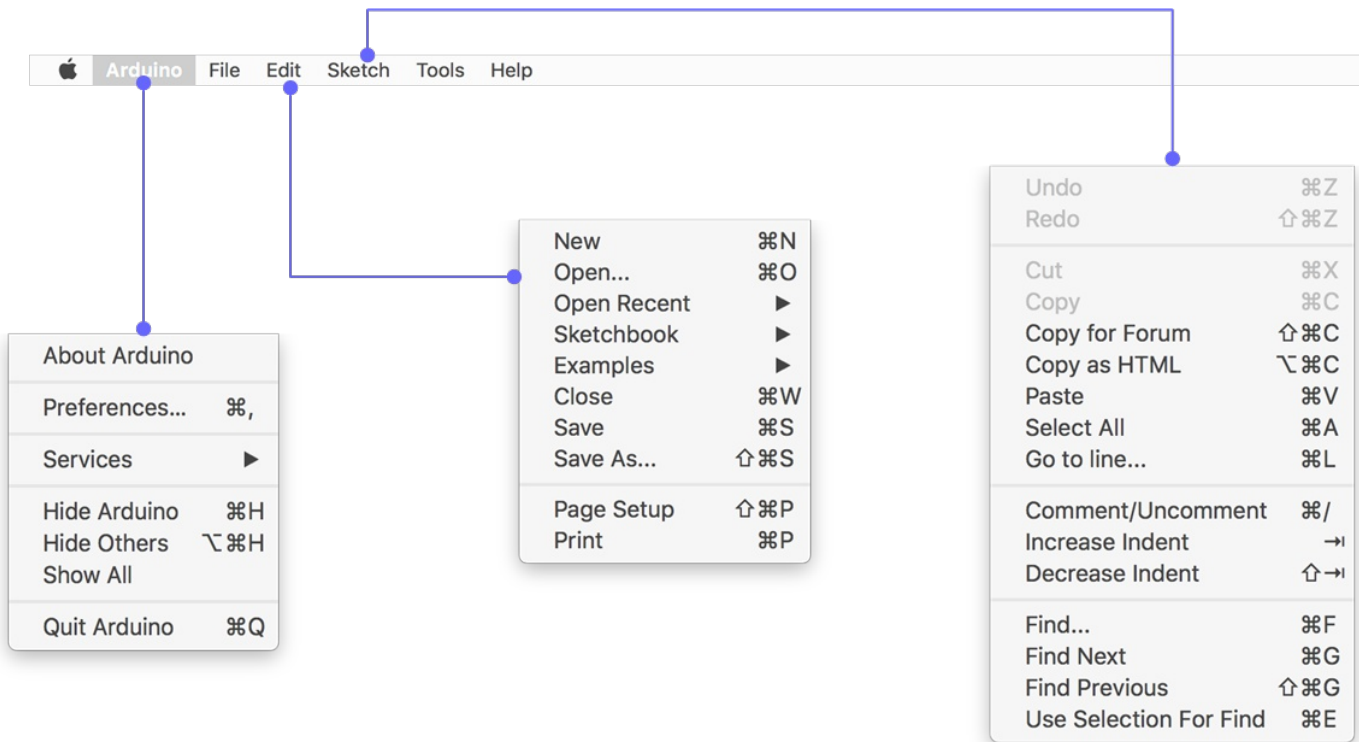
การเซตค่าบอร์ด ESP32 ใน Arduino ide



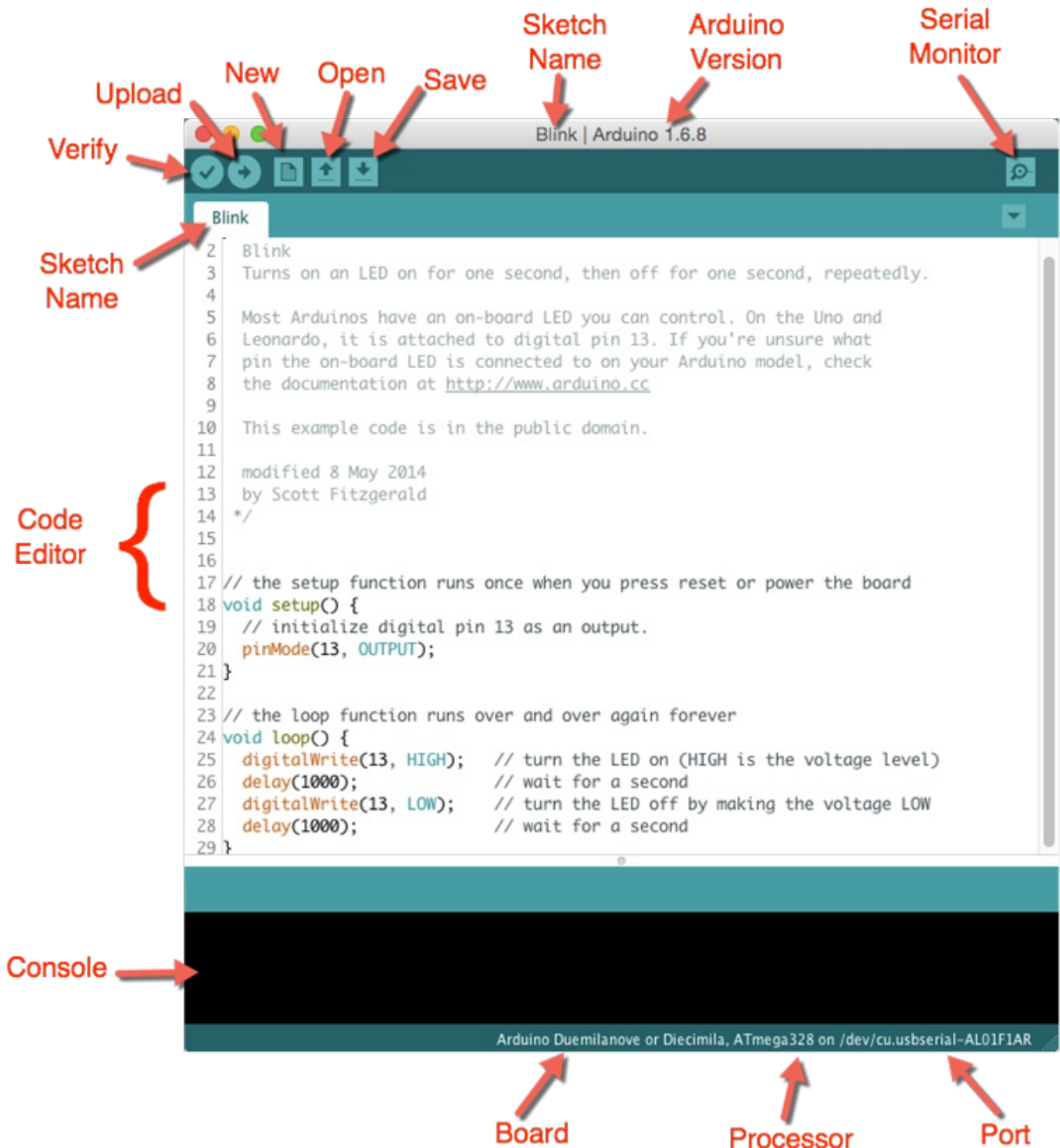
ทดสอบการเชื่อมต่อบอร์ด ESP32 ใน Arduino ide



การใช้งาน arduino ide เบื้องต้น

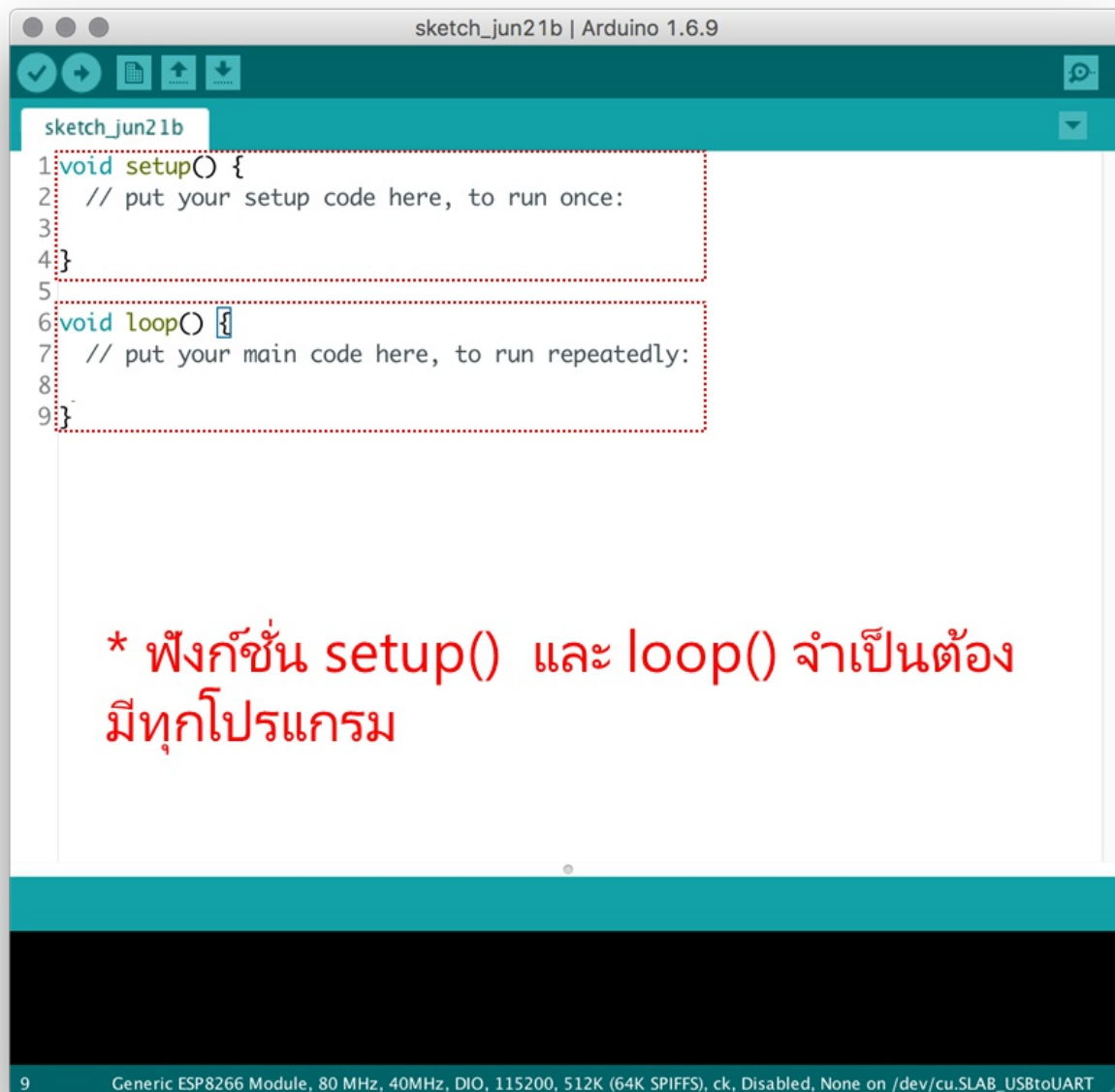


การใช้งาน arduino ide เบื้องต้น



โครงสร้างภาษา C สำหรับ Arduino เบื้องต้น

การประยุกต์ใช้ IoT สำหรับบุคคลทั่วไป



โครงสร้างภาษา C สำหรับ Arduino เบื้องต้น

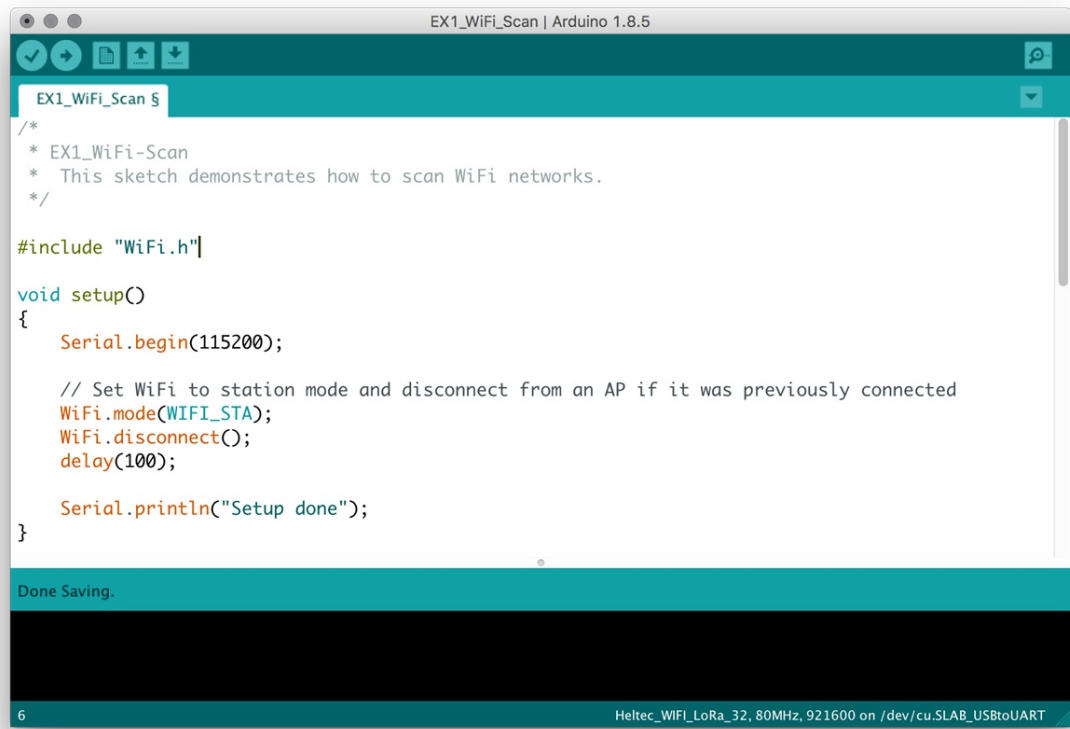


```
1 /*
2  Blink
3  Turns on an LED on for one second,
4  then off for one second, repeatedly.
5  */
6
7 int Pin = 13;
8
9 // the setup function runs once when you press reset or power the board
10 void setup() {
11   // initialize digital pin 13 as an output.
12   pinMode(Pin, OUTPUT);
13 }
14
15 // the loop function runs over and over again forever
16 void loop() {
17   digitalWrite(Pin, HIGH); // turn the LED on (HIGH is the voltage level)
18   delay(1000);             // wait for a second
19   digitalWrite(Pin, LOW);  // turn the LED off by making the voltage LOW
20   delay(1000);             // wait for a second
21 }
```

5 Generic ESP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFFS), ck, Disabled, None on /dev/cu.SLAB_USBtoUART

ทดสอบการอัปโหลด sketch

EX1_WiFi_Scan_ทดสอบการอัปโหลด sketch



The screenshot shows the Arduino IDE interface with the sketch 'EX1_WiFi_Scan' loaded. The code is as follows:

```

/*
 * EX1_WiFi-Scan
 * This sketch demonstrates how to scan WiFi networks.
 */

#include "WiFi.h"

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

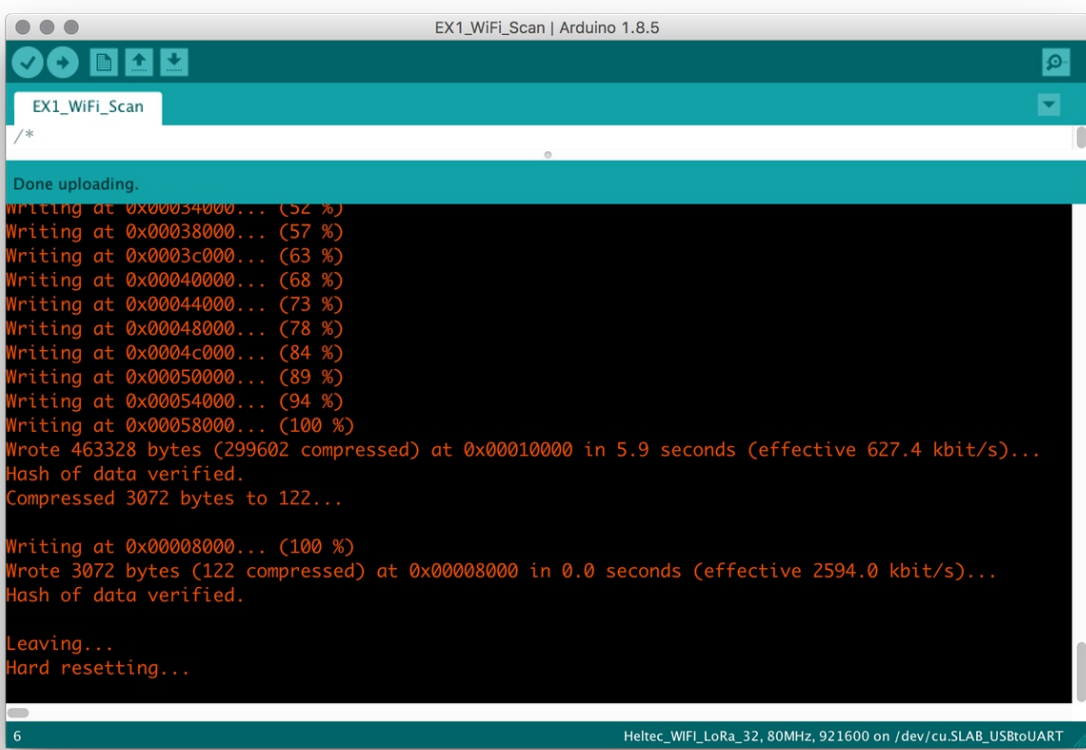
  // Set WiFi to station mode and disconnect from an AP if it was previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

  Serial.println("Setup done");
}

```

Below the code editor, a status bar indicates 'Done Saving.' and the bottom status bar shows 'Heltec_WiFi_LoRa_32, 80MHz, 921600 on /dev/cu.SLAB_USBtoUART'.

ทดสอบการอัปโหลด Sketch



The screenshot shows the Arduino IDE interface during the upload process. The output window displays the following messages:

```

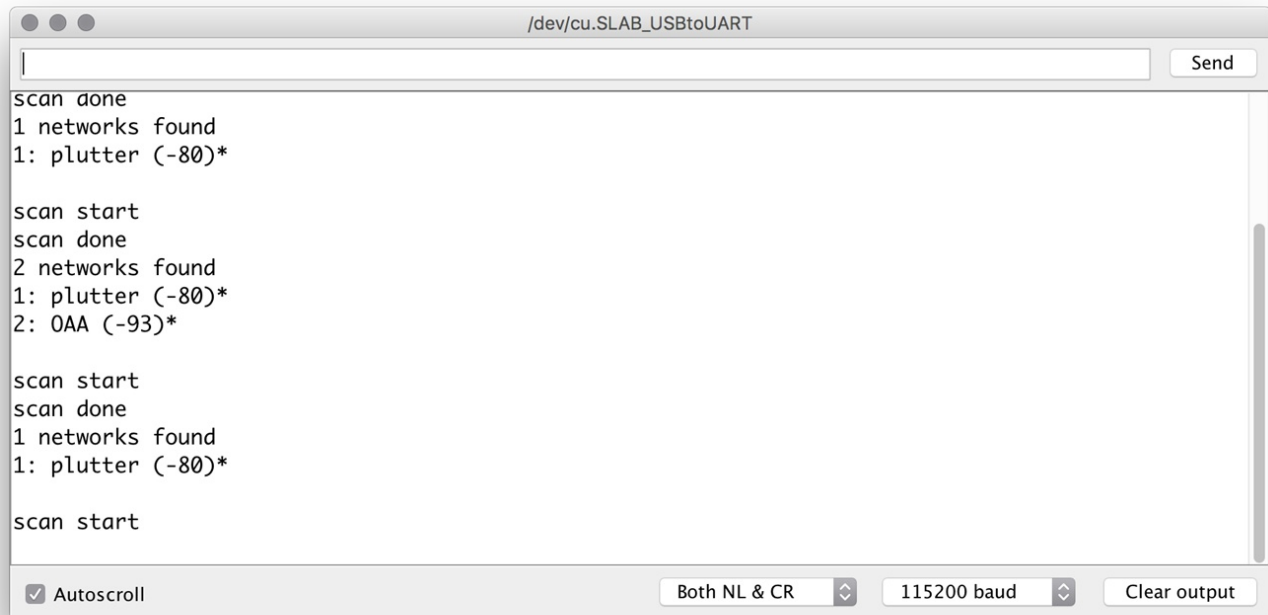
Done uploading.
Writing at 0x00034000... (52 %)
Writing at 0x00038000... (57 %)
Writing at 0x0003c000... (63 %)
Writing at 0x00040000... (68 %)
Writing at 0x00044000... (73 %)
Writing at 0x00048000... (78 %)
Writing at 0x0004c000... (84 %)
Writing at 0x00050000... (89 %)
Writing at 0x00054000... (94 %)
Writing at 0x00058000... (100 %)
Wrote 463328 bytes (299602 compressed) at 0x00010000 in 5.9 seconds (effective 627.4 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 122...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds (effective 2594.0 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting...

```

The bottom status bar shows 'Heltec_WiFi_LoRa_32, 80MHz, 921600 on /dev/cu.SLAB_USBtoUART'.

แสดงการอัปโหลดเสร็จสมบูรณ์

ทดสอบการอัปโหลด sketch



The screenshot shows a serial monitor window titled "/dev/cu.SLAB_USBtoUART". The window contains a text area with the following output:

```
scan done
1 networks found
1: plutter (-80)*

scan start
scan done
2 networks found
1: plutter (-80)*
2: OAA (-93)*

scan start
scan done
1 networks found
1: plutter (-80)*

scan start
```

At the bottom of the window, there is a control bar with the following elements:

- A checked checkbox labeled "Autoscroll".
- A dropdown menu set to "Both NL & CR".
- A dropdown menu set to "115200 baud".
- A "Clear output" button.
- A "Send" button is located at the top right of the text area.

แสดงผลพอร์มในคอนโซล

การใช้งานพอร์ตดิจิทัล และอนาล็อก GPIO

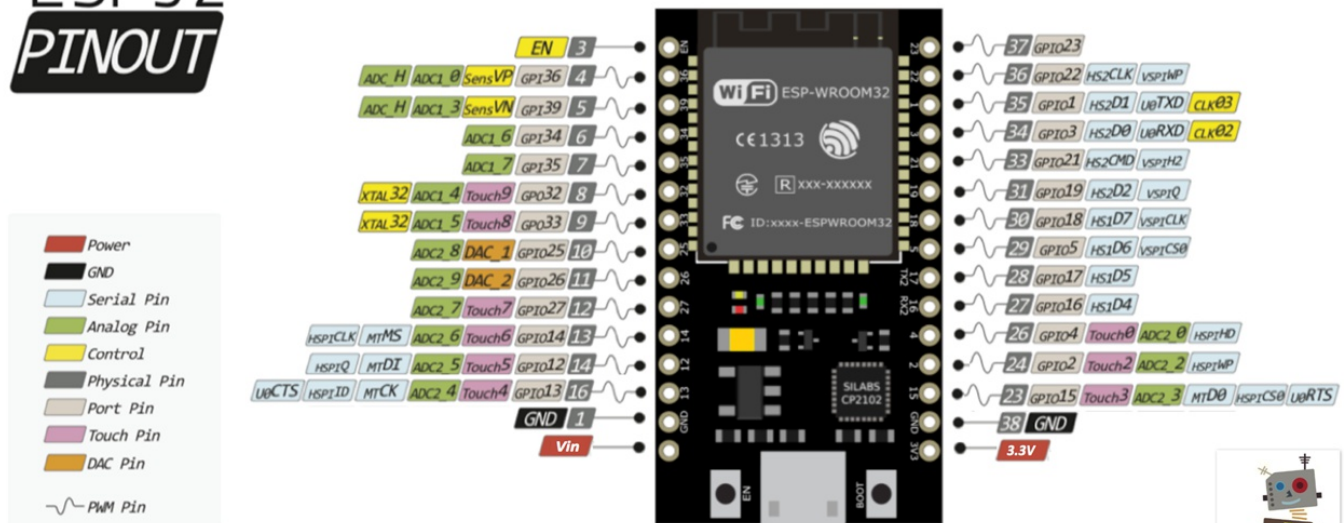
การประยุกต์ใช้ IoT สำหรับบุคคลทั่วไป



ESP32

ESP32 PINOUT

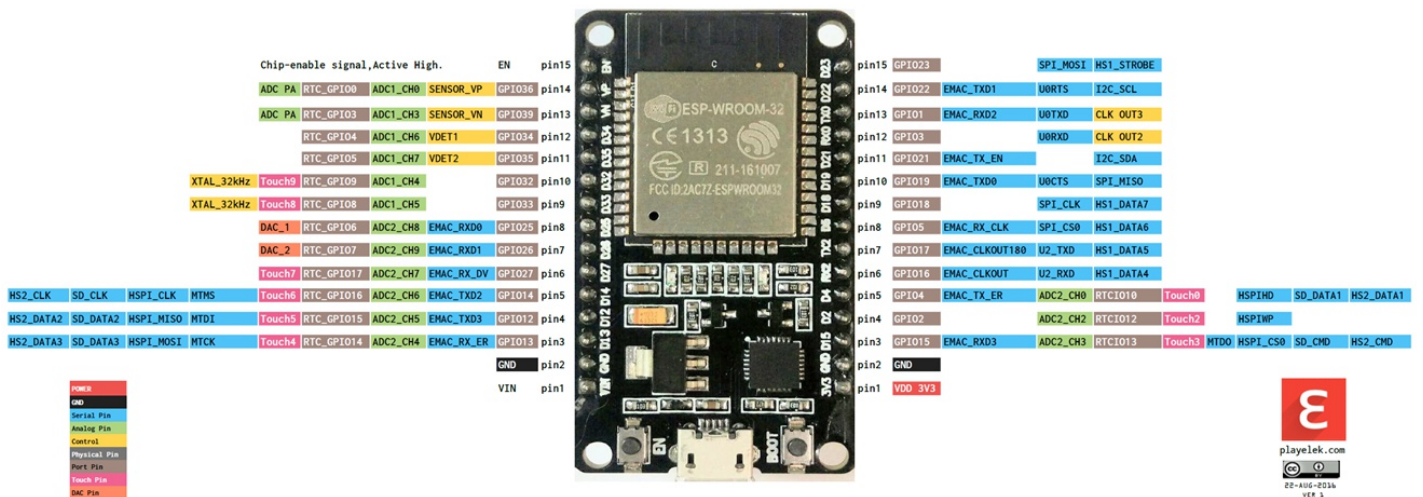
ESP32 DEVELOPMENT BOARD DUAL CORE ESP-32 & ESP-32S BOARD



Freely adapted by <https://MJRoBot.org>



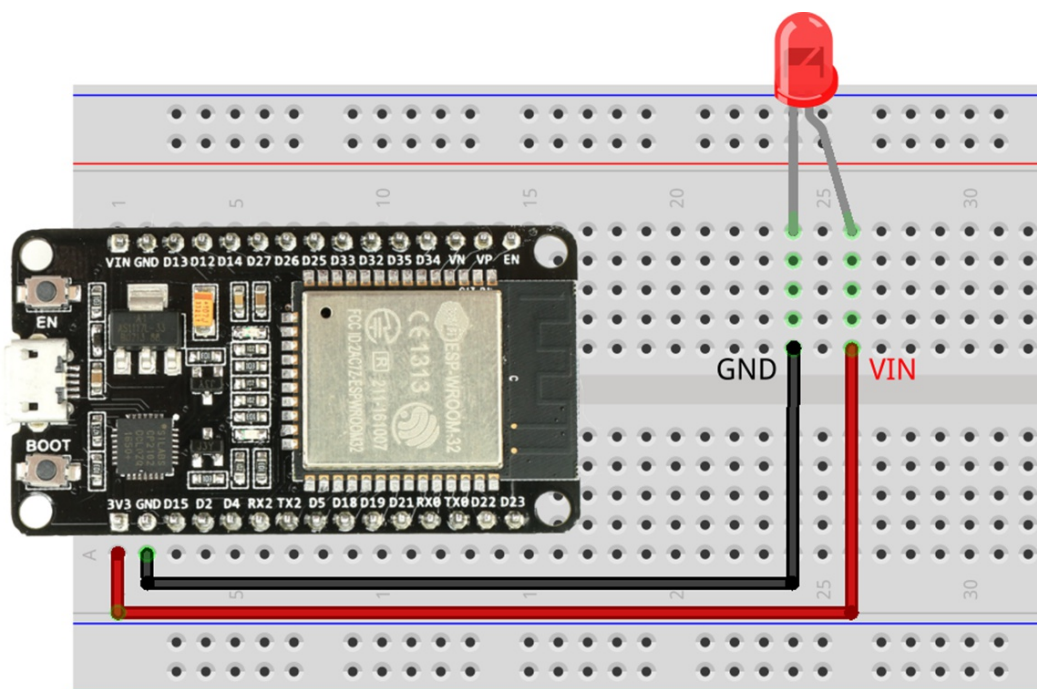
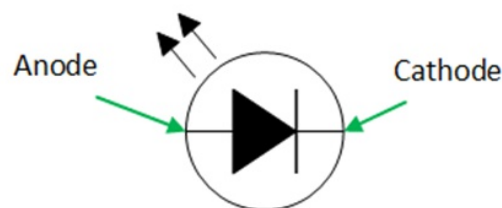
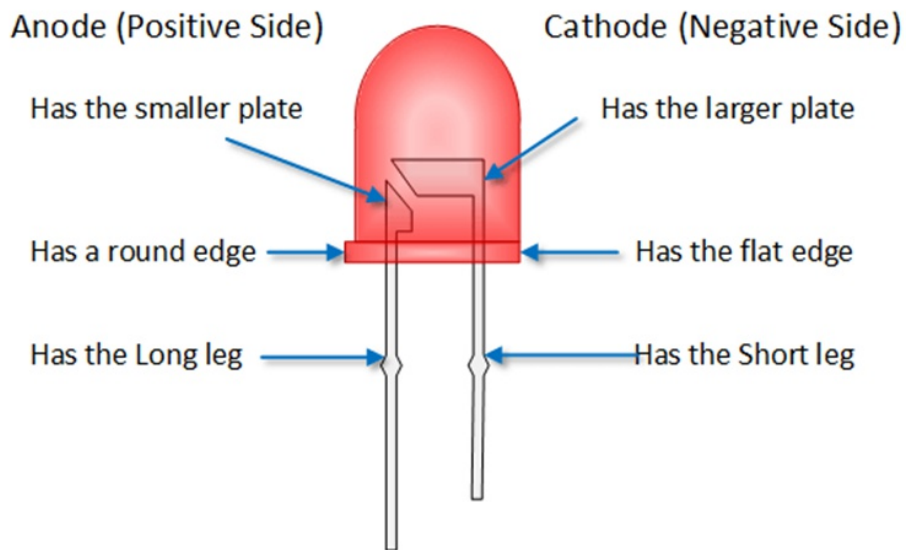
DOIT ESP32 DEVKIT V1 PINOUT



การเขียนโปรแกรมใช้งานพอร์ตดิจิทัล GPIO

ภาคปฏิบัติ

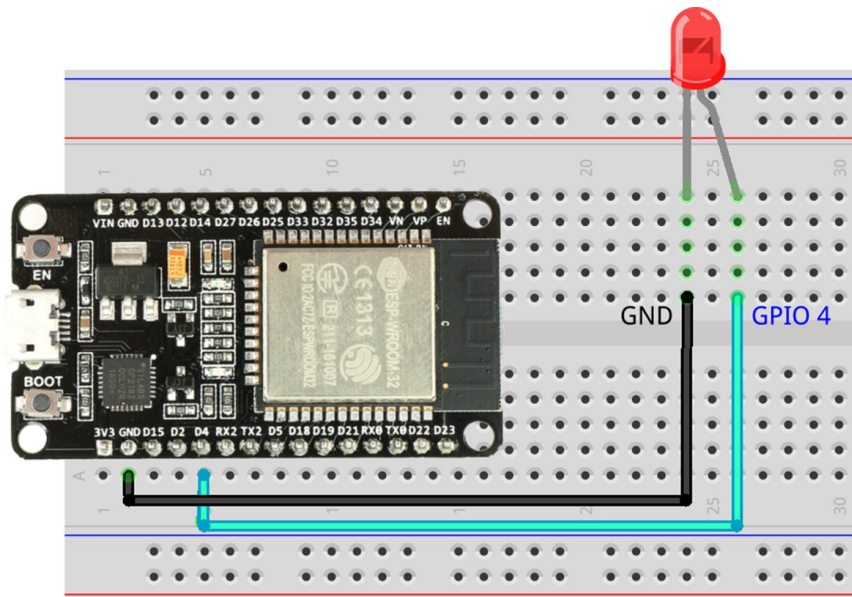
การต่อหลอด LED



EX2_LED_BLINK

ตัวอย่าง แสดงการควบคุมหลอดไฟ LED

ผลลัพธ์ที่ได้คือ หลอด LED จะติดเป็นเวลา 1 วินาทีแล้วดับ สลับกันไปไม่มีวันหยุด





```
Blink | Arduino 1.8.5

EX2_LED_BLINK

/*
  EX2_LED_BLINK
*/
int LEDPIN = 4;           // กำหนดขาที่เชื่อมต่อหลอด LED

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

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LEDPIN, HIGH); // เปิดไฟหลอด LED
  delay(1000);                // รอ 1 วินาที
  digitalWrite(LEDPIN, LOW);  // ปิดไฟหลอด LED
  delay(1000);                // รอ 1 วินาที
}

Done Saving.
Leaving...
Hard resetting...

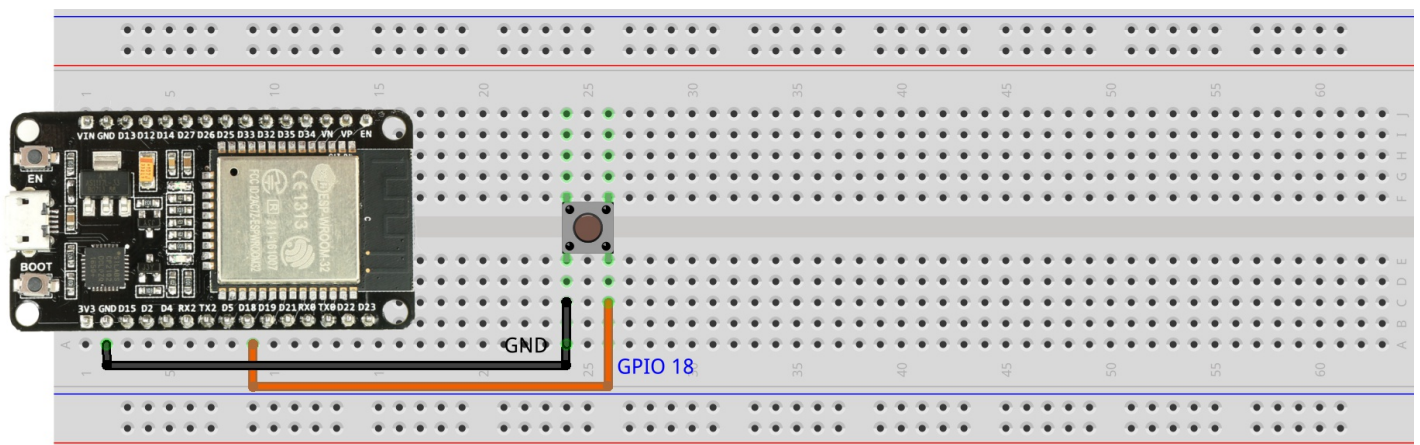
16 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART
```

EX3_READ_SWITCH

ตัวอย่าง แสดงการควบคุมหลอดไฟ LED

ผลลัพธ์ที่ได้คือ จะมีการอ่านค่าขา 2 และแสดงผลในคอนโซล

ถ้าไม่มีการกดสวิตช์ สถานะที่แสดงในคอนโซลจะเท่ากับ 1 แต่เมื่อมีการกดสวิตช์ สถานะจะเปลี่ยนเป็น 0



fritzing



```
EX3_READ_SWITCH | Arduino 1.8.5

EX3_READ_SWITCH
/*
  EX3_READ_SWITCH
*/
int SWPIN = 18; //กำหนดขาดีจิตอล 18
void setup() {

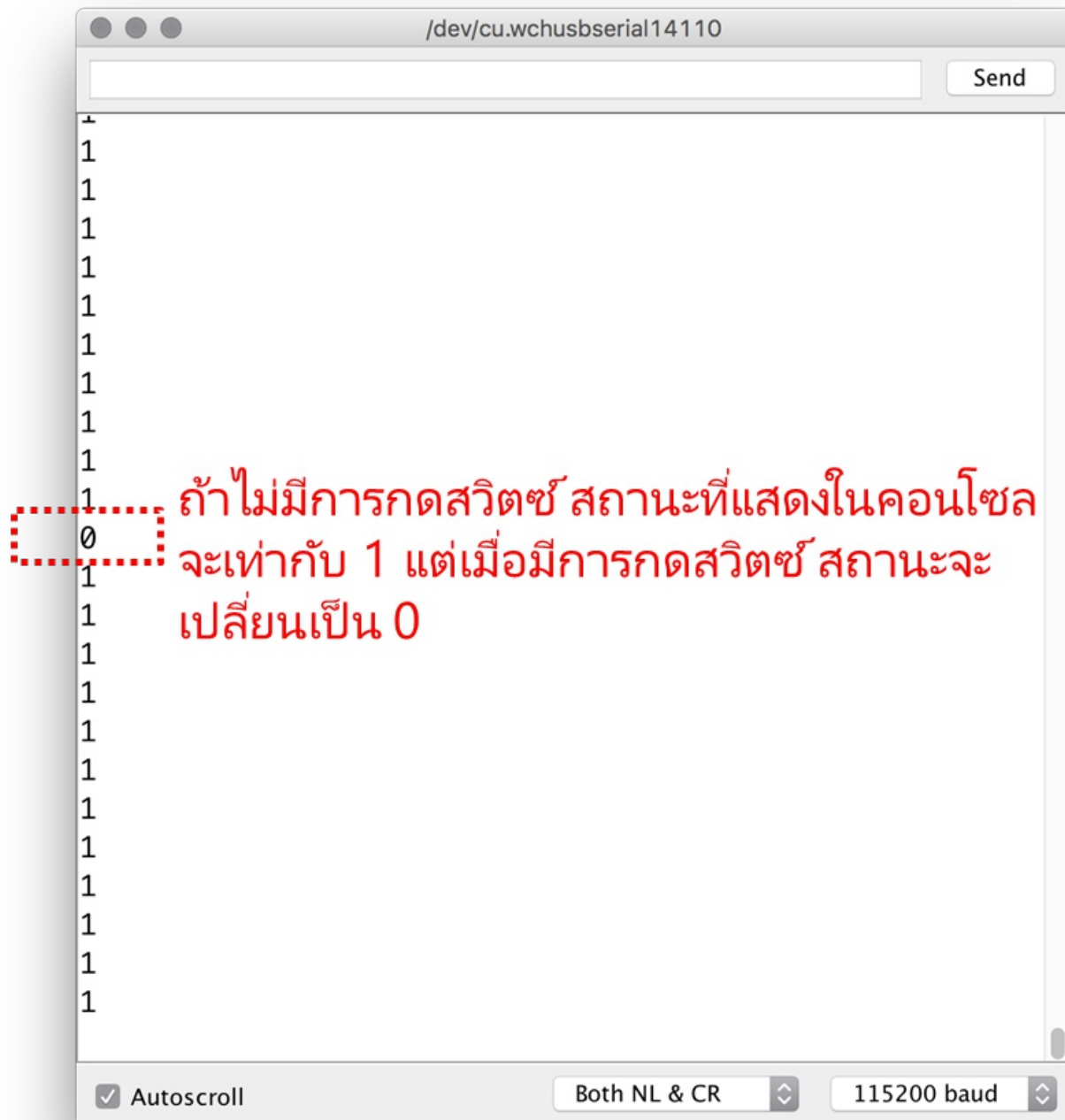
  Serial.begin(115200); //กำหนดค่าพอร์ตคอนโซล
  pinMode(SWPIN, INPUT_PULLUP); //กำหนดขาดีจิตอล 18 เป็นขาอ่านข้อมูลการกดสวิตช์
}

void loop() {

  int data = digitalRead(SWPIN); //อ่านค่าข้อมูลขาดีจิตอล 18 ในรูปแบบตัวเลข
  Serial.println(data); //แสดงค่าข้อมูลที่อ่านได้ในคอนโซล
  delay(100); //รอ 100 มิลลิวินาที
}

Done Saving.
Leaving...
Hard resetting...

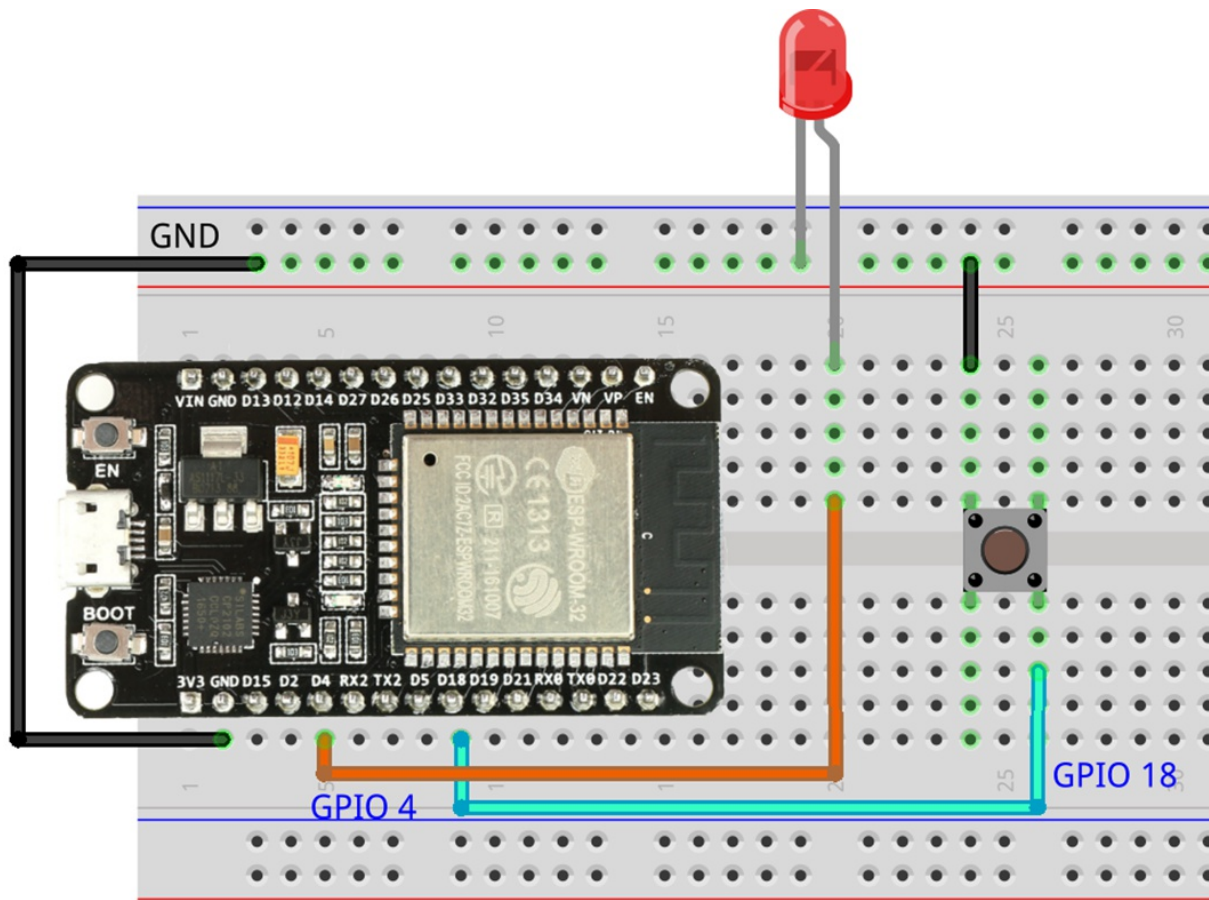
14 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART
```



EX4_READ_SWITCH2

ตัวอย่าง แสดงการควบคุมสวิตช์ดิจิทัล กดติดปล่อยดับ

ผลลัพธ์ที่ได้คือ ถ้ามีการกดสวิตช์หลอด LED จะติด ถ้าปล่อยหลอด LED จะดับ



EX4 สวิตช์ดิจิตอล กดติดปล่อยดับ



```
EX4_READ_SWITCH2 | Arduino 1.8.5

int LEDPIN = 4;
int SWPIN = 18;

void setup() {
  Serial.begin(115200);
  pinMode(LEDPIN, OUTPUT);           //กำหนดขาดิจิตอล LEDPIN เป็นขาเอาต์พุต (LED)
  pinMode(SWPIN, INPUT_PULLUP);     //กำหนดขาดิจิตอล SWPIN เป็นขาอ่านข้อมูลการกดสวิตช์
}

void loop() {
  boolean data = digitalRead(SWPIN); //อ่านค่าข้อมูลขาดิจิตอล SWPIN ในรูปแบบตัวเลข

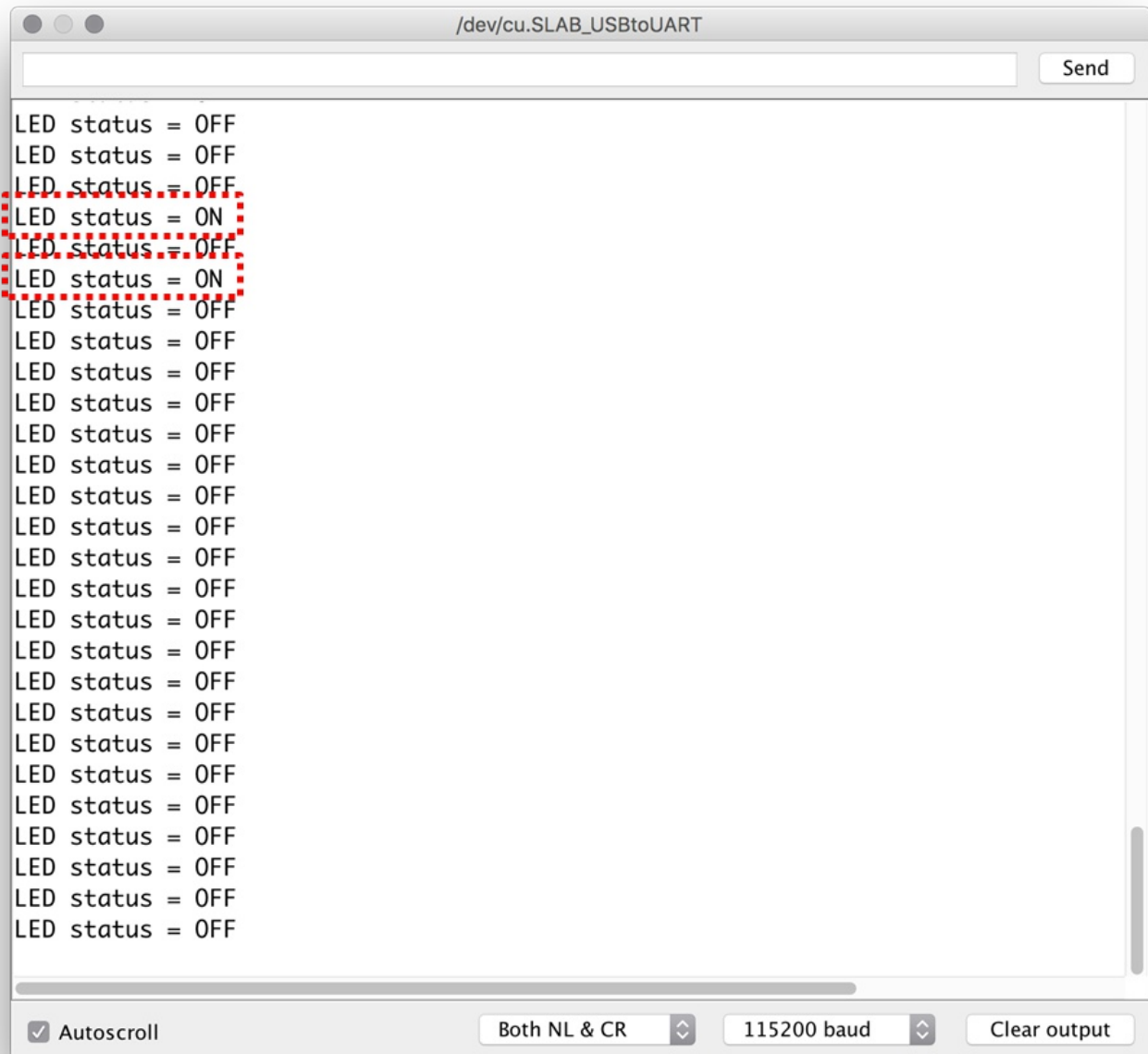
  if (data == 0) {
    digitalWrite(LEDPIN, HIGH);
    Serial.println("LED ON");
  } else {
    digitalWrite(LEDPIN, LOW);
    Serial.println("LED OFF");
  }
}
```

Done uploading.

Leaving...

Hard resetting...

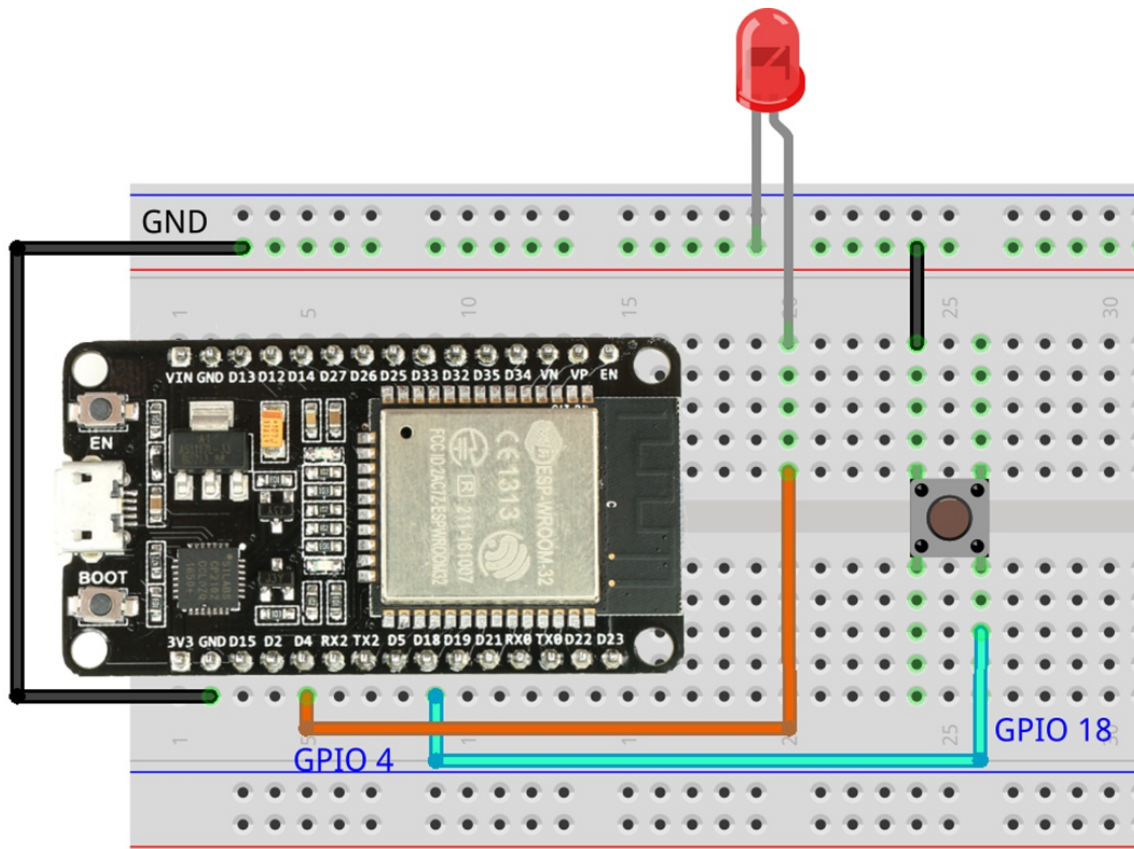
17 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART



EX5_READ_SWITCH3

ตัวอย่าง แสดงการควบคุมสวิตช์ดิจิทัล กดติด กดดับ

ผลลัพธ์ที่ได้คือ ถ้ามีการกดสวิตช์หลอด LED จะติด ถ้ากดอีกครั้งหลอด LED จะดับ



EX5 สวิตช์ดิจิทัล กดติด กดดับ



```
EX4_READ_SWITCH2 | Arduino 1.8.5

int LEDPIN = 4;
int SWPIN = 18;

void setup() {
  Serial.begin(115200);
  pinMode(LEDPIN, OUTPUT);           //กำหนดขาดีจิตอล LEDPIN เป็นขาเอาต์พุต (LED)
  pinMode(SWPIN, INPUT_PULLUP);     //กำหนดขาดีจิตอล SWPIN เป็นขาอ่านข้อมูลการกดสวิตช์
}

void loop() {
  boolean data = digitalRead(SWPIN); //อ่านค่าข้อมูลขาดีจิตอล SWPIN ในรูปแบบตัวเลข

  if (data == 0) {
    digitalWrite(LEDPIN, HIGH);
    Serial.println("LED ON");
  } else {
    digitalWrite(LEDPIN, LOW);
    Serial.println("LED OFF");
  }
}
```

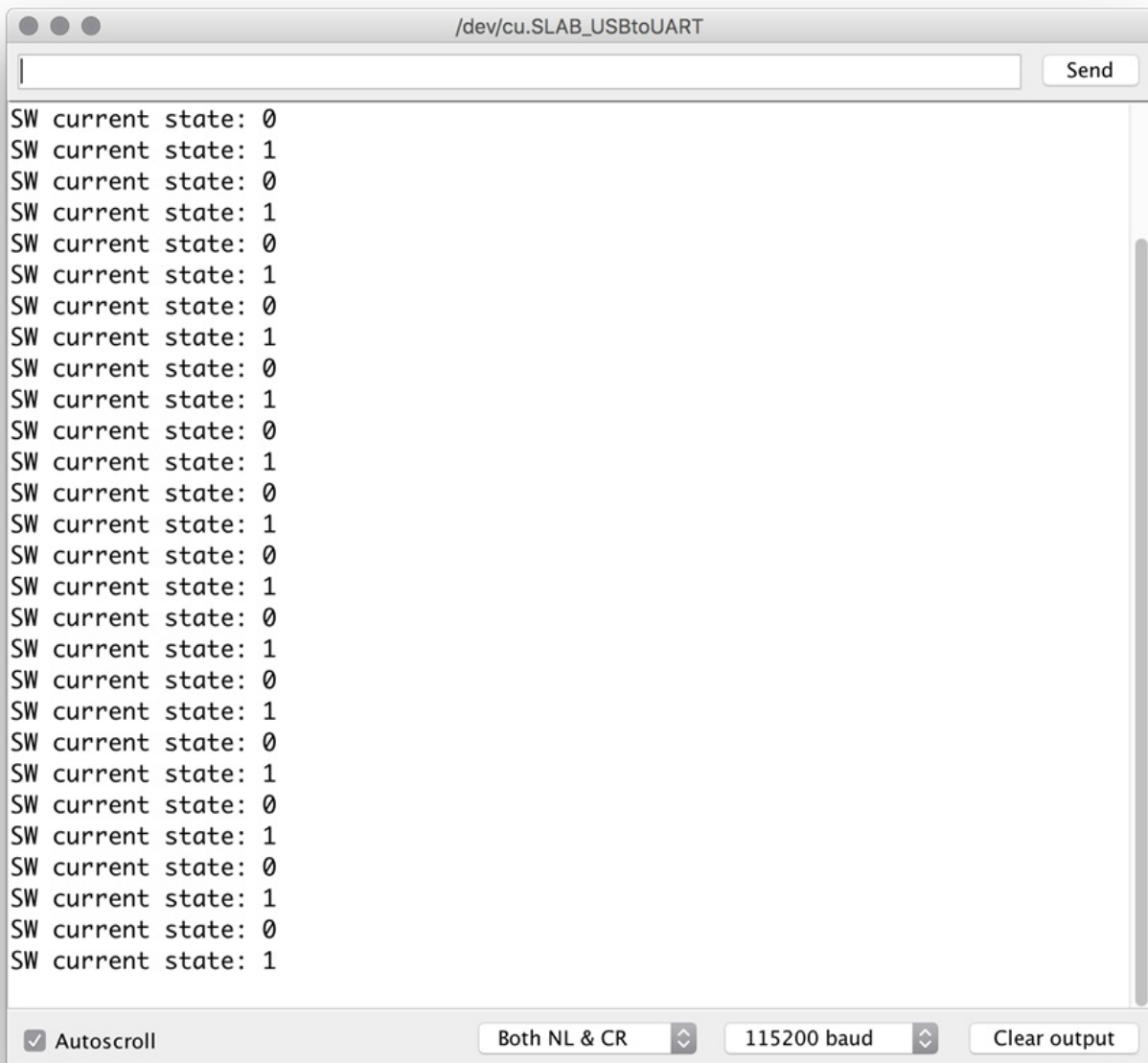
Done uploading.

Leaving...

Hard resetting...

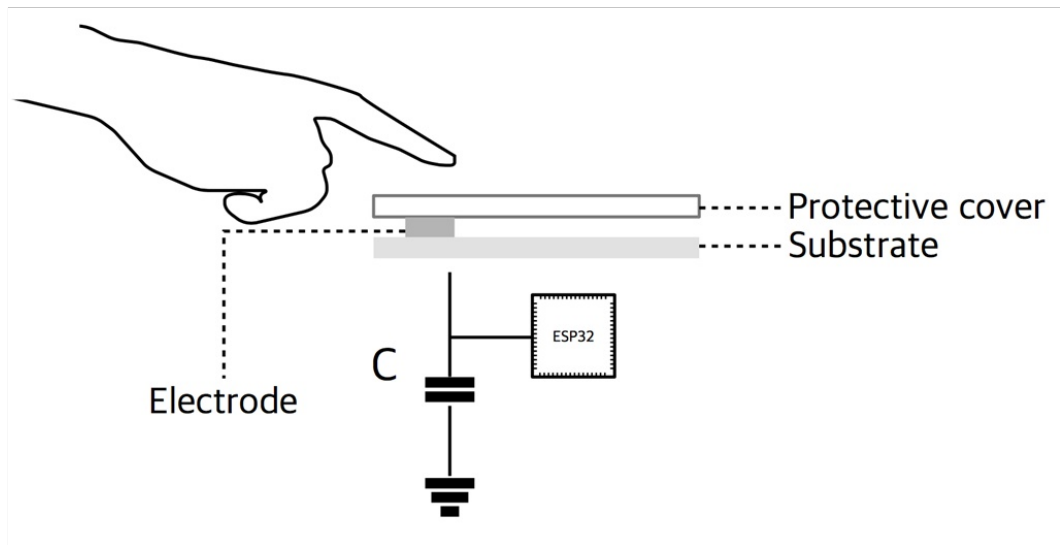
17 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

EX5 สวิตช์ดิจิตอล กดติด กดดับ



Touch sensor

A touch sensor system is built on a substrate which carries electrodes and relevant connections with a flat protective surface. When a user touches the surface, the capacitance variation is triggered, and a binary signal is generated to indicate whether the touch is valid.



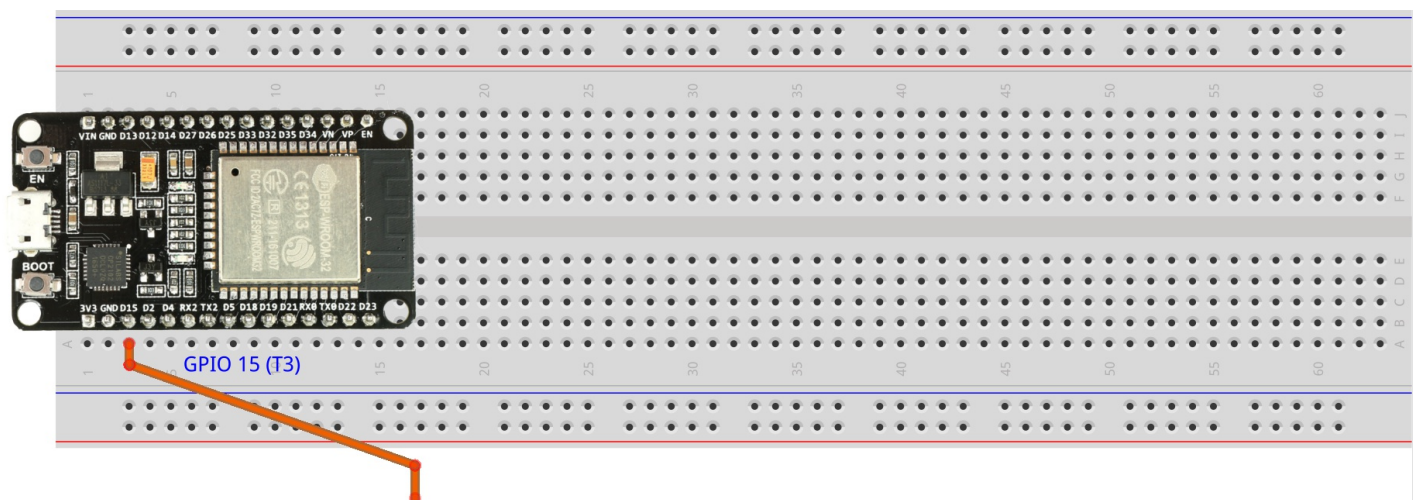
EX6_TOUCH_SENSOR

ตัวอย่างทดสอบการอ่านค่าเซ็นเซอร์สัมผัสที่มากับ ESP32 และแสดงผลในคอนโซล

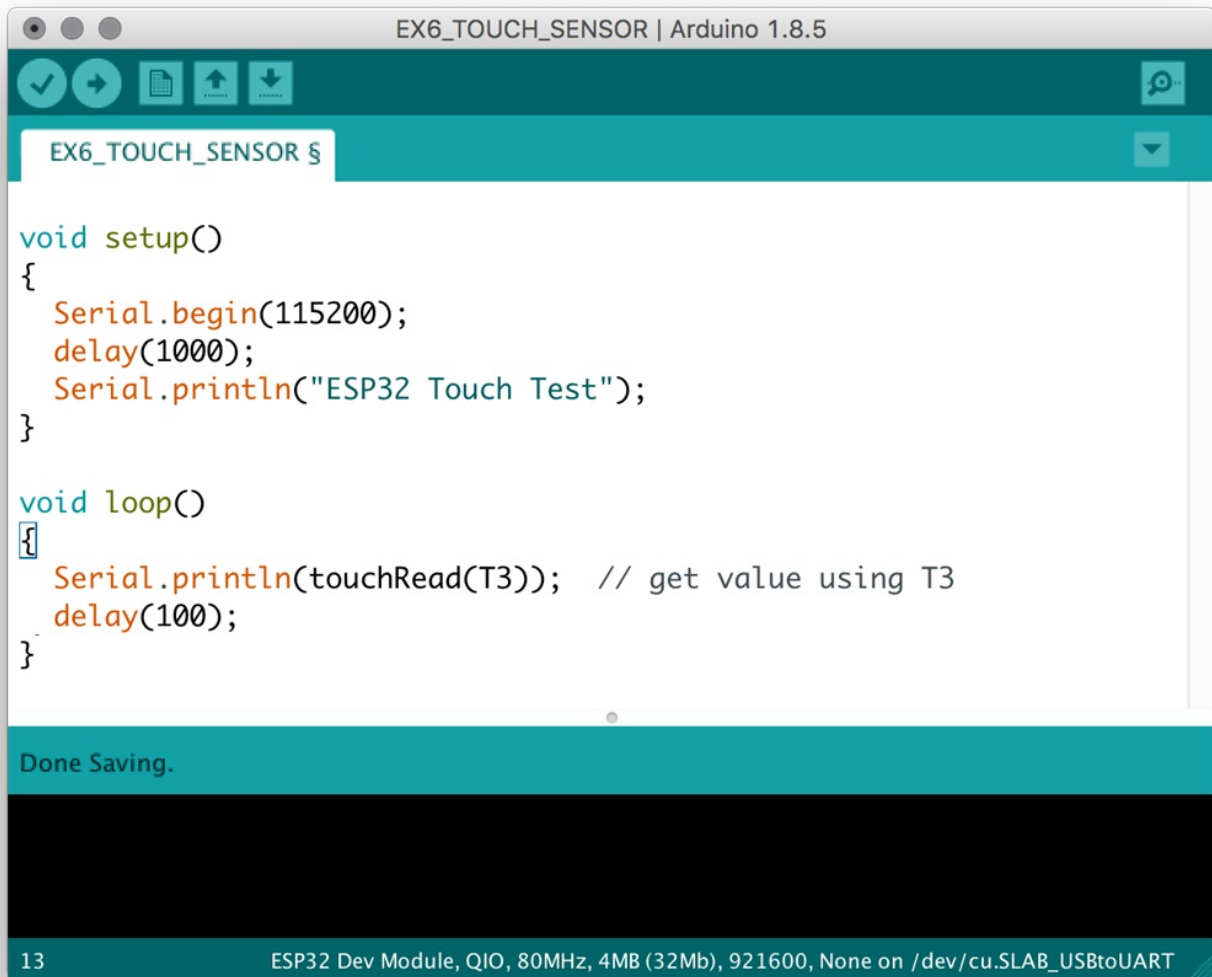
ผลลัพธ์ที่ได้คือ จะมีการพล็อตกราฟใน Serial Plotter

ขาที่ต่อทดสอบ คือ

TOUCH3 (GPIO15)



fritzing



```
EX6_TOUCH_SENSOR | Arduino 1.8.5

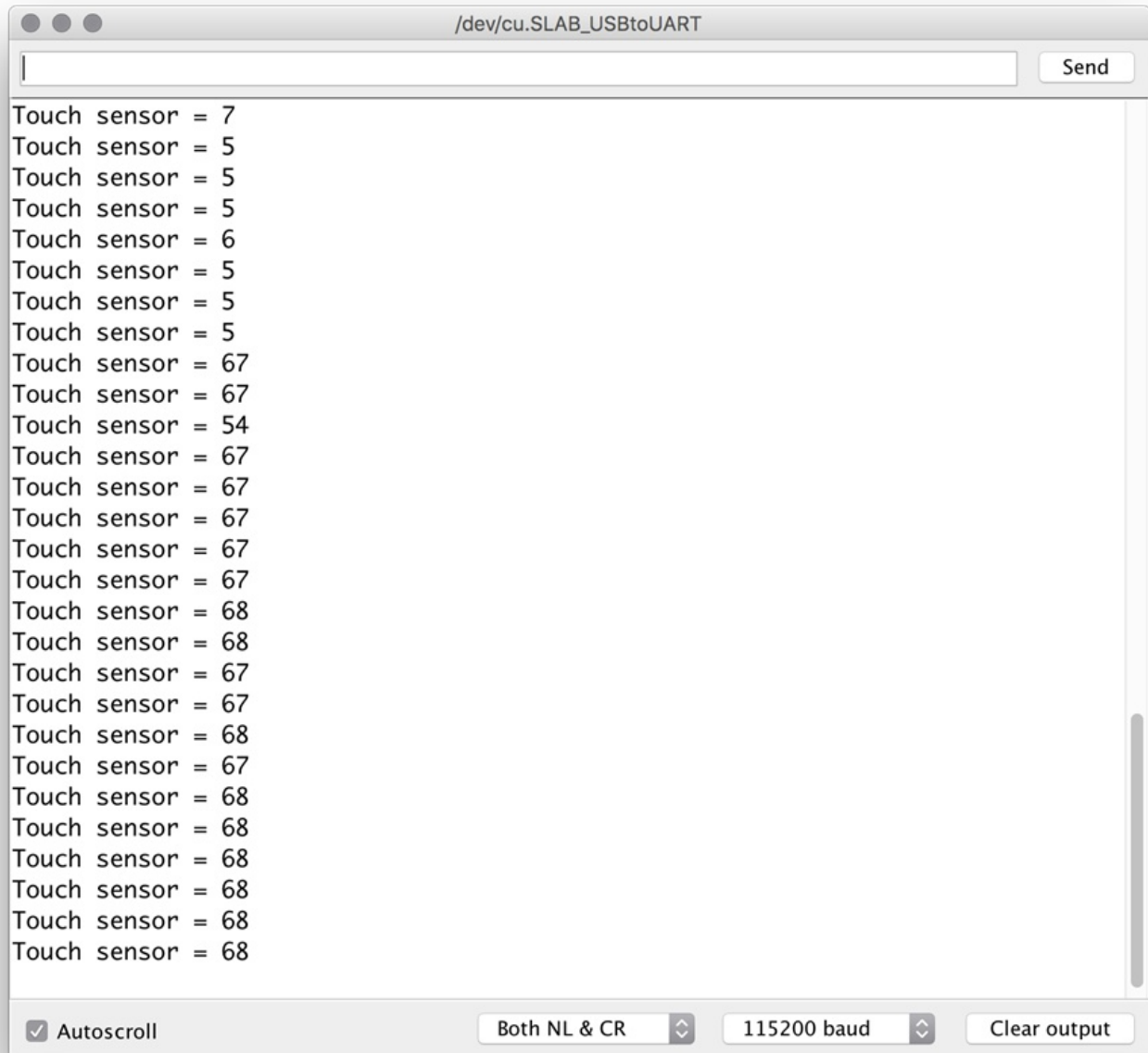
EX6_TOUCH_SENSOR $

void setup()
{
  Serial.begin(115200);
  delay(1000);
  Serial.println("ESP32 Touch Test");
}

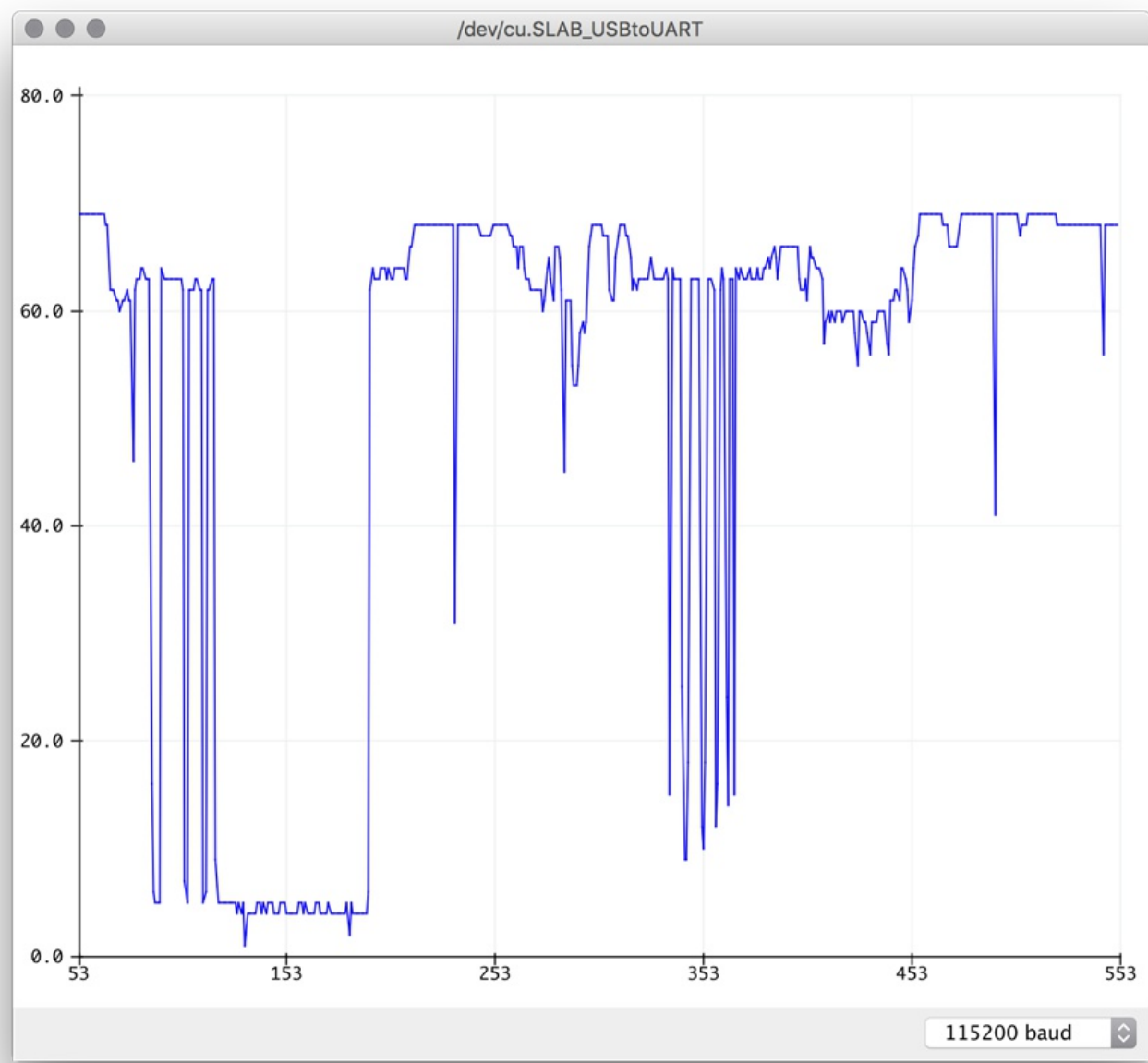
void loop()
{
  Serial.println(touchRead(T3)); // get value using T3
  delay(100);
}

Done Saving.

13 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART
```



EX6 เซ็นเซอร์สัมผัส



รู้จักเซ็นเซอร์ประเภทต่างๆ

สำหรับ ESP32



101

เซ็นเซอร์



Piranha LED



Sound Sensor



photo interrupter



PIR Motion



IR Receiver



Bluetooth Module



White LED



Rotation Sensor



Capacitive touch



Flame Sensor



Rotary Encoder



TEMT6000



3W LED



Passive Buzzer



Knock Sensor



Vibration Sensor



LM35



HC-SR04



RGB LED



Active Buzzer



Hall Magnetic



Gas Sensor



18B20



Joystick



Analog temperature



Push Button



Line Tracking



Alcohol Sensor



ADXL345



DS3231



Photocell sensor



Tilt Sensor



IR Obstacle Avoidance



IR Transmitter



DHT11



5V Relay

เซ็นเซอร์วัดคุณภาพน้ำ



Conductivity sensor



pH sensor



Nitrate Ion (NO₃⁻) Sensor Probe



Chloride Ion (Cl⁻) Sensor Probe



Double Junction Reference Probe



Water Leakage / Liquid Detection
Sensor (Point)



Fluoride Ion (F⁻) Sensor Probe



Fluoroborate Ion (BF₄⁻) Sensor Probe



Bromide Ion (Br⁻) Sensor Probe

เซ็นเซอร์สำหรับการเกษตร



Solar radiation sensor



Watermark Soil Moisture Sensor 1,5m



Platinum thermal resistance sensor



Weather Station WS-3000



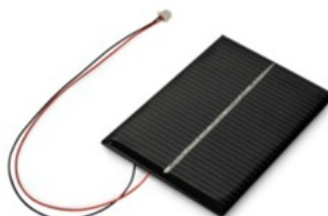
Soil Moisture Probe for Open Garden



Watermark Soil Moisture Sensor 8m



Leaf Wetness Sensor



Solar Panel 70x50mm for Open Garden



Digital Humidity Sensor (RH&T)

เซ็นเซอร์เกี่ยวกับสุขภาพ



Spirometer Air Capacity Sensor PRO for MySignals (eHealth Medical Development Platform)



Mindwave EEG Electroencephalogram Sensor PRO for MySignals HW (eHealth Medical Development Platform)



e-Health Sensor Platform Complete Kit V2.0 for Arduino, Raspberry Pi and Intel Galileo [Biometric /



Electrocardiogram Sensor (ECG) for e-Health Platform [Biometric / Medical Applications]



Electromyography Sensor (EMG) for e-Health Platform [Biometric / Medical Applications]



EMG Disposable Electrodes Pack for e-Health Platform [Biometric / Medical Applications]



EMG Electromyography Sensor PRO for MySignals (eHealth Medical Development Platform)

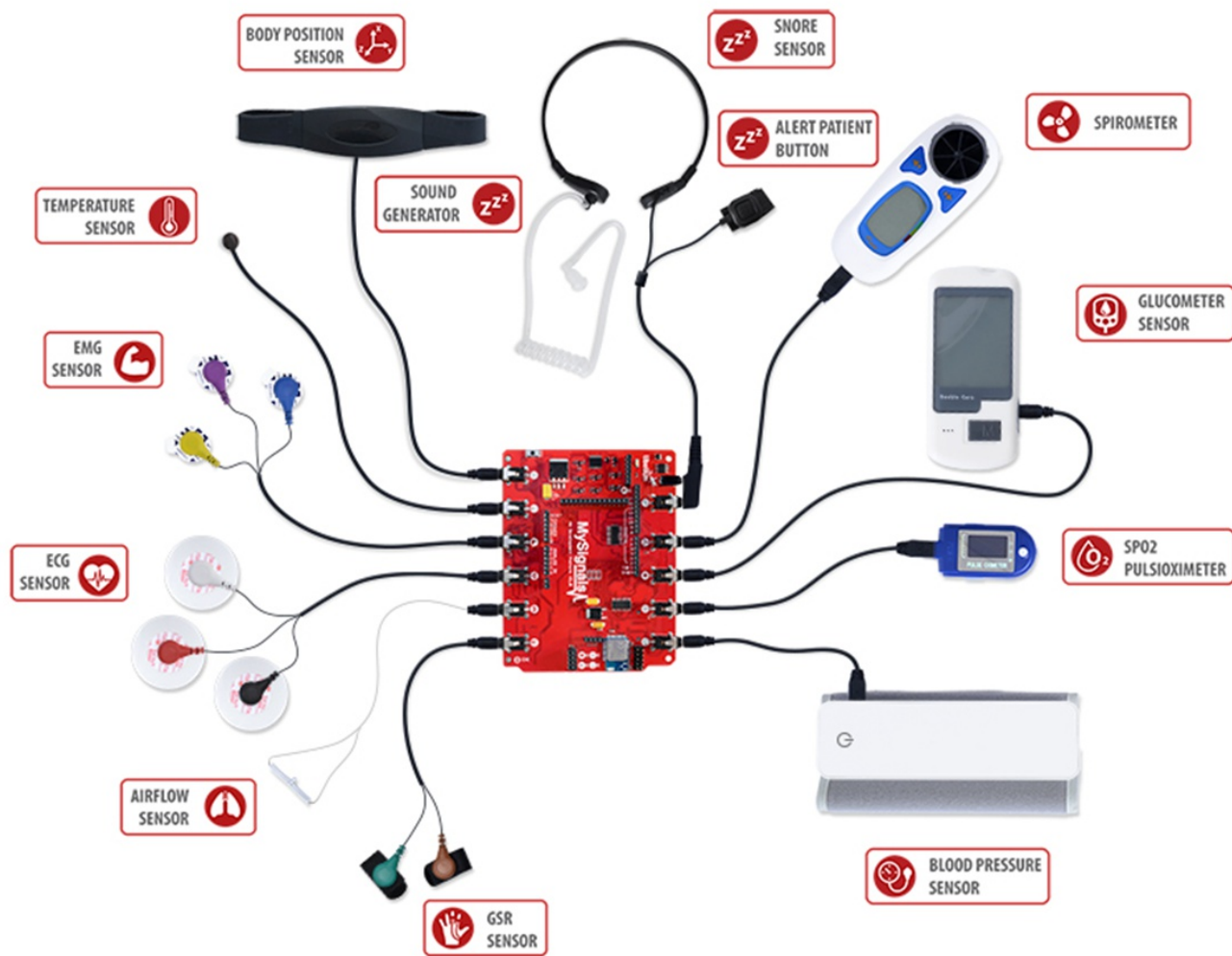


GSR Galvanic Skin Response Sensor PRO for MySignals (eHealth Medical Development Platform)



Body Position Sensor PRO for MySignals (eHealth Medical Development Platform)

เซ็นเซอร์เกี่ยวกับสุขภาพ



เซ็นเซอร์เกี่ยวกับการวัดแก๊ส



Calibrated Hydrogen Sulfide (H₂S) gas sensor



Calibrated Ethylene (ETO) gas sensor



Calibrated Hydrogen Chloride (HCl) gas sensor



Calibrated Carbon Dioxide (CO₂) gas sensor



Alcohol Derivatives Gas Sensor



Calibrated Chlorine (Cl₂) gas sensor



Calibrated Ozone (O₃) gas sensor



NO₂ Gas Sensor



Calibrated Carbon Monoxide (CO) gas sensor

เซ็นเซอร์เกี่ยวกับการวัดทั่วไป



Hall-Effect Sensor (doors and windows)



Vibration Film Sensor for thread



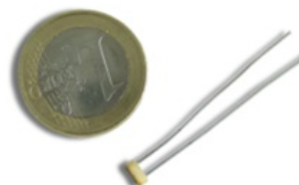
Vibration Film Sensor



Ultraviolet sensor



Luminosity (luxes accuracy)



LDR Sensor



Water flow large

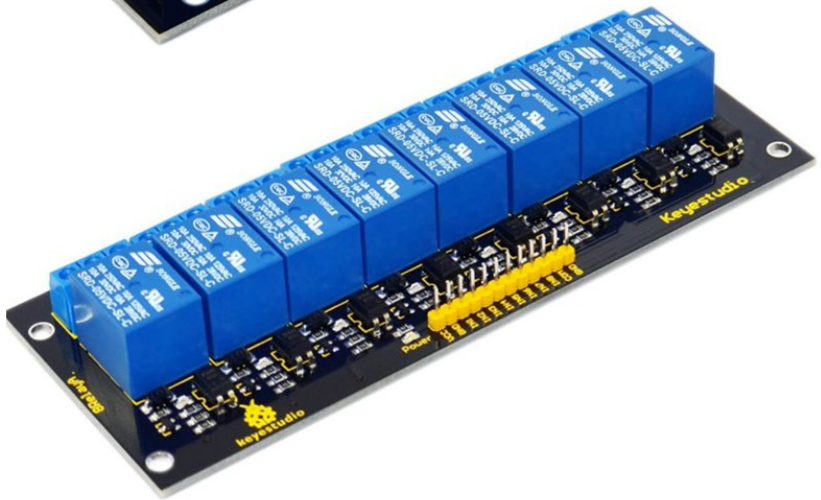
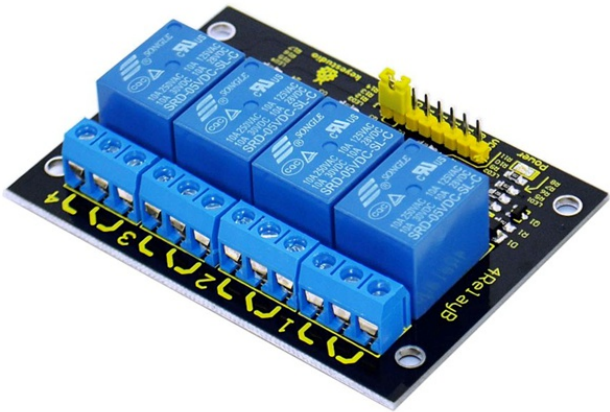
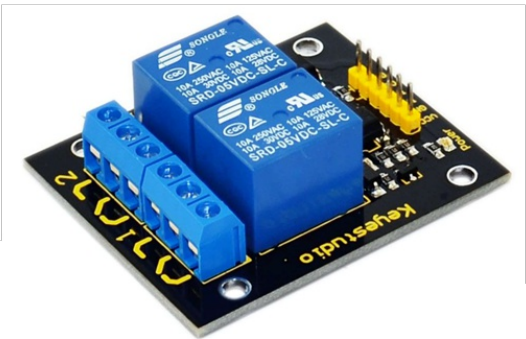


Water flow medium



Water flow small

รีเลย์



การเขียนโปรแกรมใช้งานพอร์ต อนาล็อก GPIO

ภาคปฏิบัติ



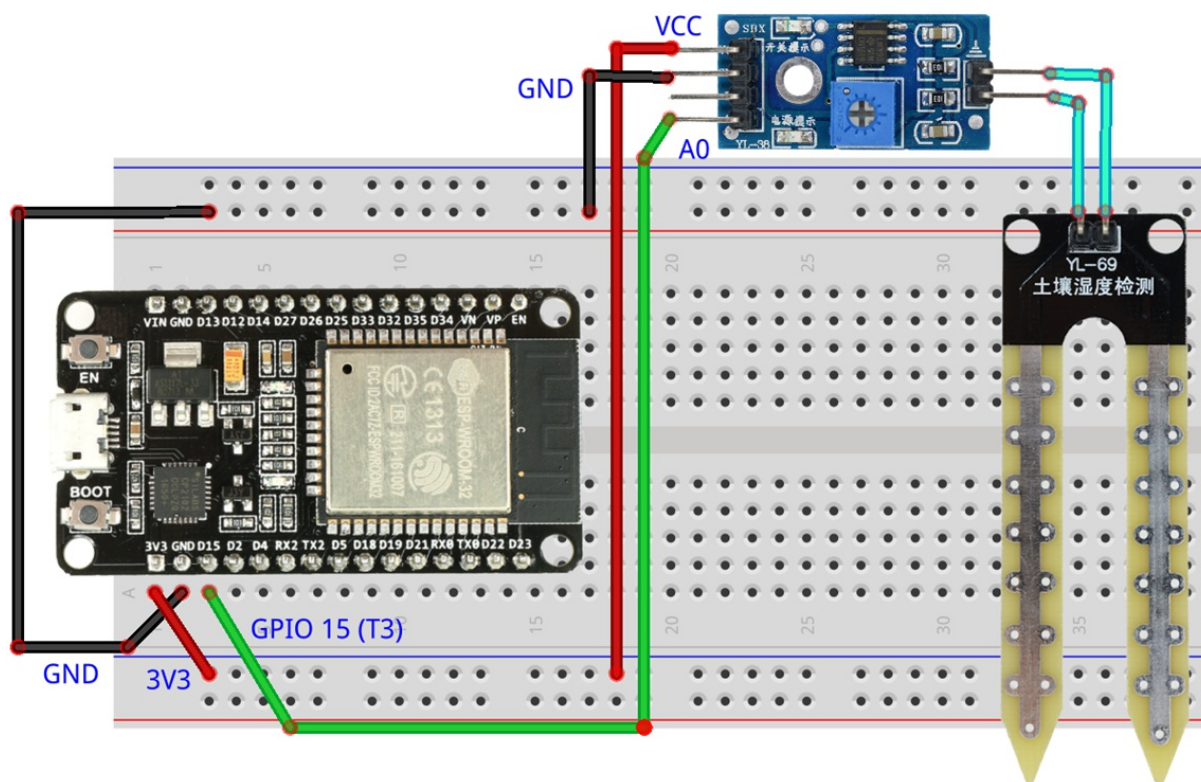
EX7_SOIL_SENSOR

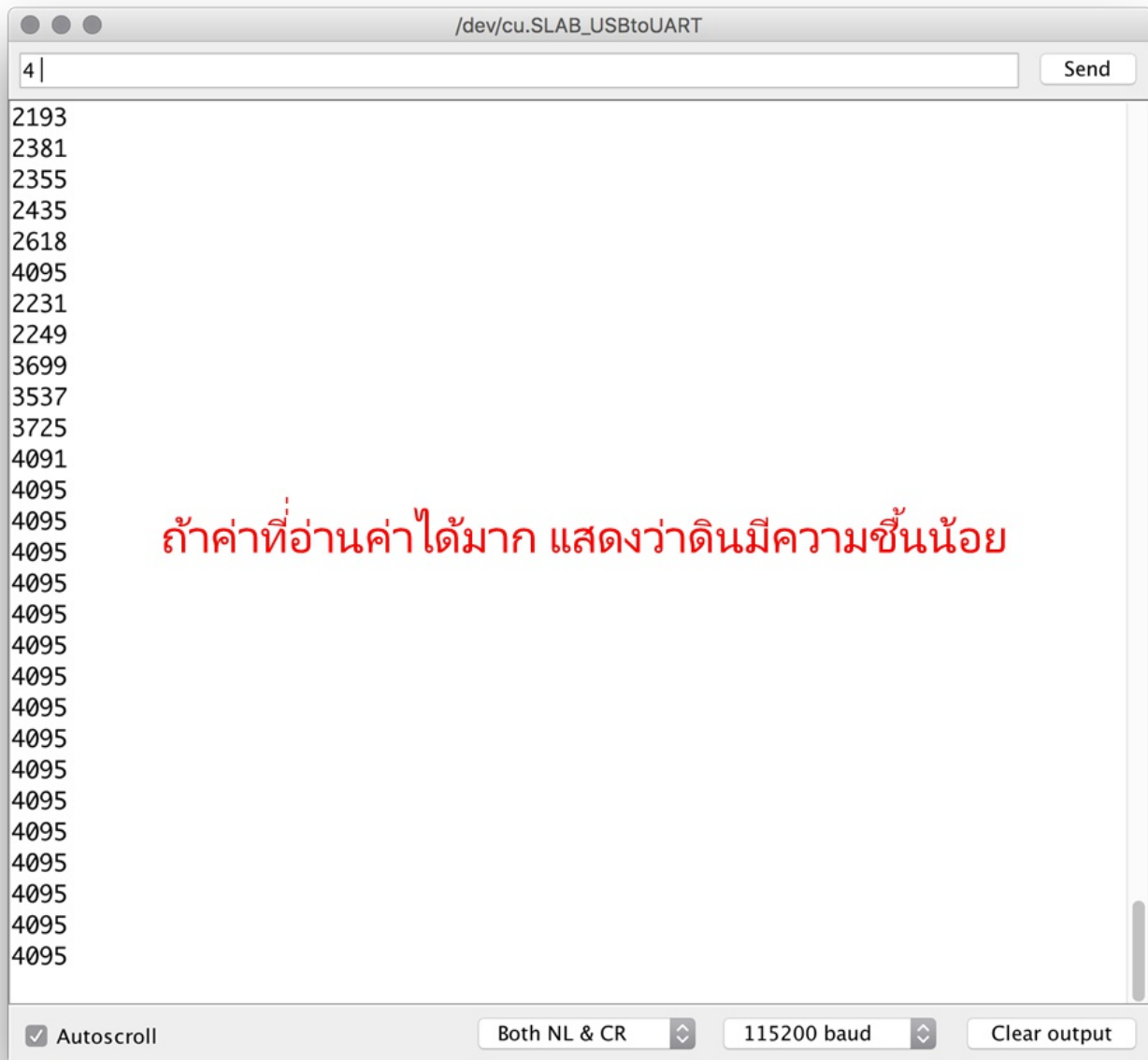
ตัวอย่างทดสอบการอ่านค่าความชื้นในดิน ซึ่งเป็นการอ่านค่าแบบอนาล็อกและแสดงผลในคอนโซล

ผลลัพธ์ที่ได้คือ จะมีการพล็อตกราฟใน Serial Plotter

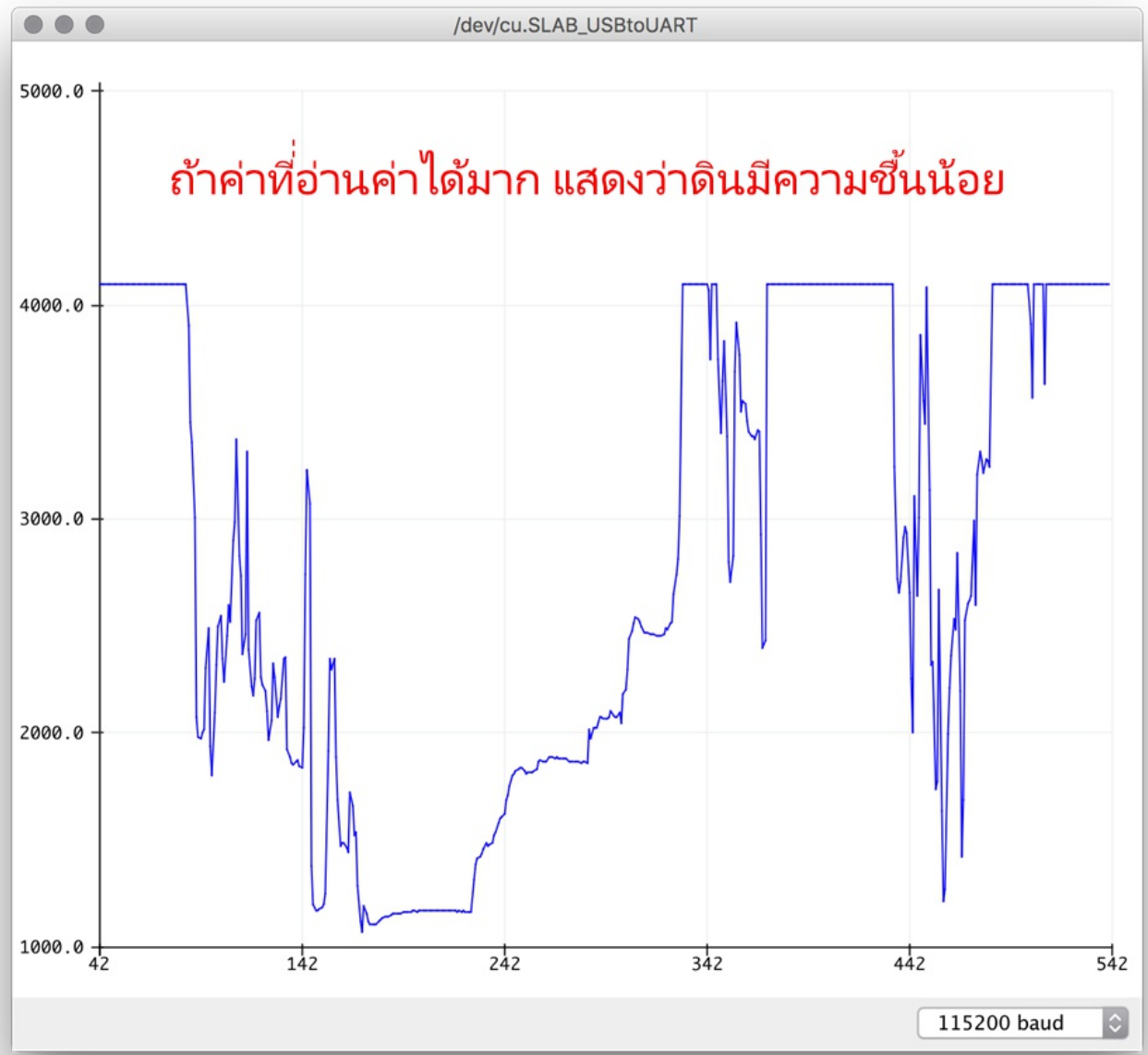
ขาที่ต่อทดสอบ คือ

GPIO15 (ADC2_CH3)





EX7 วัดค่าความชื้นดิน



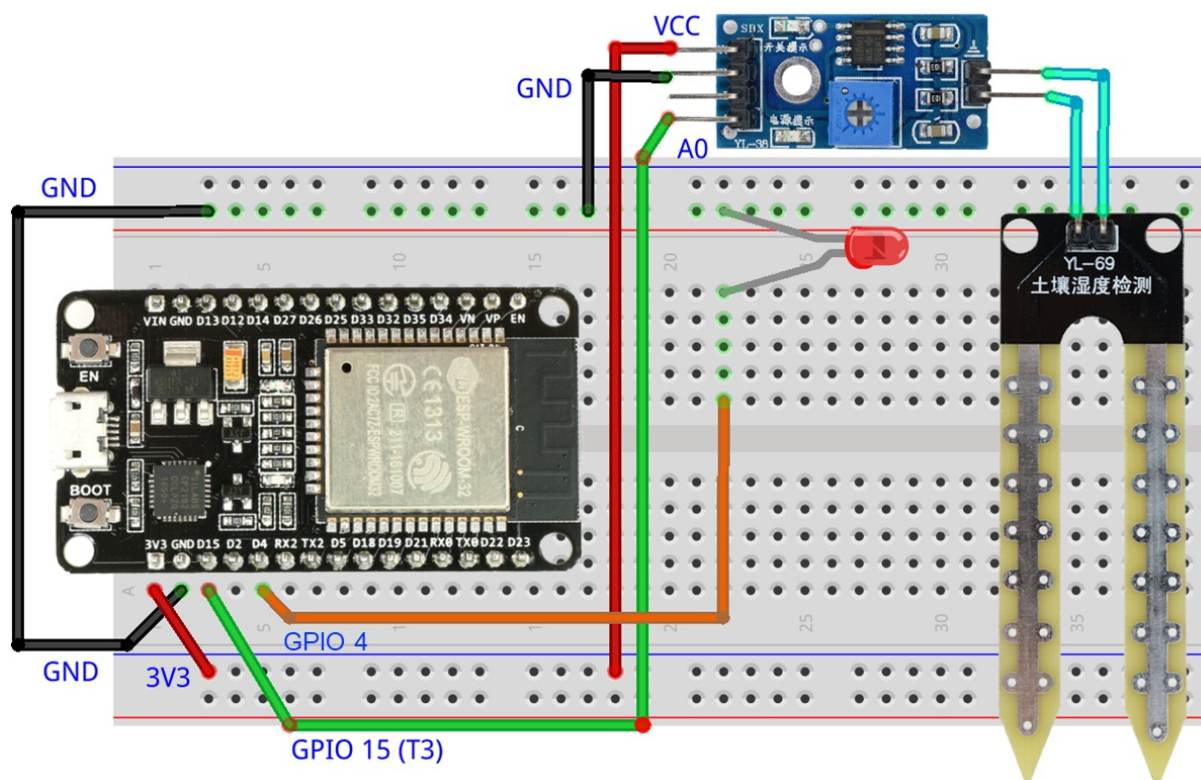
EX8_SOIL_SENSOR_LED_STATUS

ตัวอย่างทดสอบการอ่านค่าความชื้นในดิน ซึ่งเป็นการอ่านค่าแบบอนาล็อกและแสดงผลในคอนโซล

ผลลัพธ์ที่ได้คือ จะมีการพล็อตกราฟใน Serial Plotter และถ้าค่าความชื้นดินมากกว่า 1300 (ดินแห้ง) หลอดไฟ LED จะติด แต่ถ้าน้อยกว่าหลอดไฟ LED จะดับ (ดินชุ่ม)

ขาที่ต่อทดสอบ คือ

GPIO15 (ADC2_CH3) //Soil sensor
GPIO4 //LED



EX8 วัดค่าความชื้นดินและแสดงสถานะบนหลอด LED

```

EX8_SOIL_SENSOR_LED_STATUS
int LEDPIN = 4;
int SOILSENSOR = 15;           //GPIO15 (ADC2_CH3)   //Soil sensor

void setup() {
  Serial.begin(115200);
  pinMode(LEDPIN, OUTPUT);     //กำหนดขา LEDPIN เป็นขาออกที่เชื่อมต่อ LED
}

void loop() {
  int value = analogRead(SOILSENSOR); //อ่านค่าความชื้นดิน
  if (value < 1300) {           //ถ้าค่าความชื้นดินมากกว่า 1200 (ชื้นมาก)
    digitalWrite(LEDPIN, LOW);  //ปิดไฟ LED
    Serial.println("NORMAL STATUS");
  } else {
    digitalWrite(LEDPIN, HIGH); //เปิดไฟ LED
    Serial.println("WATER INJECT!");
  }

  Serial.print("Soil sensor values = ");
  Serial.println(value);
  delay(1000);                 //รอ 1 วินาที
}

```

Done uploading.
Leaving...
Hard resetting...

24 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

```

4
NORMAL STATUS
Soil sensor values = 1164
NORMAL STATUS
Soil sensor values = 1164
NORMAL STATUS
Soil sensor values = 1164
NORMAL STATUS
Soil sensor values = 1160
NORMAL STATUS
Soil sensor values = 1260
NORMAL STATUS
Soil sensor values = 1056
NORMAL STATUS
Soil sensor values = 1160
NORMAL STATUS
Soil sensor values = 1162
NORMAL STATUS
Soil sensor values = 1159
NORMAL STATUS
Soil sensor values = 1159
NORMAL STATUS
Soil sensor values = 1157
NORMAL STATUS
Soil sensor values = 1157
NORMAL STATUS
Soil sensor values = 1155
NORMAL STATUS
Soil sensor values = 1156

```

Autoscroll Both NL & CR 115200 baud Clear output

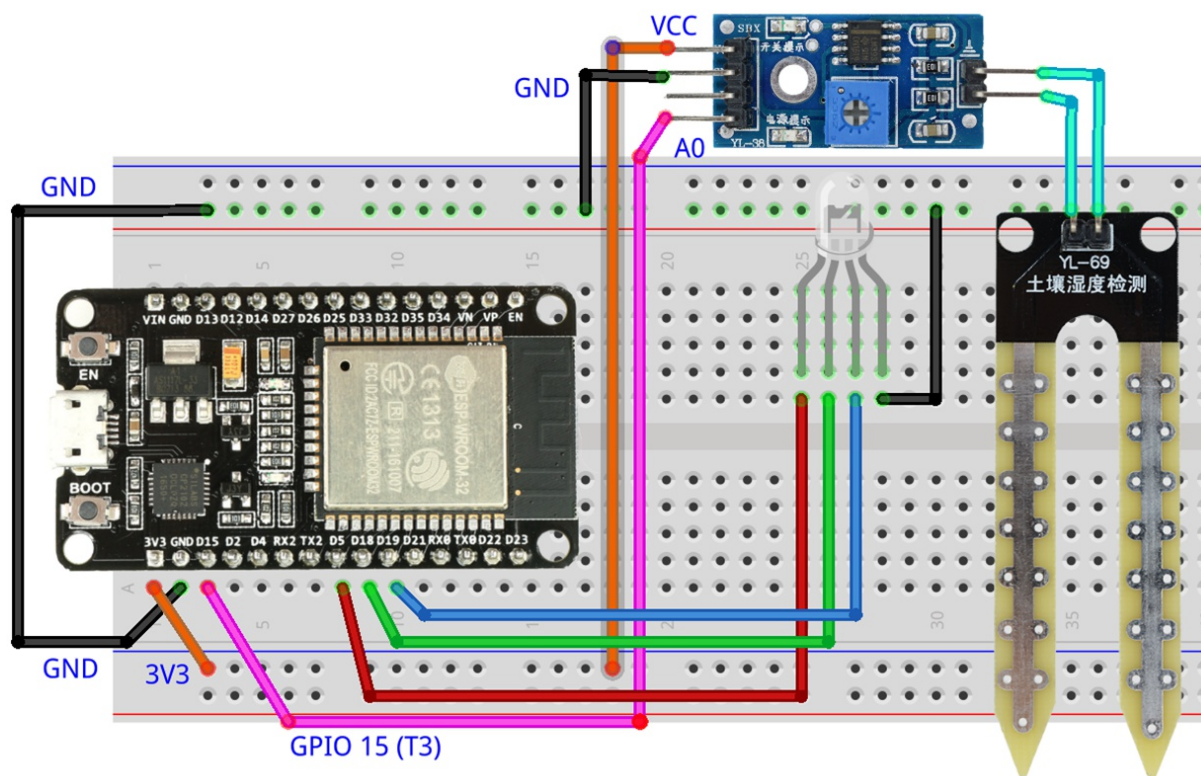
EX9_SOIL_SENSOR_RGBLED_STATUS

ตัวอย่างทดสอบการอ่านค่าความชื้นในดิน ซึ่งเป็นการอ่านค่าแบบอนาล็อกและแสดงผลในคอนโซล

ผลลัพธ์ที่ได้คือ จะมีการพล็อตกราฟใน Serial Plotter และถ้าค่าความชื้นดินน้อยกว่า 1000 (ดินเปียก) หลอดไฟ LED สีเขียวจะติด ถ้าค่ามากกว่า 2000 (ดินชุ่ม) หลอดไฟ LED สีน้ำเงินจะติด แต่ถ้าค่ามากกว่า 3000 (ดินแห้ง) หลอดไฟสีแดงจะติด

ขาที่ต่อทดสอบ คือ

GPIO15 (ADC2_CH3)//Soil sensor



EX9 วัดค่าความชื้นดินและแสดงสถานะบนหลอด RGB LED

```

EX9_SOIL_SENSOR_RGBLED_STATUS | Arduino 1.8.5

EX9_SOIL_SENSOR_RGBLED_STATUS §
int R_LEDPIN = 5;
int G_LEDPIN = 18;
int B_LEDPIN = 19;
int SOILSENSOR = 15;                                //GPIO15 (ADC2_CH3)    //Soil sensor

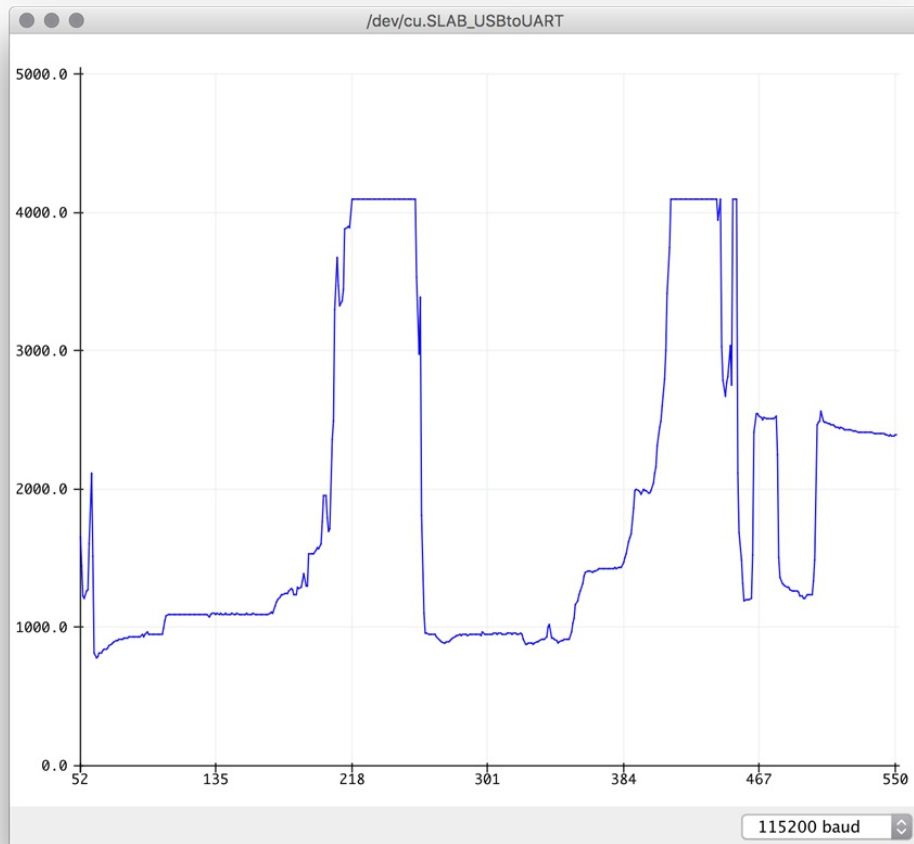
void setup() {
  Serial.begin(115200);
  pinMode(R_LEDPIN, OUTPUT);                          //กำหนดขา R_LEDPIN เป็นขาออกที่เชื่อมต่อ LED RED
  pinMode(G_LEDPIN, OUTPUT);                          //กำหนดขา G_LEDPIN เป็นขาออกที่เชื่อมต่อ LED GREEN
  pinMode(B_LEDPIN, OUTPUT);                          //กำหนดขา B_LEDPIN เป็นขาออกที่เชื่อมต่อ LED BLUE
}

void loop() {
  int value = analogRead(SOILSENSOR);                  //อ่านค่าความชื้นดิน
  if (value < 1000) {                                  //ถ้าค่าความชื้นดินมากกว่า 1000 (ดินเปียกมาก)
    digitalWrite(R_LEDPIN, LOW); digitalWrite(G_LEDPIN, HIGH); digitalWrite(B_LEDPIN, LOW); //เปิดไฟ LED สีเขียว
  } else if (value >= 1000 and value <= 3000) {        //ถ้าค่าความชื้นดินอยู่ระหว่าง 1000 และ 3000 (ดินชุ่ม)
    digitalWrite(R_LEDPIN, LOW); digitalWrite(G_LEDPIN, LOW); digitalWrite(B_LEDPIN, HIGH); //เปิดไฟ LED สีน้ำเงิน
  } else if (value > 3000) {                            //ถ้าค่าความชื้นดินมากกว่า 3000 (ดินแห้ง)
    digitalWrite(R_LEDPIN, HIGH); digitalWrite(G_LEDPIN, LOW); digitalWrite(B_LEDPIN, LOW); //เปิดไฟ LED สีแดง
  }
  Serial.println(value);
  delay(100);                                          //รอ 1 วินาที
}

Done Saving.
Leaving...
Hard resetting...

14 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

```



EX10_SOIL_SENSOR_RGBLED_CALIBRATE

ตัวอย่างทดสอบอ่านค่าความชื้นในดินในรูปแบบเปอร์เซ็นต์ที่เทียบค่าแล้ว ซึ่งเป็นการอ่านค่าแบบอนาล็อกและแสดงผลในคอนโซล

ผลลัพธ์ที่ได้คือ ถ้าค่าความชื้นดินมากกว่า 80% (ดินเปียก) หลอดไฟ LED สีเขียวจะติด

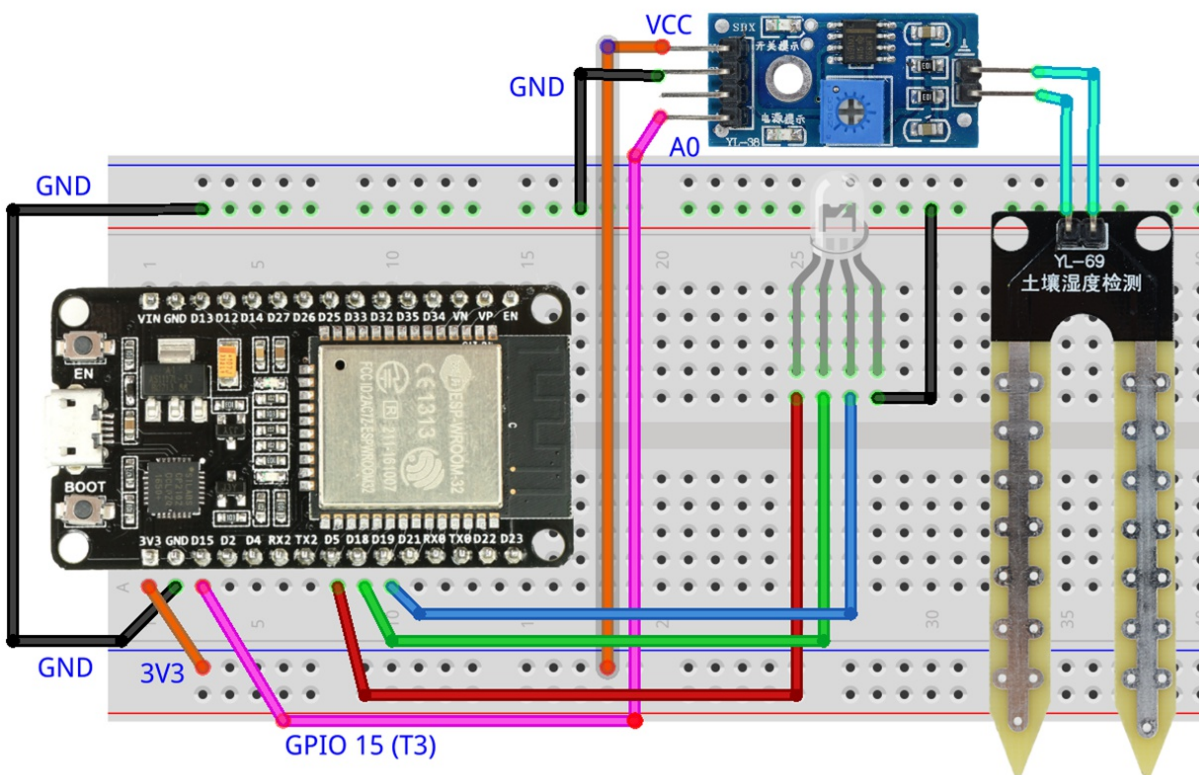
ถ้าค่ามากกว่า 40% (ดินชุ่ม) หลอดไฟ LED สีน้ำเงินจะติด

แต่ถ้าค่าน้อยกว่า 40% (ดินแห้ง) หลอดไฟสีแดงจะติด

ขาที่ต่อทดสอบ คือ

GPIO15 (ADC2_CH3) //Soil sensor

GPIO5 / GPIO18 / GPIO19 //LED



EX10_SOIL_SENSOR_RGBLED_CALIBRATE

```

EX10_SOIL_SENSOR_RGBLED_CALIBRATE | Arduino 1.8.5

int R_LEDPIN = 5;
int G_LEDPIN = 18;
int B_LEDPIN = 19;
int SOILSENSOR = 15;                                //GPIO15 (ADC2_CH3)    //Soil sensor

void setup() {
  Serial.begin(115200);
  pinMode(R_LEDPIN, OUTPUT);                          //กำหนดขา R_LEDPIN เป็นขาออกที่เชื่อมต่อ LED RED
  pinMode(G_LEDPIN, OUTPUT);                          //กำหนดขา G_LEDPIN เป็นขาออกที่เชื่อมต่อ LED GREEN
  pinMode(B_LEDPIN, OUTPUT);                          //กำหนดขา B_LEDPIN เป็นขาออกที่เชื่อมต่อ LED BLUE
}

void loop() {
  int value = analogRead(SOILSENSOR);                  //อ่านค่าความชื้นดิน

  if (value < 900) {                                    //ให้ค่าที่อ่านได้น้อยกว่า 900 เท่ากับ 900
    value = 900;
  } else if (value > 4095) {                             //ให้ค่าที่อ่านได้สูงกว่า 4095 เท่ากับ 4095
    value = 4095;
  }

  int PercentValue = map(value, 900, 4095, 100, 0);    //หาเปอร์เซ็นต์ ให้ค่าต่ำสุดคือ 900 และสูงสุดคือ 4096

  if (PercentValue > 80) {                               //ถ้าค่าความชื้นดินมากกว่า 80% (ดินเปียกมาก)
    digitalWrite(R_LEDPIN, LOW); digitalWrite(G_LEDPIN, HIGH); digitalWrite(B_LEDPIN, LOW); //เปิดไฟ LED สีเขียว
  } else if (PercentValue <= 80 and PercentValue >= 40) { //ถ้าค่าความชื้นดินอยู่ระหว่าง 40-80% (ดินชุ่ม)
  }

  Done uploading.
  Leaving...
  Hard resetting...
39
ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

```

```

/dev/cu.SLAB_USBtoUART

Soil Moisture = 87% : Soil value = 1328
Soil Moisture = 86% : Soil value = 1349
Soil Moisture = 86% : Soil value = 1365
Soil Moisture = 85% : Soil value = 1408
Soil Moisture = 82% : Soil value = 1490
Soil Moisture = 79% : Soil value = 1571
Soil Moisture = 73% : Soil value = 1771
Soil Moisture = 55% : Soil value = 2358
Soil Moisture = 0% : Soil value = 4095
Soil Moisture = 25% : Soil value = 3299
Soil Moisture = 29% : Soil value = 3187
Soil Moisture = 29% : Soil value = 3170
Soil Moisture = 30% : Soil value = 3155
Soil Moisture = 30% : Soil value = 3138
Soil Moisture = 31% : Soil value = 3119
Soil Moisture = 32% : Soil value = 3103
Soil Moisture = 33% : Soil value = 3071
Soil Moisture = 33% : Soil value = 3067
Soil Moisture = 33% : Soil value = 3057
Soil Moisture = 33% : Soil value = 3043
Soil Moisture = 34% : Soil value = 3039
Soil Moisture = 34% : Soil value = 3029
Soil Moisture = 34% : Soil value = 3019
Soil Moisture = 34% : Soil value = 3015
Soil Moisture = 35% : Soil value = 3002
Soil Moisture = 35% : Soil value = 2998
Soil Moisture = 35% : Soil value = 2997
Soil Moisture = 35% : Soil value = 2992
Soil Moi

Autoscroll Both NL & CR 115200 baud Clear output

```

ตัวอย่างนี้เหมาะกับการนำผลเปอร์เซ็นต์ที่ได้ ไปใช้งานต่อ เช่นนำไปพล็อตกราฟแสดงผล เพราะเป็นค่าที่คนทั่วไปเข้าใจได้ง่าย โดยค่าเปอร์เซ็นต์ที่สูงจะหมายถึงดินชื้นสูง และเปอร์เซ็นต์ที่ต่ำหมายถึงดินแห้ง นั่นเอง

EX11_DHT11

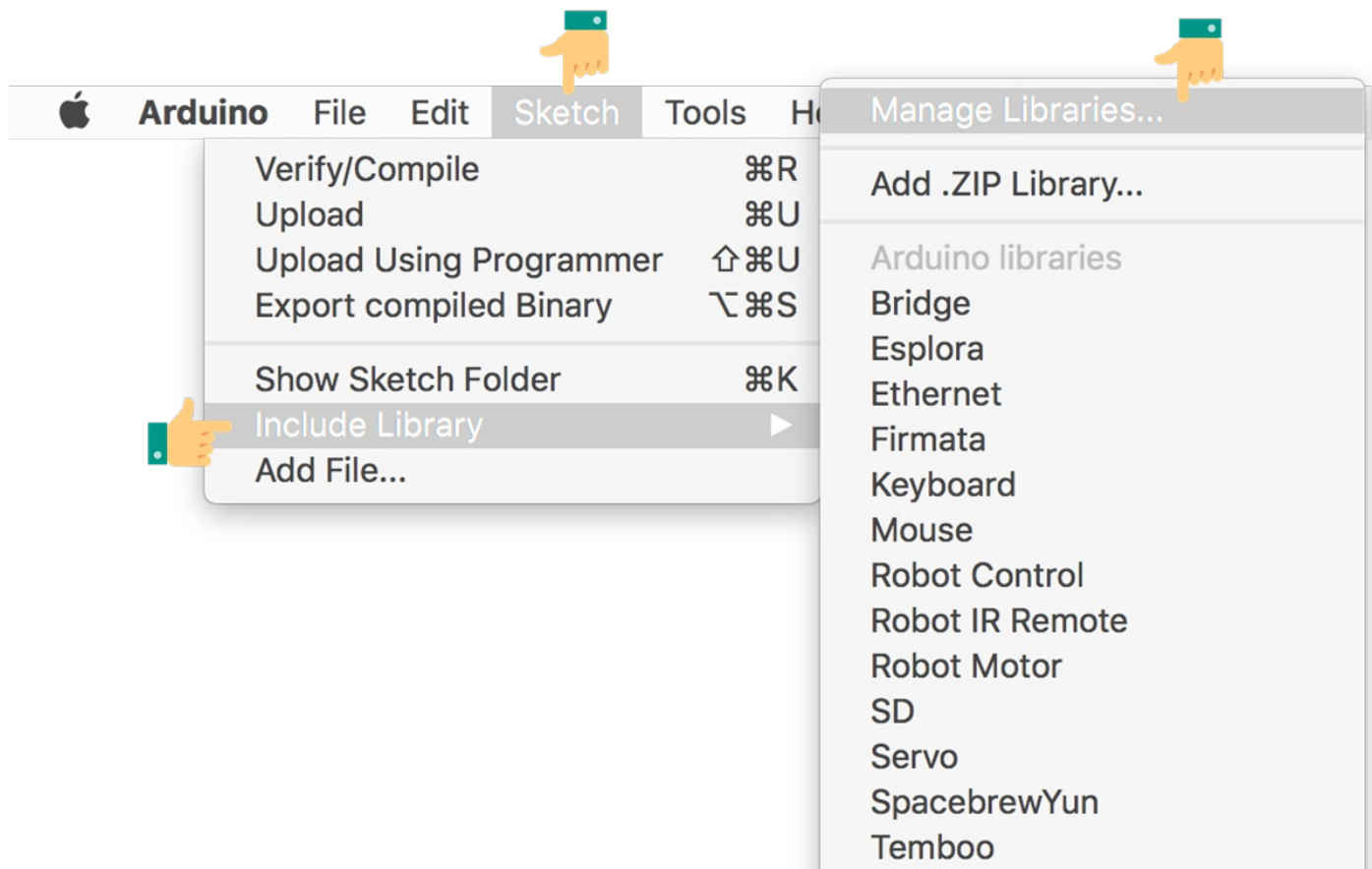
ตัวอย่างทดสอบ การอ่านค่าอุณหภูมิและเปอร์เซ็นต์ความชื้นสัมพัทธ์ของเซ็นเซอร์ DHT11

ผลลัพธ์ที่ได้คือ จะแสดงค่าอุณหภูมิและเปอร์เซ็นต์ความชื้นสัมพัทธ์ที่อ่านได้บนคอนโซล

ขาที่ต่อทดสอบ คือ

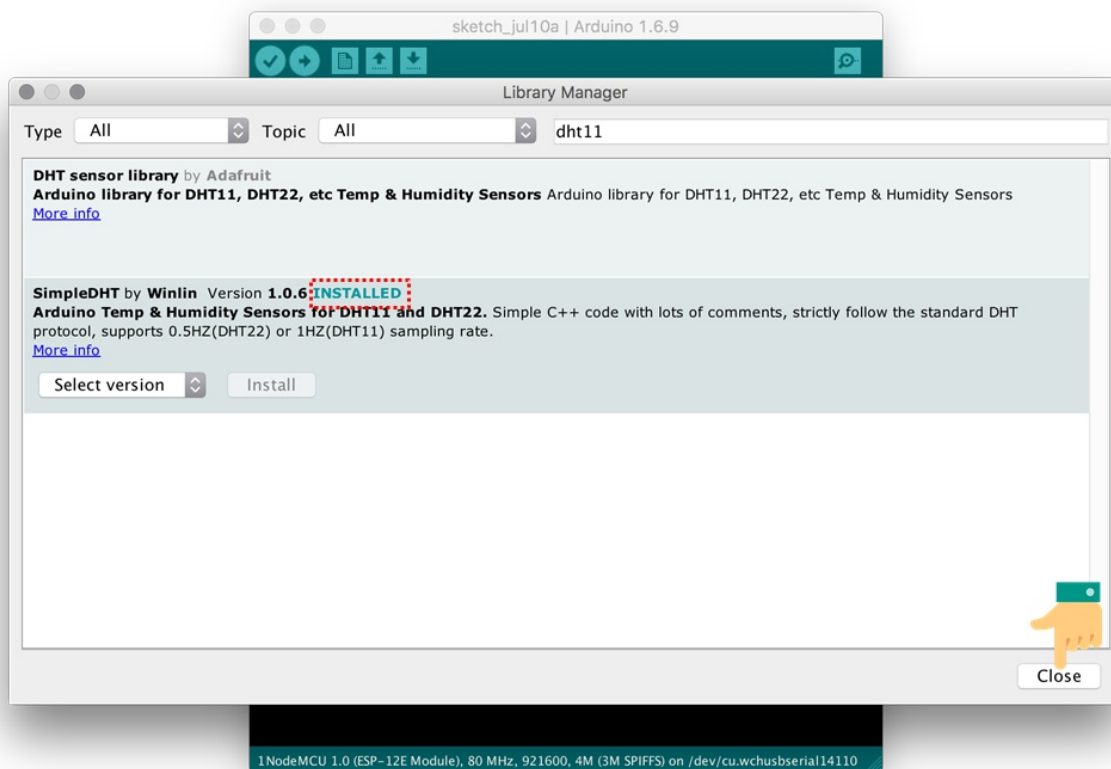
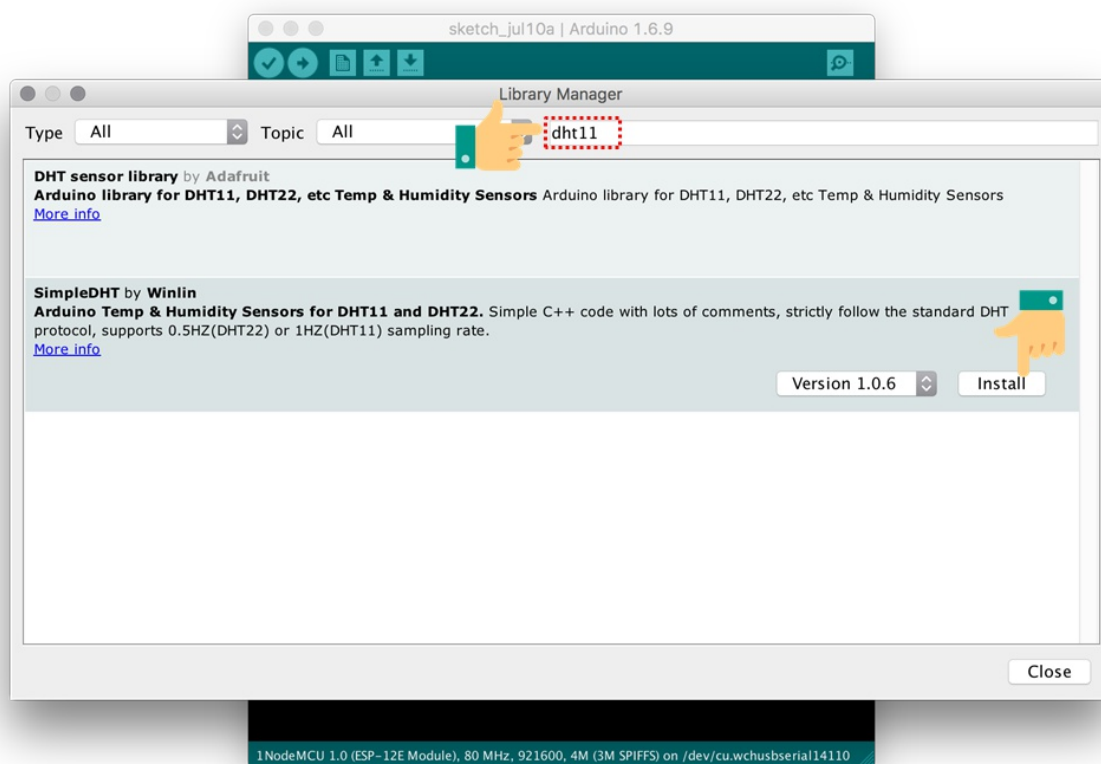
GPIO15 //DHT11 sensor

การติดตั้งไลบรารี DHT11

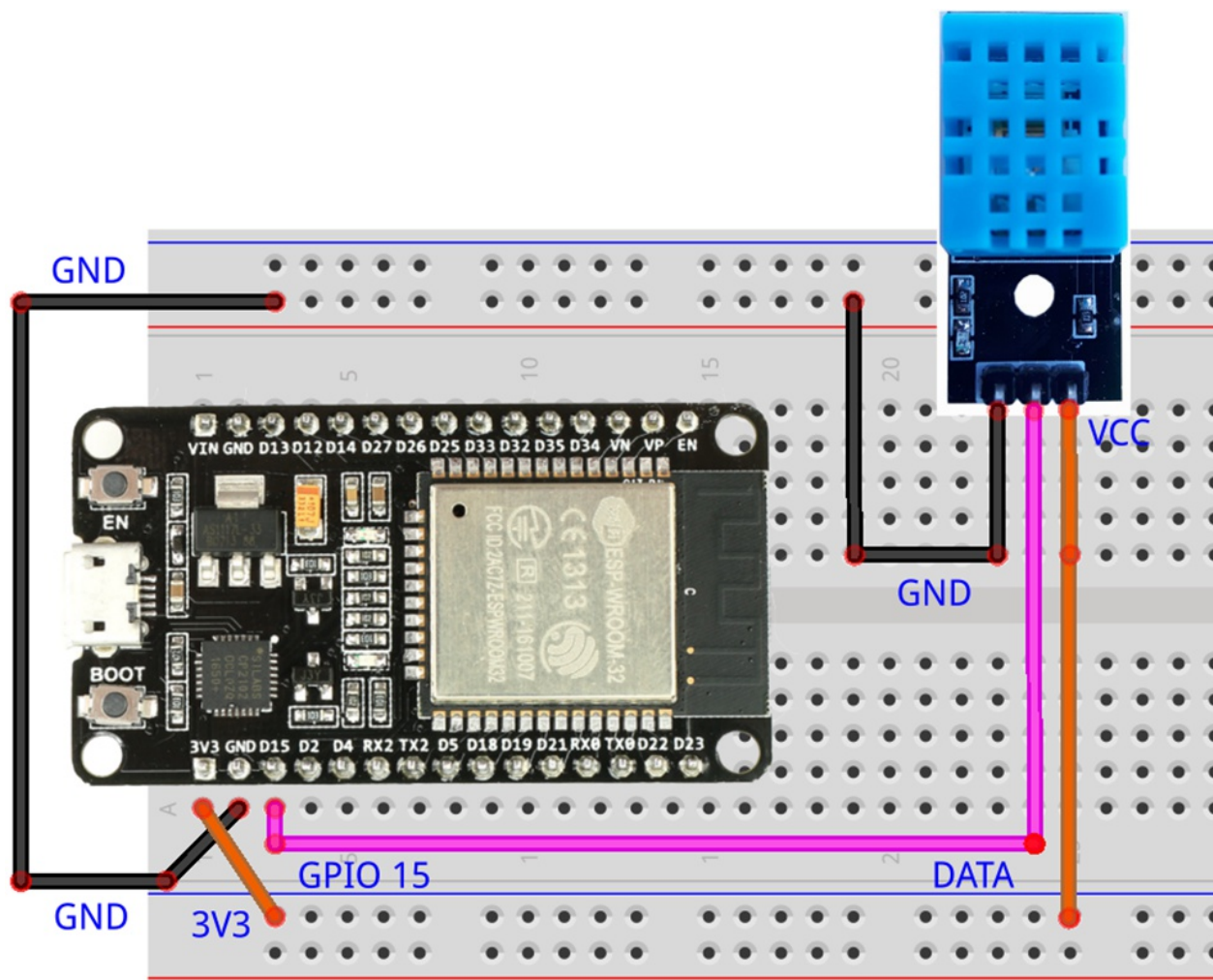


EX11 อ่านค่าอุณหภูมิและความชื้นสัมพัทธ์

การติดตั้งไลบรารี DHT11



EX11 อ่านค่าอุณหภูมิและความชื้นสัมพัทธ์



EX11 อ่านค่าอุณหภูมิและความชื้นสัมพัทธ์

```

EX11_DHT11 | Arduino 1.8.5
Serial Monitor

EX11_DHT11 §
#include <SimpleDHT.h>           //โหลดไลบรารี DHT11

int DHTPIN = 15;                 //GPIO15 (ADC2_CH3)    //DHT11 sensor

SimpleDHT11 dht11;               //ระบุพินเซ็นเซอร์ DHT11
byte temperature = 0;            //กำหนดตัวแปรเก็บค่าอุณหภูมิ
byte humidity = 0;              //กำหนดตัวแปรเก็บค่าความชื้นสัมพัทธ์

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

void loop() {
  dht11.read(DHTPIN, &temperature, &humidity, NULL); //อ่านค่าจากเซ็นเซอร์
  delay(1500);                  //รอ 1.5 วินาที

  Serial.print("DHT11 read: ");
  Serial.print((int)temperature); Serial.print(" *C, ");
  Serial.print((int)humidity); Serial.println(" %H");
}

Done uploading.
Leaving...
Hard resetting...

12 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

```

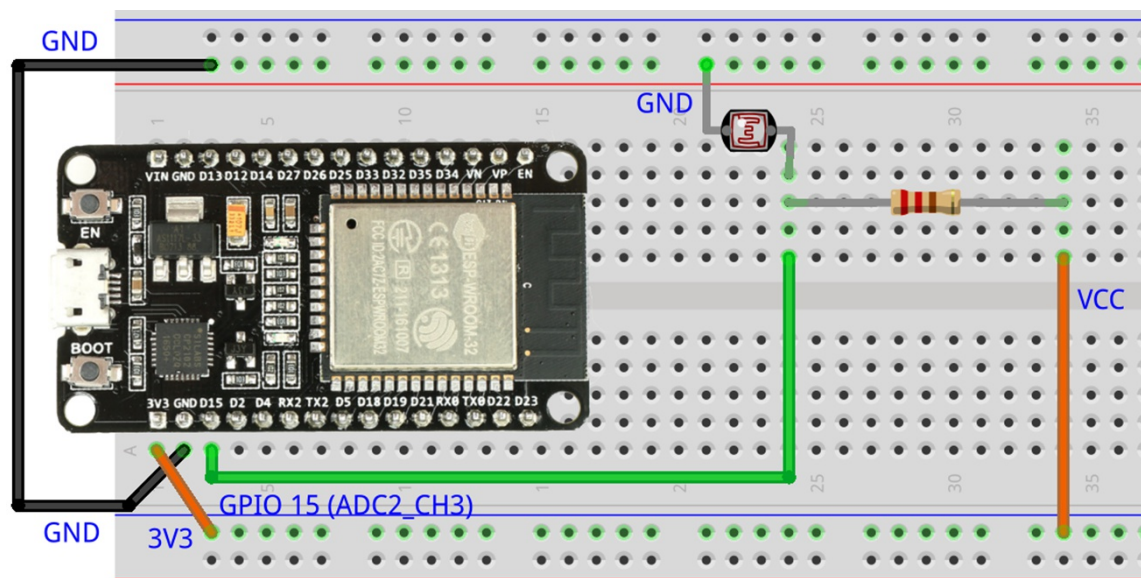

EX12_LDR_SENSOR

ตัวอย่างทดสอบ การอ่านค่าความเข้มแสงของเซ็นเซอร์ LDR

ผลลัพธ์ที่ได้คือ จะแสดงค่าความเข้มแสงที่อ่านได้บนคอนโซล ค่าแสงสว่างต่ำเข้าใกล้ 0 จะหมายถึงแสงสว่างมาก

ขาที่ต่อทดสอบ คือ

GPIO15 (ADC2_CH3) //LDR sensor



EX12 อ่านค่าความเข้มแสงของ LDR sensor



```
EX12_LDR_SENSOR | Arduino 1.8.5

EX12_LDR_SENSOR
int LDRSENSOR = 15; //GPIO15 (ADC2_CH3) //LDR sensor

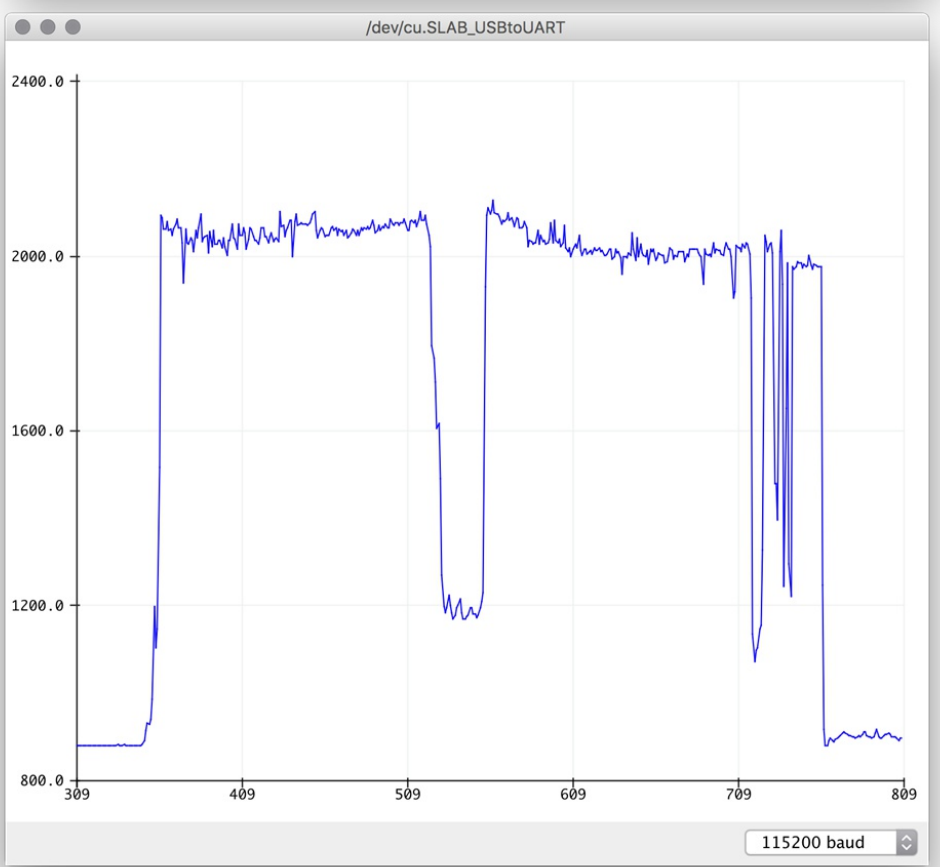
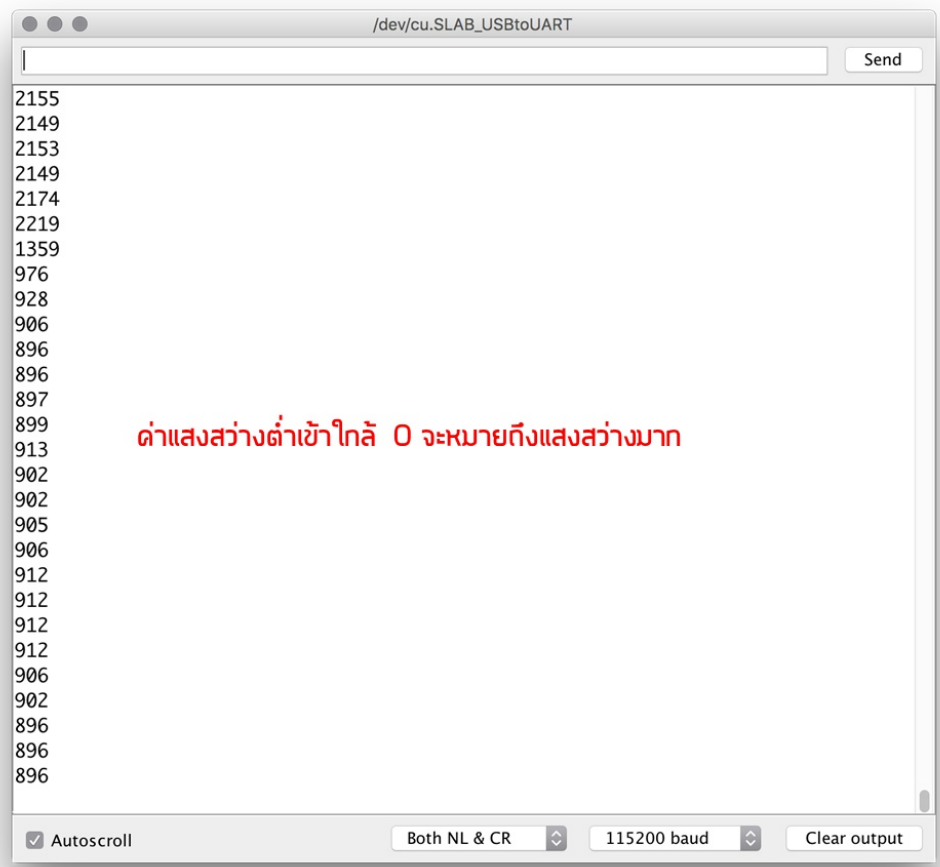
void setup() {
  Serial.begin(115200);
}

void loop() {
  int value = analogRead(LDRSENSOR); //อ่านค่าความเข้มแสง
  Serial.println(value);
  delay(100); //รอ 1 วินาที
}

Done uploading.
Leaving...
Hard resetting...

11 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART
```

EX12 อ่านค่าความเข้มแสงของ LDR sensor



EX13_LDR_LED_SENSOR

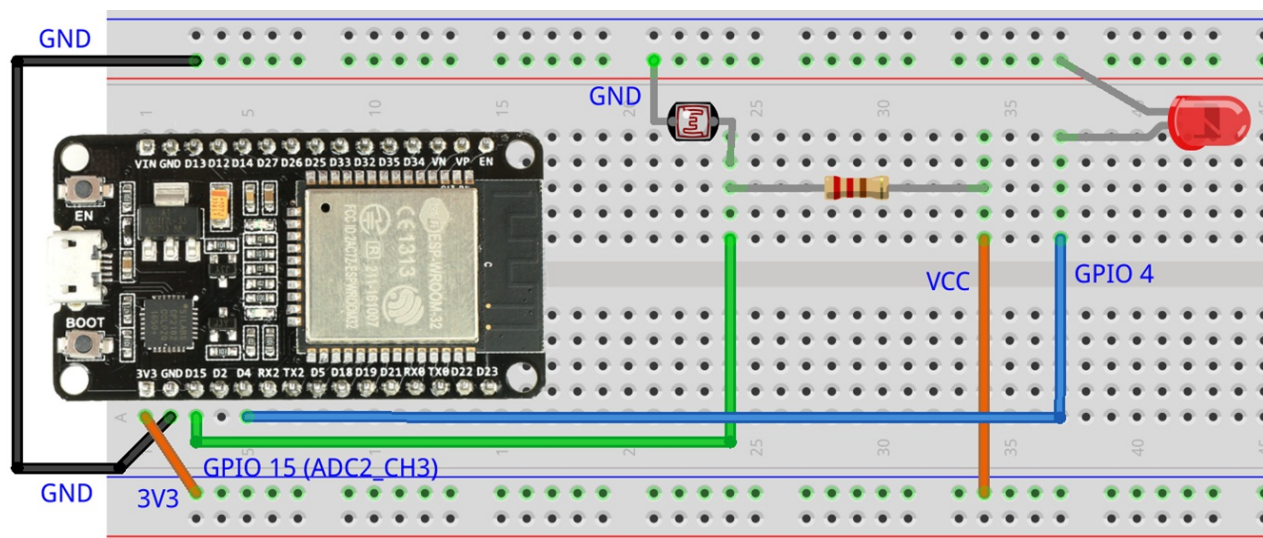
ตัวอย่างทดสอบ การอ่านค่าความเข้มแสงของเซ็นเซอร์ LDR

ผลลัพธ์ที่ได้คือ จะแสดงค่าความเข้มแสงที่อ่านได้บนคอนโซล ค่าแสงสว่างต่ำเข้าใกล้ 0 จะหมายถึงแสงสว่างมาก ถ้าค่าแสงสว่างมากกว่า 2000 หลอดไฟ LED จะติดสว่าง

ขาที่ต่อทดสอบ คือ

GPIO15 (ADC2_CH3)//LDR sensor

GPIO4 //LED



EX13 ควบคุมหลอด LED ตามความเข้มแสง

```

EX13_LDR_LED_SENSOR
int LEDPIN = 4; //GPIO4 LED
int LDRSENSOR = 15; //GPIO15 (ADC2_CH3) //LDR SENSOR

void setup() {
  Serial.begin(115200);
  pinMode(LEDPIN, OUTPUT); //กำหนดขา LEDPIN เป็นขาออกที่เชื่อมต่อ LED
}

void loop() {
  int value = analogRead(LDRSENSOR); //อ่านค่าความเข้มแสง
  if (value < 800) { //ให้ค่าที่อ่านได้ต่ำกว่า 800 เท่ากับ 800
    value = 800;
  } else if (value > 2000) { //ให้ค่าที่อ่านได้สูงกว่า 2000 เท่ากับ 2000
    value = 2000;
  }

  int PercentValue = map(value, 800, 2000, 100, 0); //หาเปอร์เซ็นต์ ให้ค่าต่ำสุดคือ 800 และสูงสุดคือ 2000

  if (PercentValue < 30) { //ถ้าค่าความเข้มแสงน้อยกว่า 30% (มืดมาก)
    digitalWrite(LEDPIN, HIGH); //เปิดไฟ LED
    Serial.println("LED TURN ON");
  } else {
    digitalWrite(LEDPIN, LOW); //ปิดไฟ LED
    Serial.println("LED TURN OFF");
  }

  Serial.print("LDR percent = "); //แสดงค่าในคอนโซล
  Serial.print(PercentValue); //แสดงค่าในคอนโซล
}

```

Done Saving.
Leaving...
Hard resetting...

26 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

```

LDR percent = 14% : LDR value = 1837
LED TURN ON
LDR percent = 17% : LDR value = 1798
LED TURN ON
LDR percent = 16% : LDR value = 1811
LED TURN ON
LDR percent = 16% : LDR value = 1808
LED TURN ON
LDR percent = 17% : LDR value = 1802
LED TURN ON
LDR percent = 15% : LDR value = 1820
LED TURN ON
LDR percent = 17% : LDR value = 1806
LED TURN OFF
LDR percent = 66% : LDR value = 1211
LED TURN OFF
LDR percent = 94% : LDR value = 877
LED TURN OFF
LDR percent = 94% : LDR value = 877
LED TURN OFF
LDR percent = 94% : LDR value = 876
LED TURN OFF
LDR percent = 93% : LDR value = 885
LED TURN OFF
LDR percent = 92% : LDR value = 896
LED TURN OFF
LDR percent = 92% : LDR value = 901

```

☒ Autoscroll Both NL & CR 115200 baud Clear output

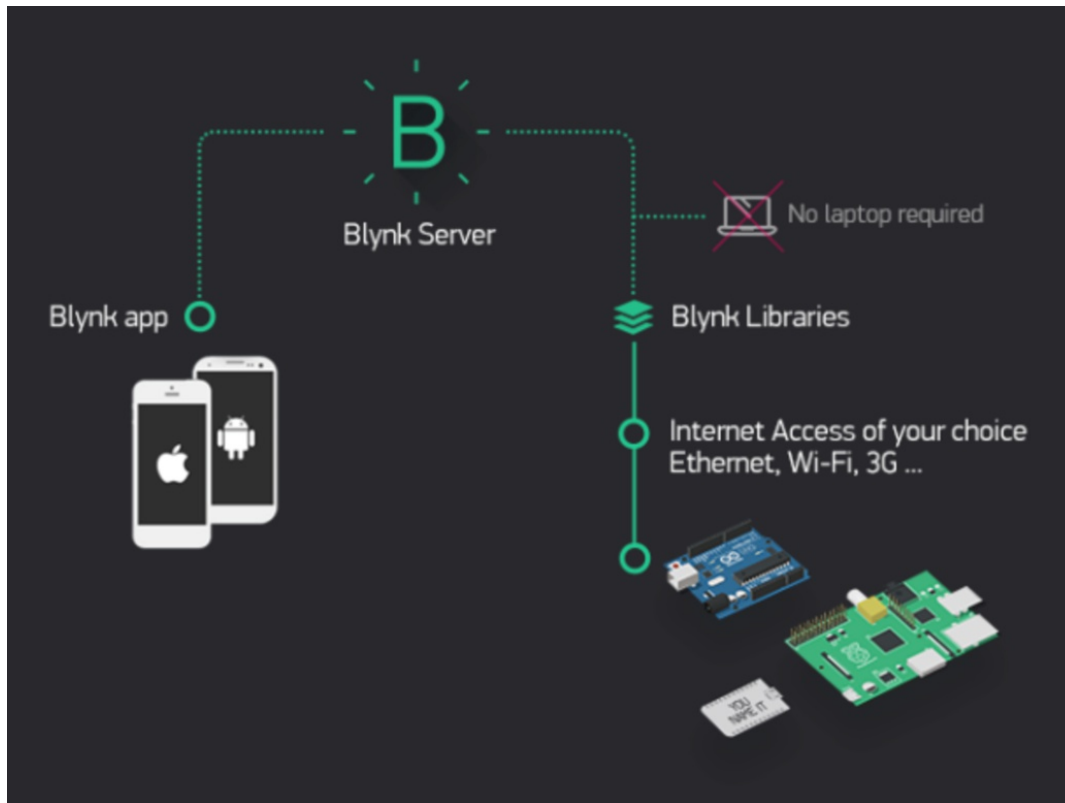
Blynk

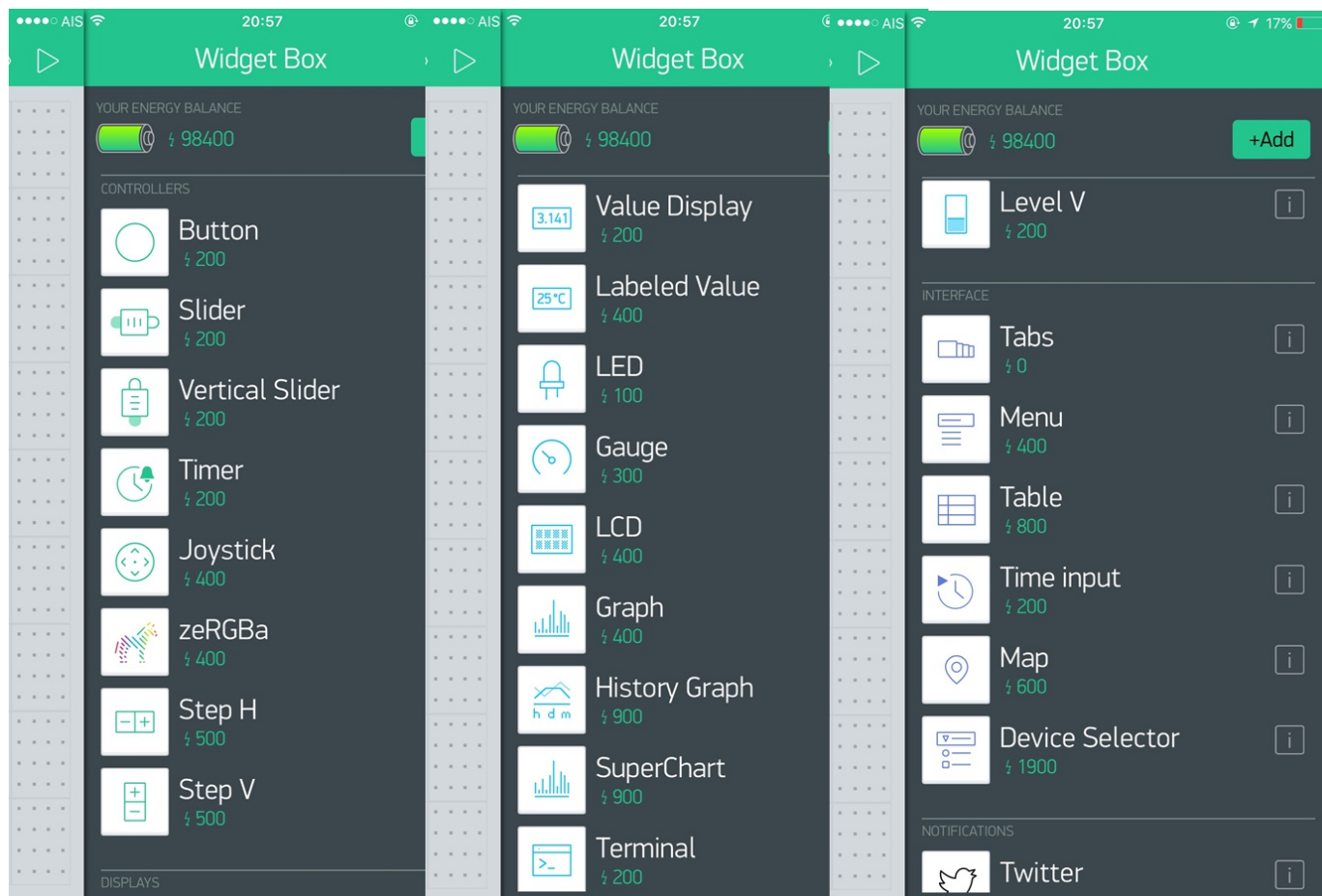
รู้จัก blynk apP



130

BLYNK



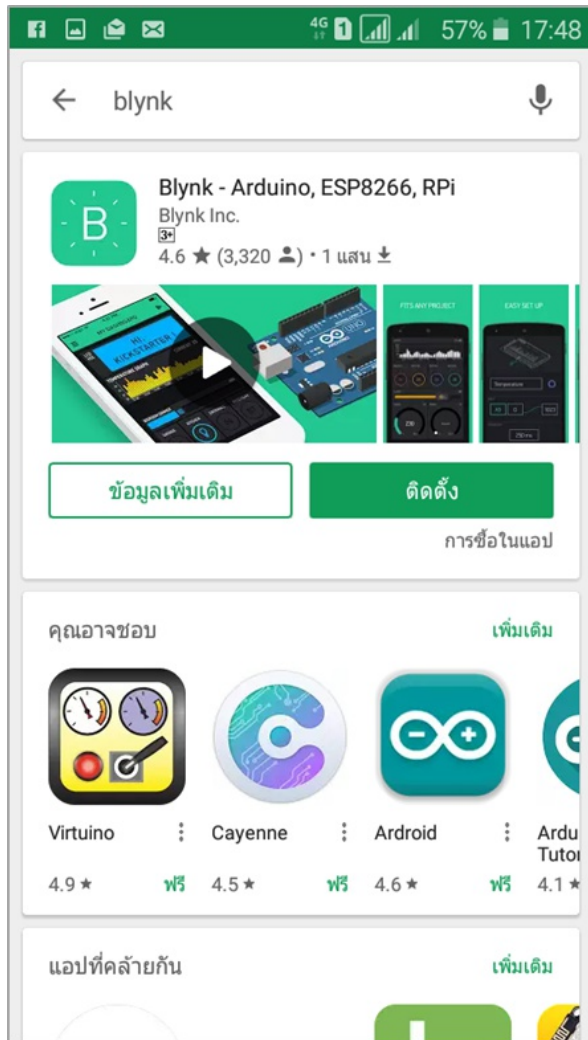


Blynk เป็นโครงการที่ระดมทุนผ่าน Kickstarter ได้สำเร็จไปเมื่อช่วงต้นปี 2015 ที่ผ่านมา เรียกได้ว่าเป็นน้องใหม่ไฟแรงเลยก็ว่าได้ครับ ความน่าสนใจคงอยู่ตรง Blynk เป็นแพลตฟอร์มที่จะช่วยให้สามารถสร้างระบบ IoT สำหรับงานต่างๆ ไปได้ง่ายกว่าเดิม จากเครื่องมือที่ Blynk ได้เตรียมไว้ให้ ในรูปที่ 1 เป็นส่วนการทำงานซึ่งประกอบกันเป็นระบบ IoT ของ Blynk จะเห็นได้ว่าโครงสร้างหลักประกอบด้วย 3 ส่วน ได้แก่ สมาร์ทโฟน, เซิร์ฟเวอร์ของ Blynk และ ส่วนฮาร์ดแวร์ของระบบสมองกลฝังตัวระบบสมองกลฝังตัว เป็นส่วนที่เราอาจจะสร้างเองก็ได้ หรือถ้าต้องการความสะดวกก็อาจใช้โอเพนซอร์สฮาร์ดแวร์ (Open-Source Hardware) ได้เช่นกัน โดยในเวลานี้ Blynk สามารถใช้งานร่วมกับฮาร์ดแวร์ตระกูลหลักๆ ได้แก่ Arduino, Raspberry Pi, Beaglebone, ESP8266, Electric Imp, Spark Core เป็นต้น (ยังคงมีการอัปเดต เพื่อรองรับฮาร์ดแวร์ใหม่ๆ อยู่ตลอดเวลาแต่คงต้องรอดูใจรอกันสักนิด) การพัฒนาซอฟต์แวร์บนฮาร์ดแวร์ที่ง่ายที่เดียวครับ จากไลบรารี (library) ที่ Blynk เตรียมไว้ให้แบบเสร็จสำเร็จรูปนั่นเอง

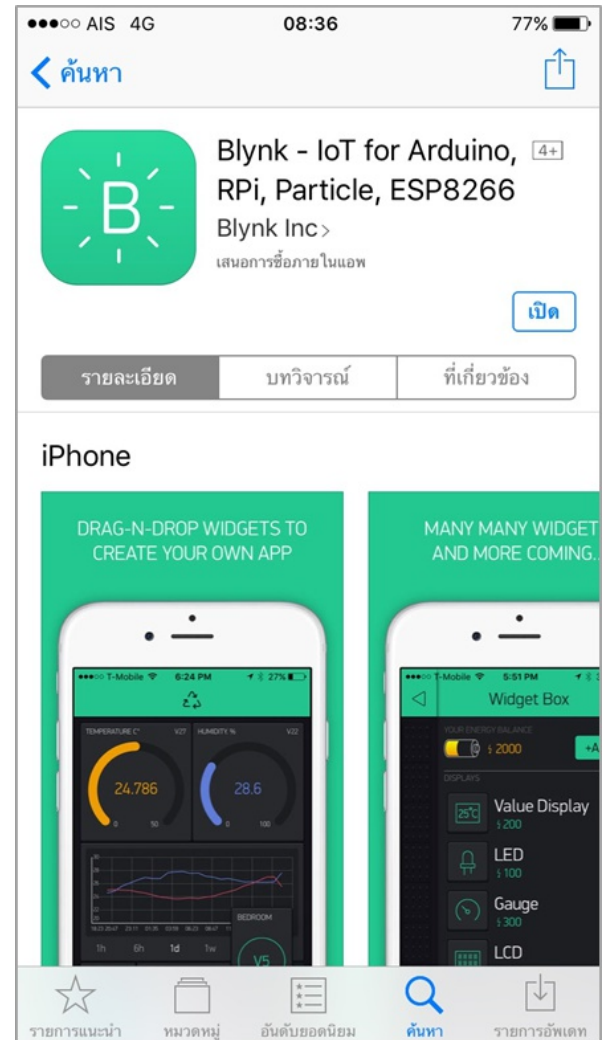
การติดตั้ง Blynk

รู้จัก blynk app และการใช้งาน

ดาวน์โหลด Blynk บน smart phone

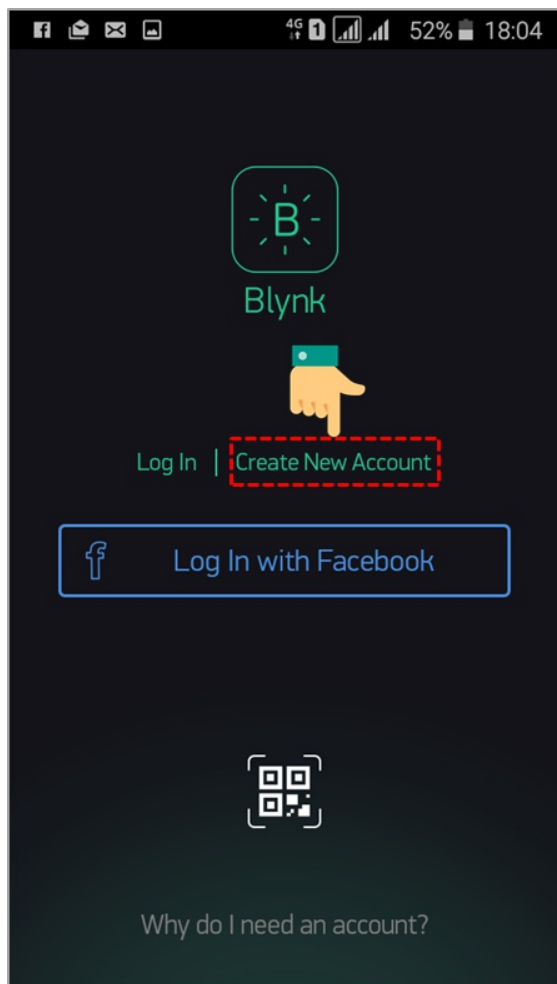


Android OS

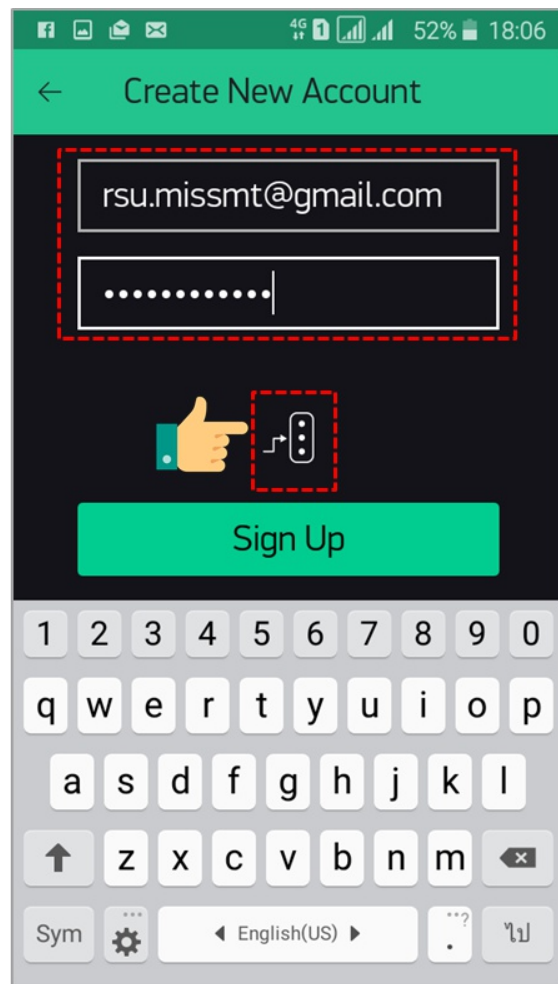


iOS

ลงทะเบียนใช้งานบน blynk server

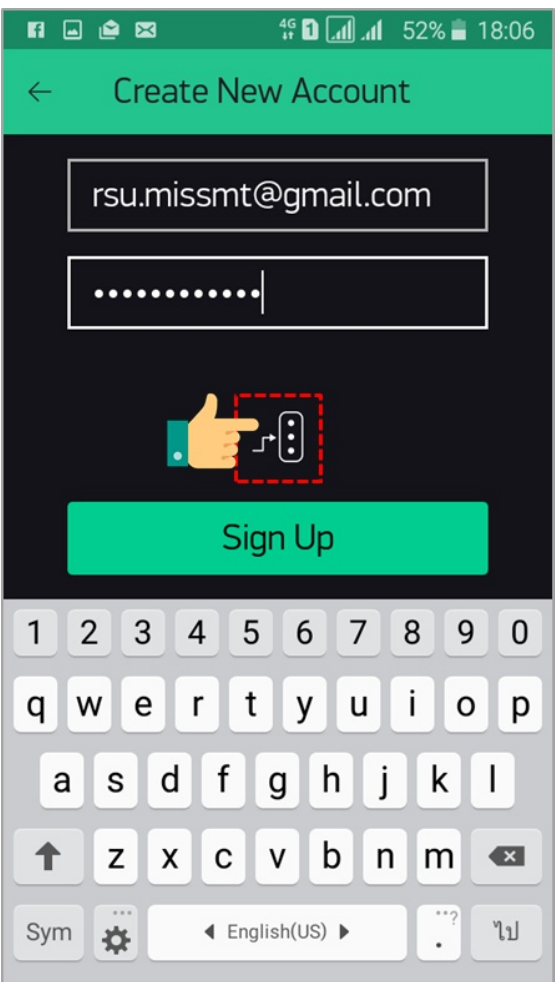


สร้างผู้ใช้งานใหม่

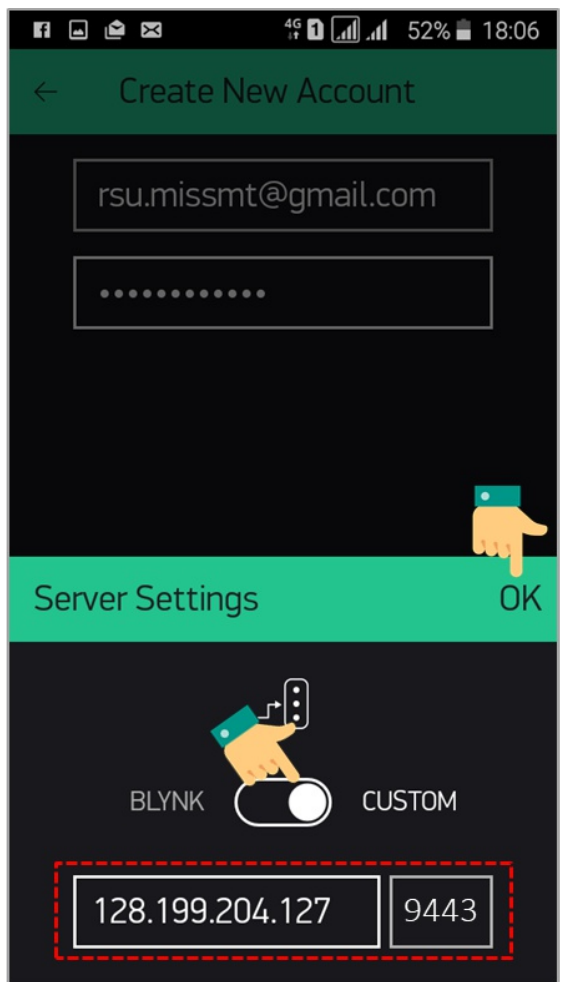


ระบุที่อยู่อีเมลและรหัสผ่าน

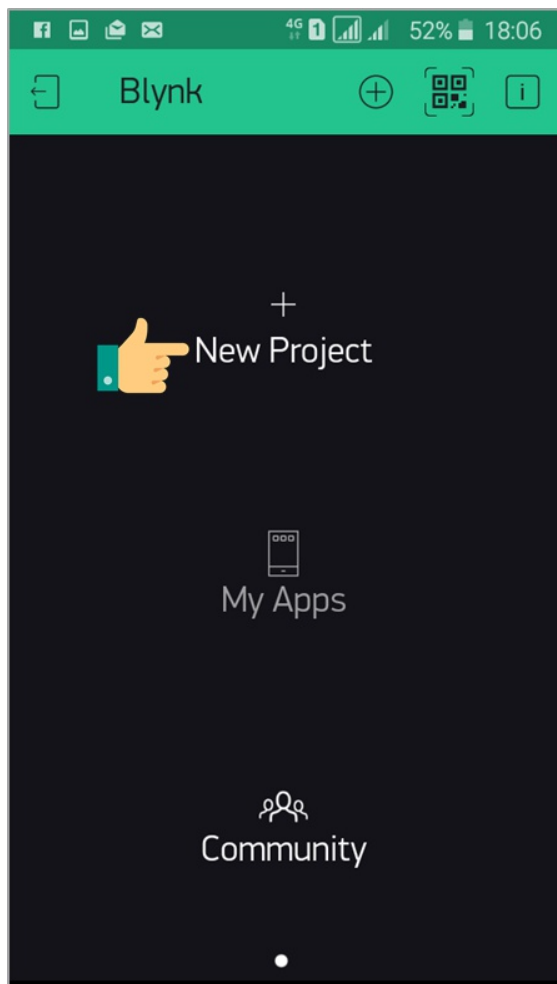
เลือก MISSMT IoT Cloud SERVER (MIC)



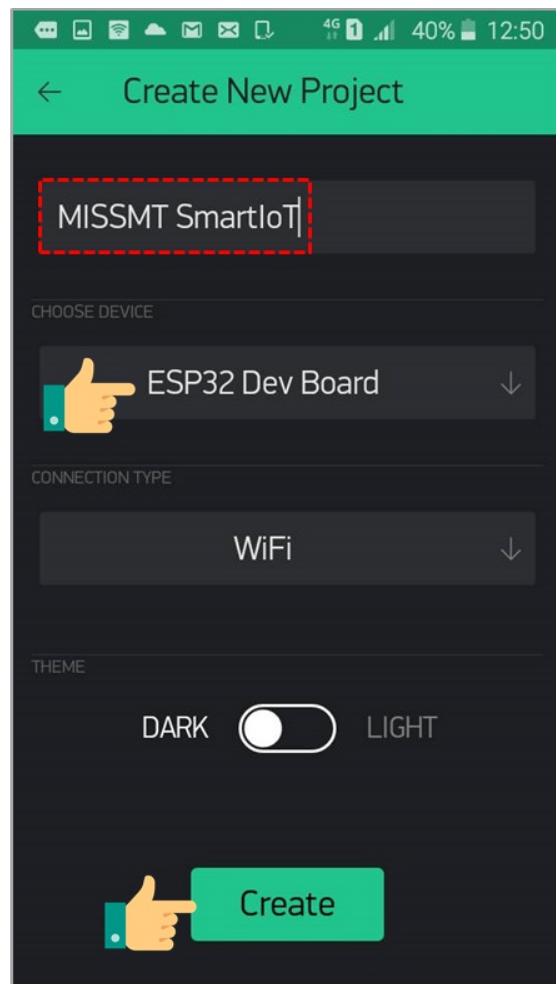
เลือกตั้งค่า Server



ระบุ IP Server และพอร์ต

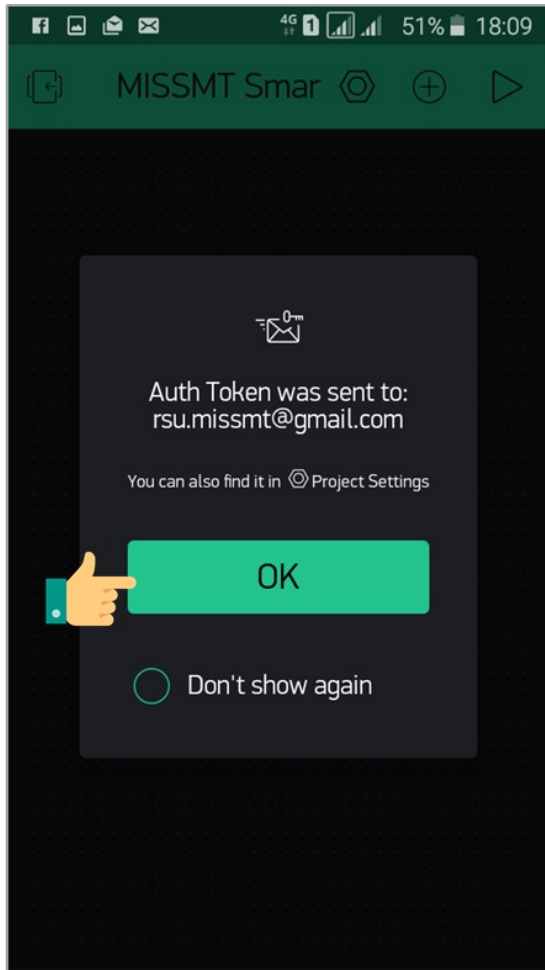


สร้างโปรเจคใหม่



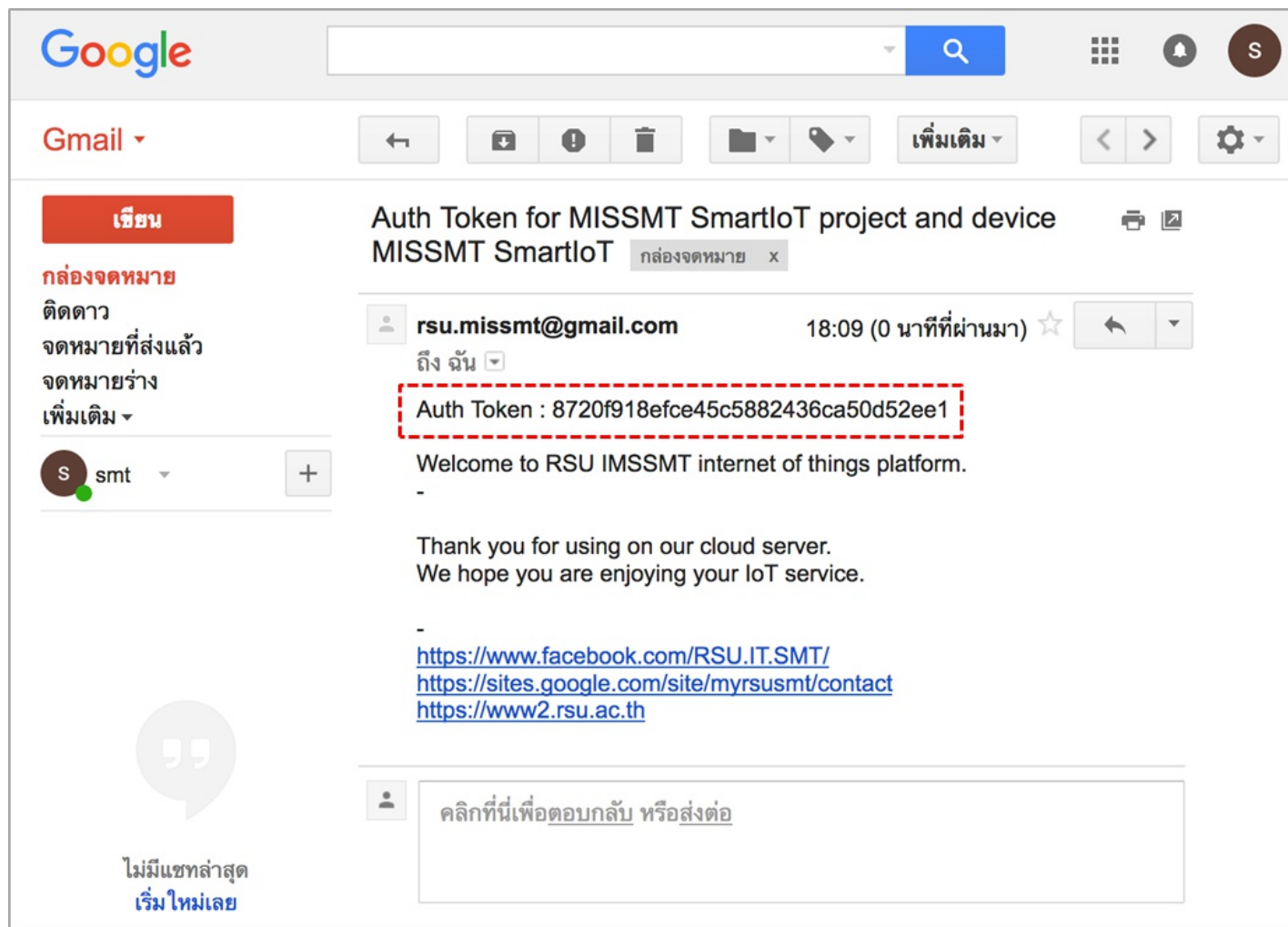
สร้างโปรเจค

ลงทะเบียนใช้งานบน blynk server



ระบบส่งพาส์รหัส TOKEN ไปให้





เลือกตั้งค่า Server

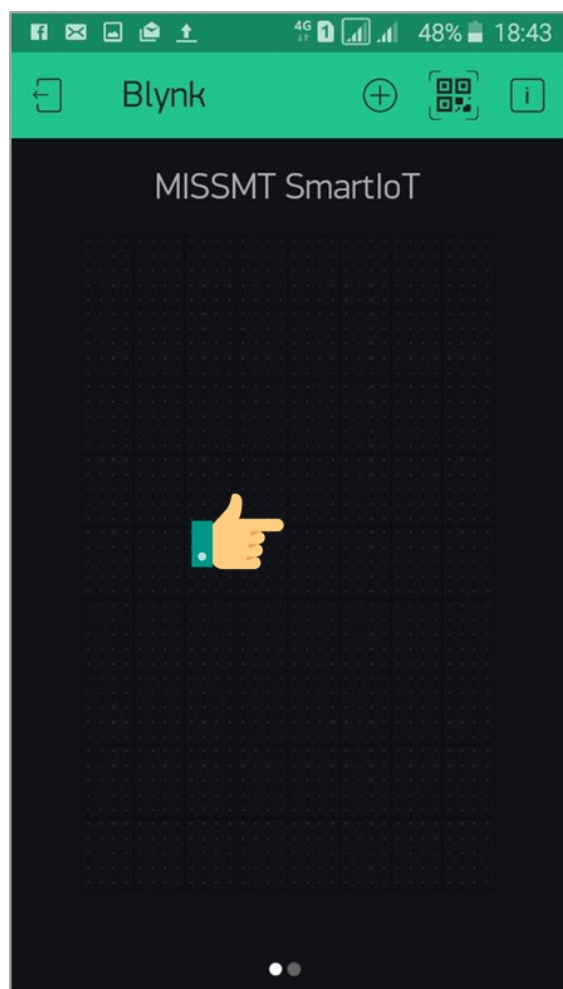
EX14_LED_BLYNK

ตัวอย่างทดสอบ การเชื่อมต่อ ESP32 กับ CLOUD SERVER ในที่นี่จะใช้ BLYNK SERVER ซึ่งเป็น OPEN SOURCE

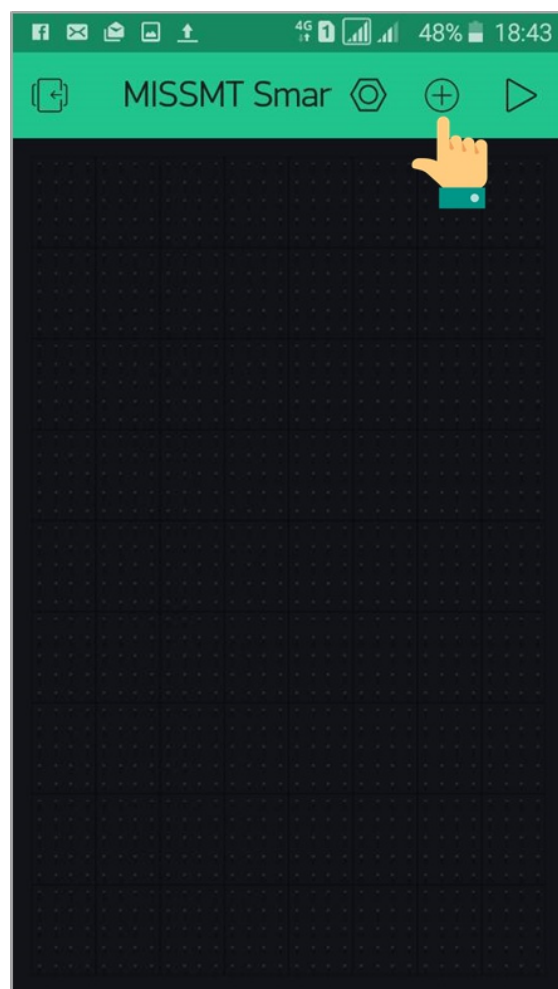
ผลลัพธ์ที่ได้คือ จะสามารถควบคุมปิด เปิด หลอด LED ผ่านทาง แอปพลิเคชันบนสมาร์ตโฟนได้

ขาที่ต่อทดสอบ คือ

GPIO4 //LED

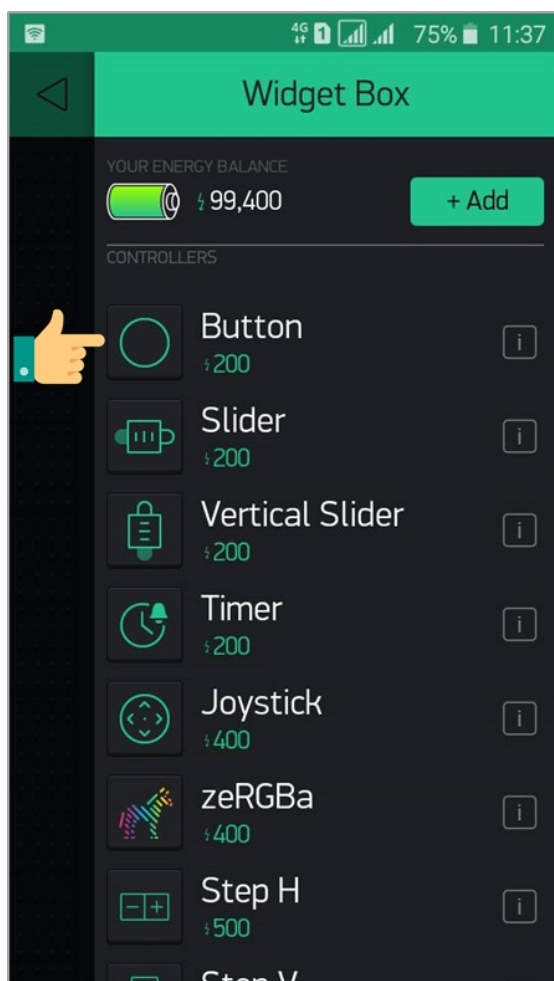


คลิกพื้นที่ว่าง

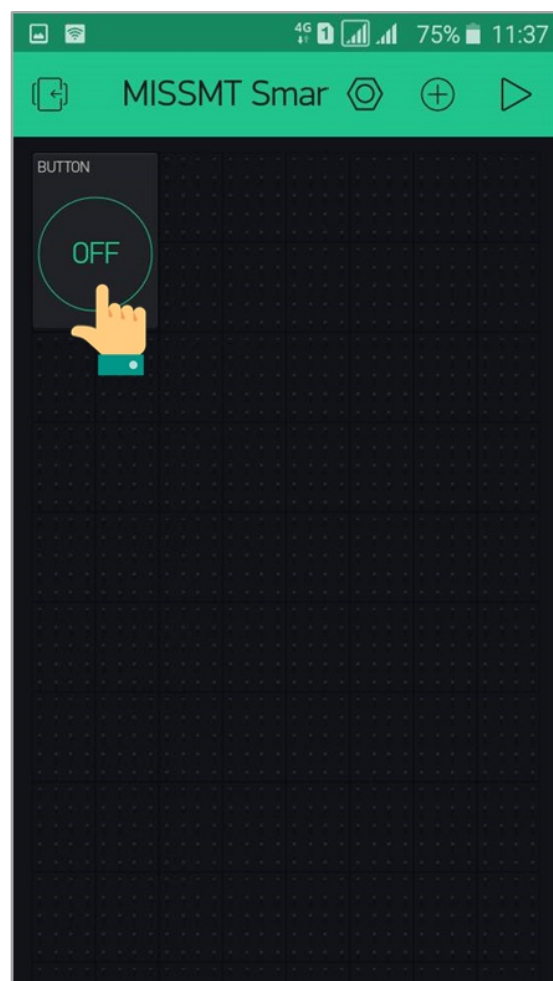


คลิกเครื่องหมายบวก

EX14 สร้างโปรเจกควบคุมไฟ LED ออนไลน์

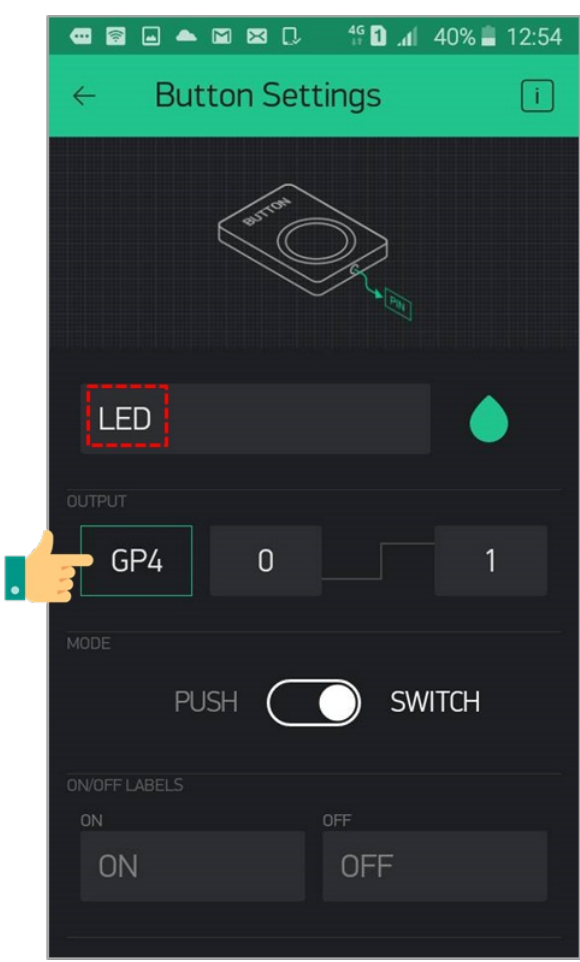


เลือก Button

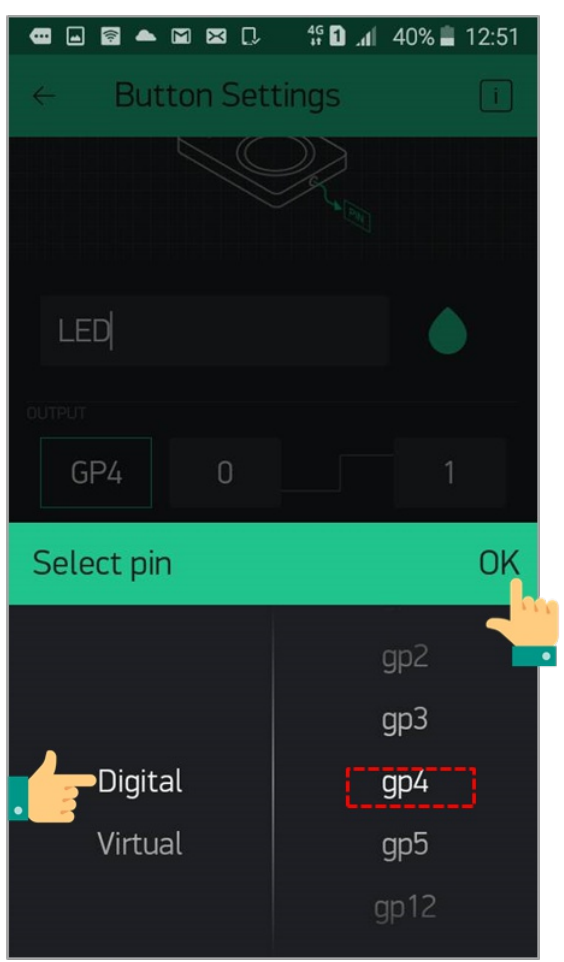


คลิกที่ปุ่ม OFF

EX14 สร้างโปรเจกควบคุมไฟ LED ออนไลน์

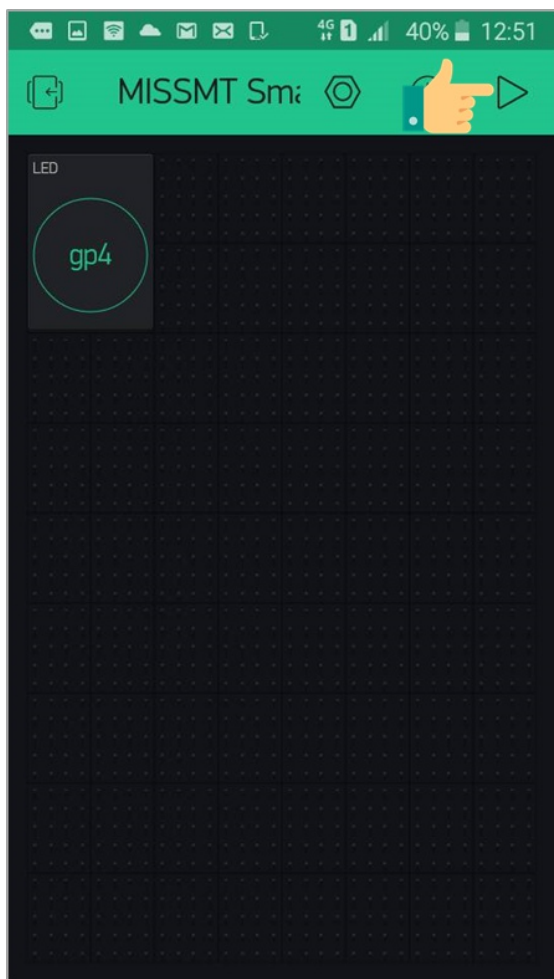


ตั้งค่า Output เป็น GP4

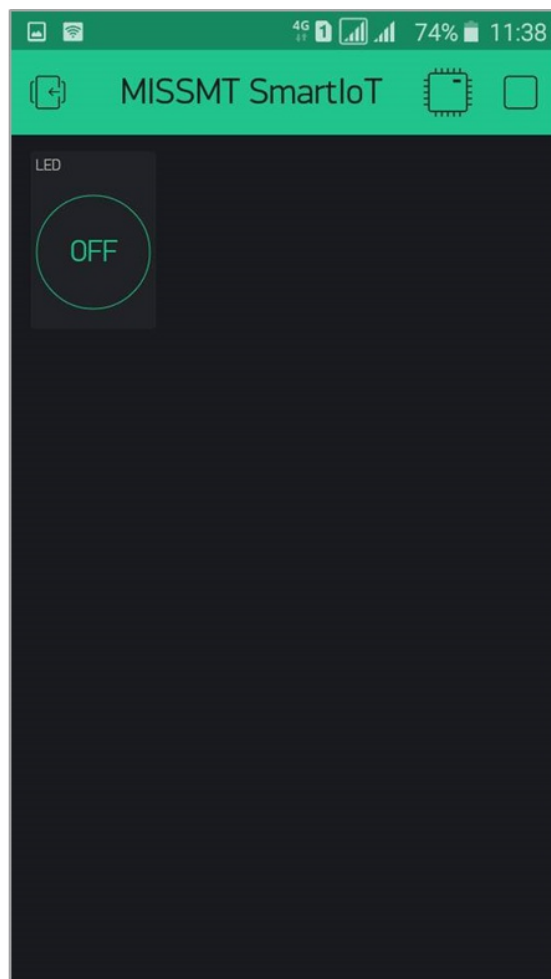


ตั้งค่า Output เป็น GP4

EX14 สร้างโปรเจกควบคุมไฟ LED ออนไลน์

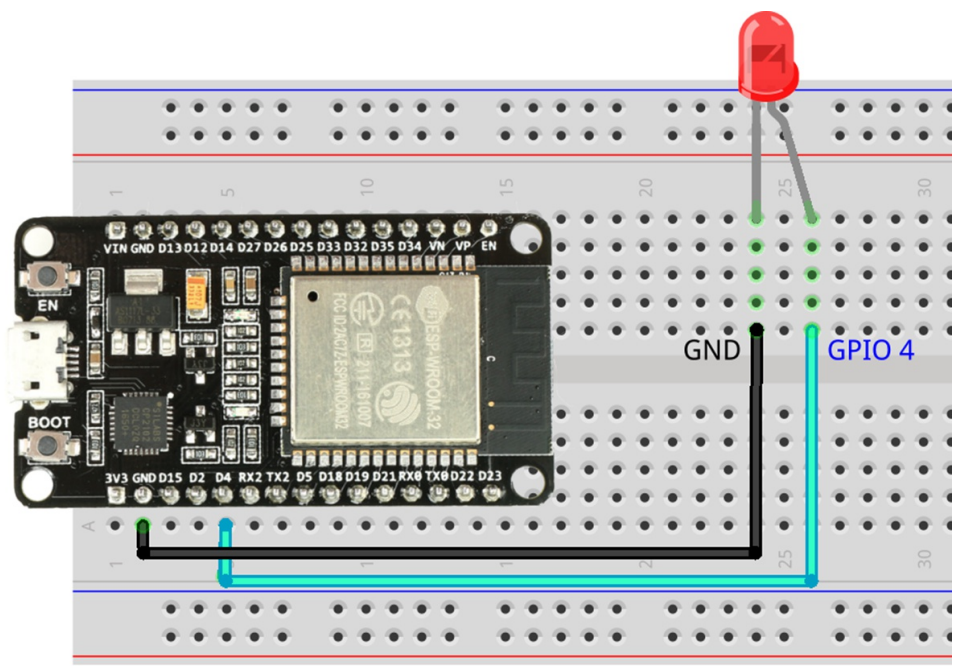


เริ่มใช้งาน



ควบคุมการ ปิด เปิด ไฟ

EX14 สร้างโปรเจกควบคุมไฟ LED ออนไลน์



```

ESP32_WiFi | Arduino 1.8.5

ESP32_WiFi §
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = "a82da6ce85244d1ebcef95ade5ed24a6";
char ssid[] = "plutter";
char pass[] = "visit012";

void setup()
{
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass, IPAddress(128,199,204,127), 8442);
}

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

Done uploading.
Leaving...
Hard resetting...

7 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART
  
```

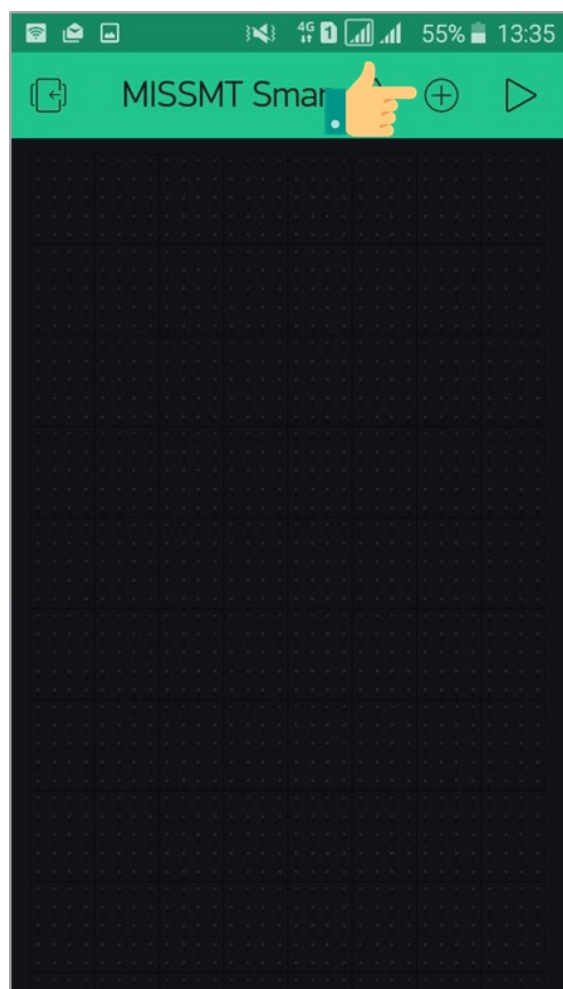

EX15_DHT11_BLYNK

ตัวอย่างทดสอบ การเชื่อมต่อ ESP32 กับ CLOUD SERVER ในที่นี่จะใช้ BLYNK SERVER
ซึ่งเป็น OPEN SOURCE

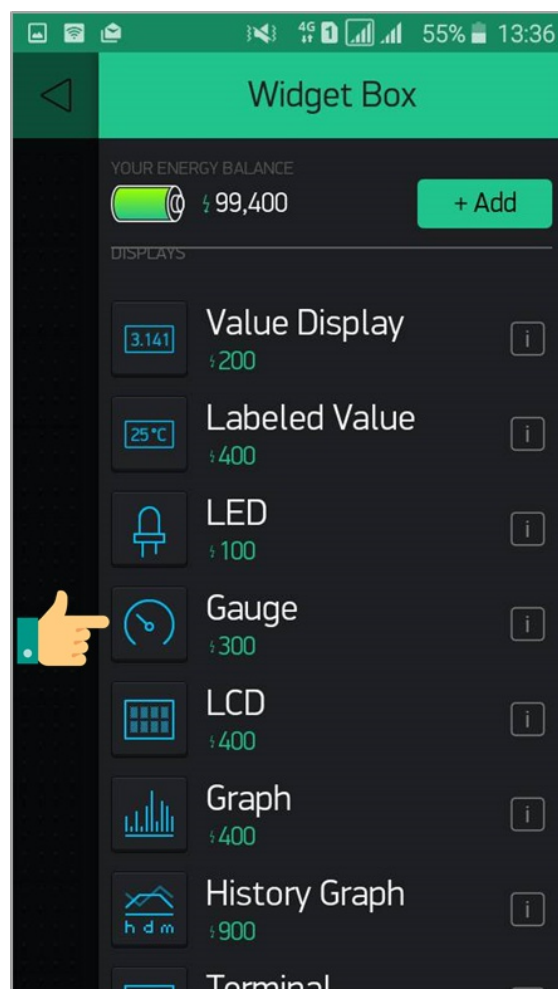
ผลลัพธ์ที่ได้คือ จะสามารถดูค่าอุณหภูมิและความชื้นและแสดงกราฟข้อมูล ผ่านทางแอปพลิเคชันบนสมาร์ตโฟนได้

ขาที่ต่อทดสอบ คือ

GPIO15 //DHT11 Sensor

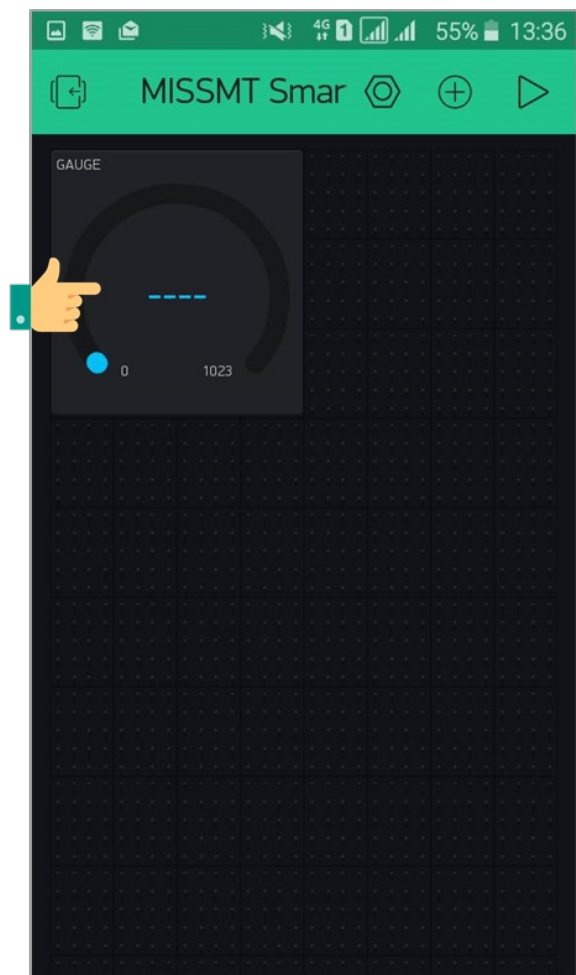


เพิ่มเครื่องมือ

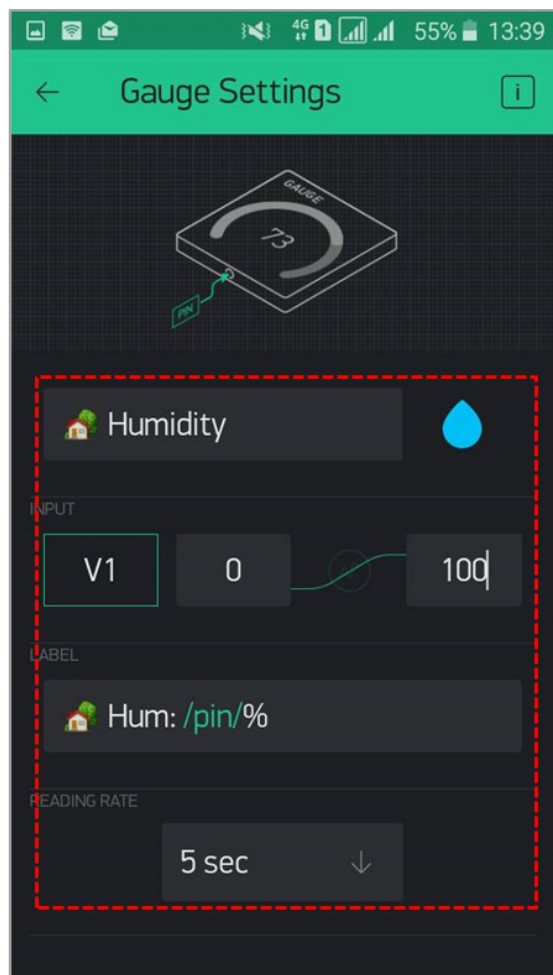


เลือก Gauge

EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

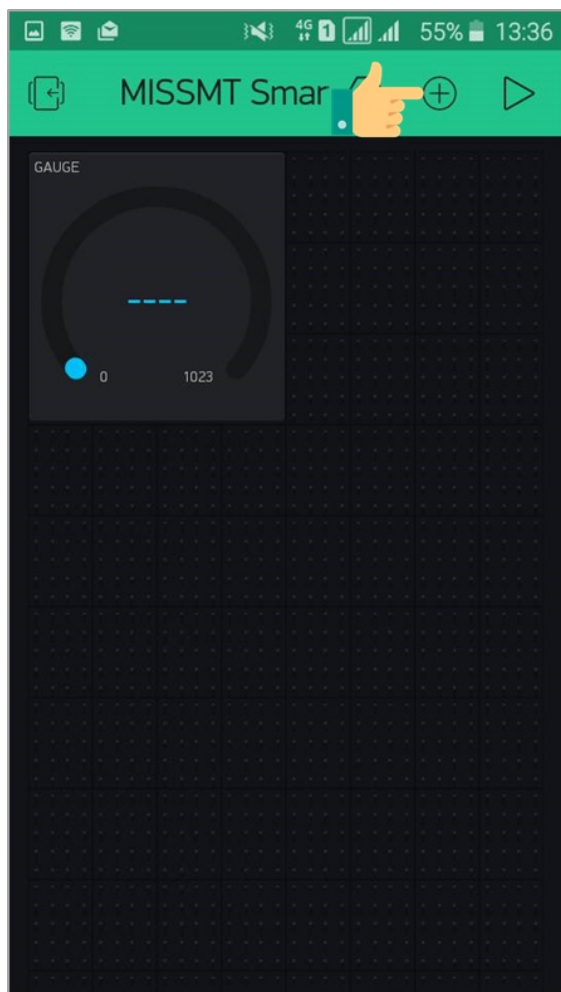


คลิกที่ปุ่ม Gauge

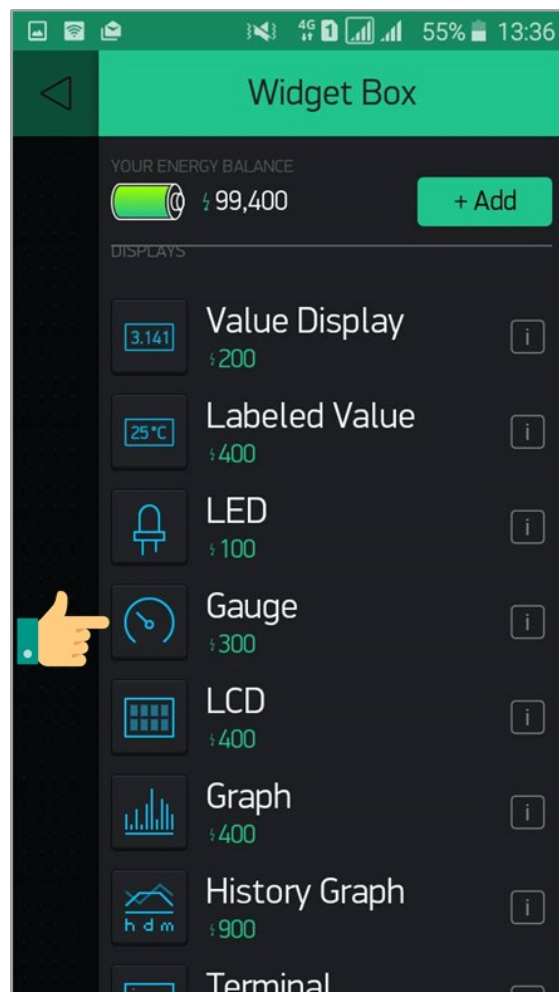


ตั้งค่าตามในภาพ

EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

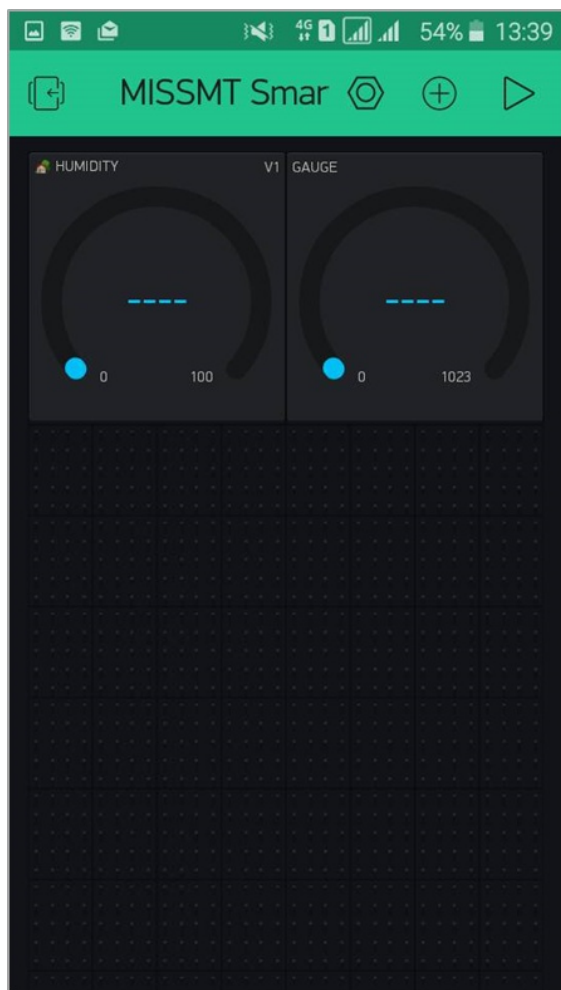


เพิ่มเครื่องมือ

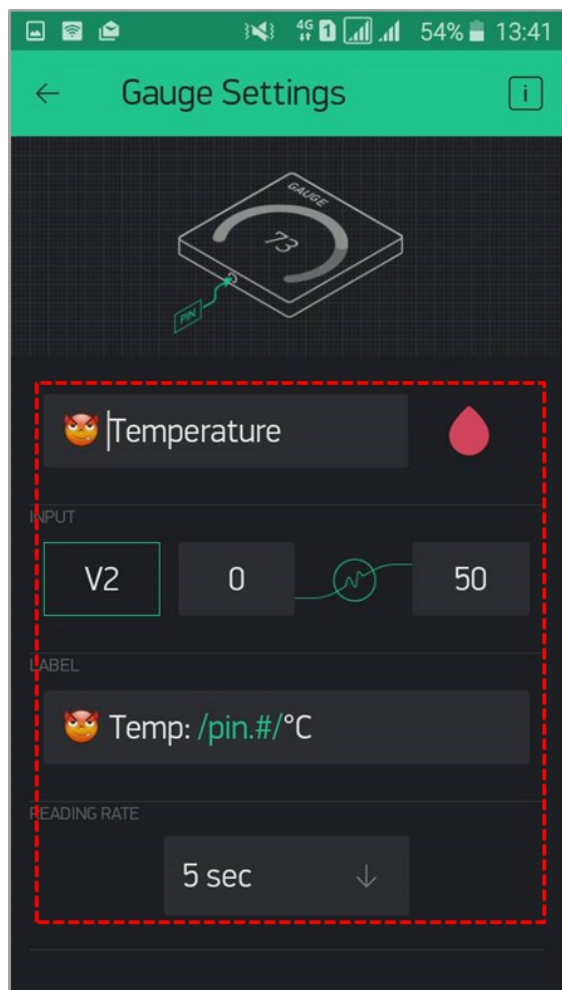


เลือก Gauge

EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

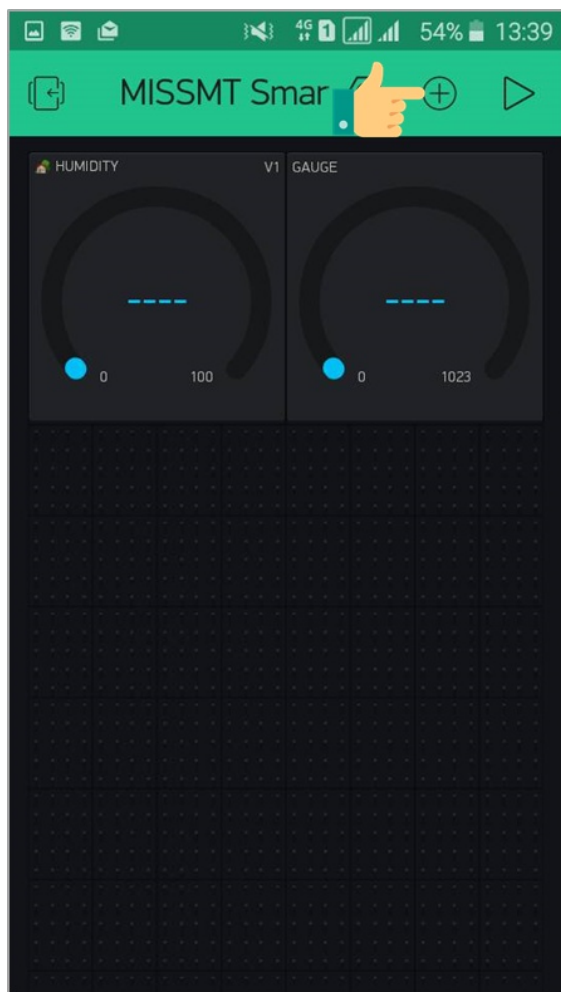


คลิกที่ปุ่ม Gauge

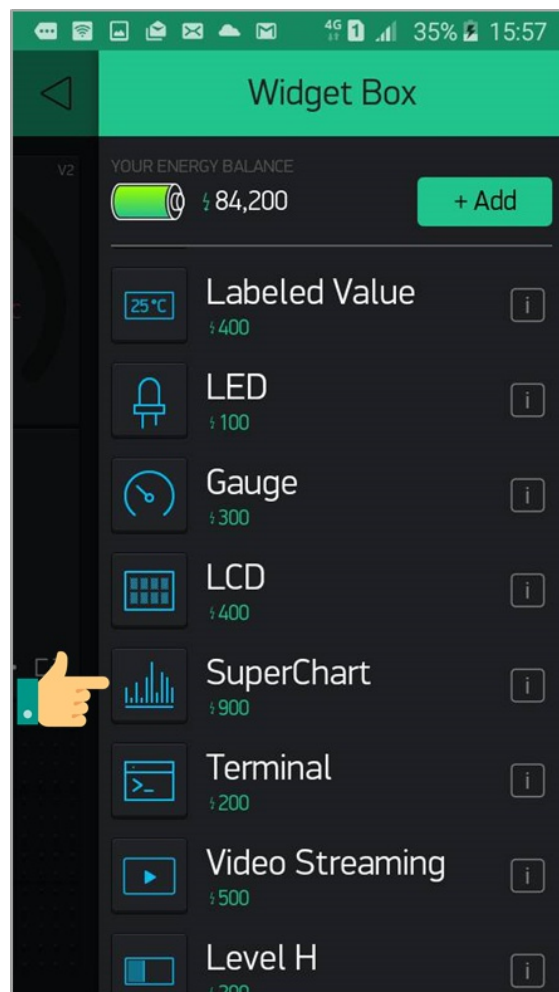


ตั้งค่าตามในภาพ

EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

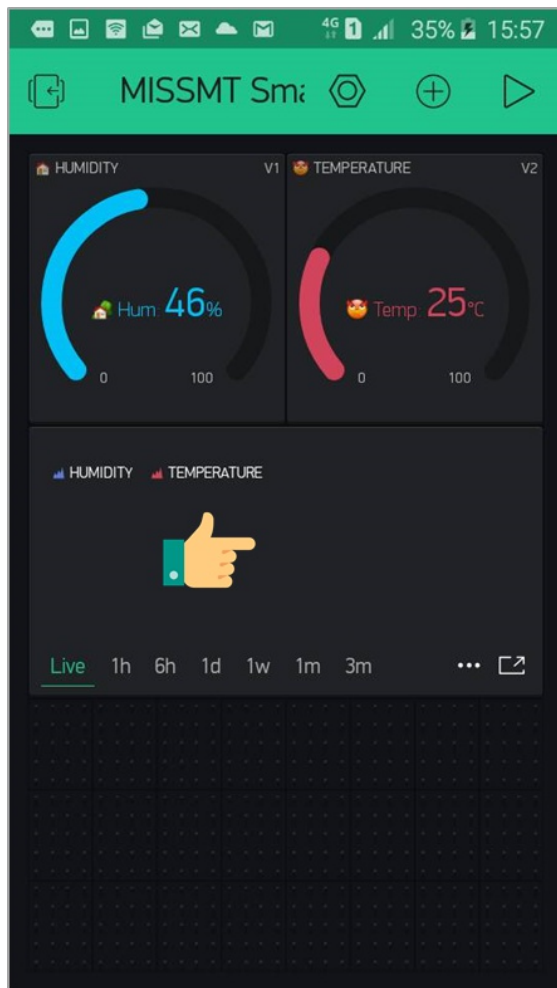


เลือกตั้งค่า Server

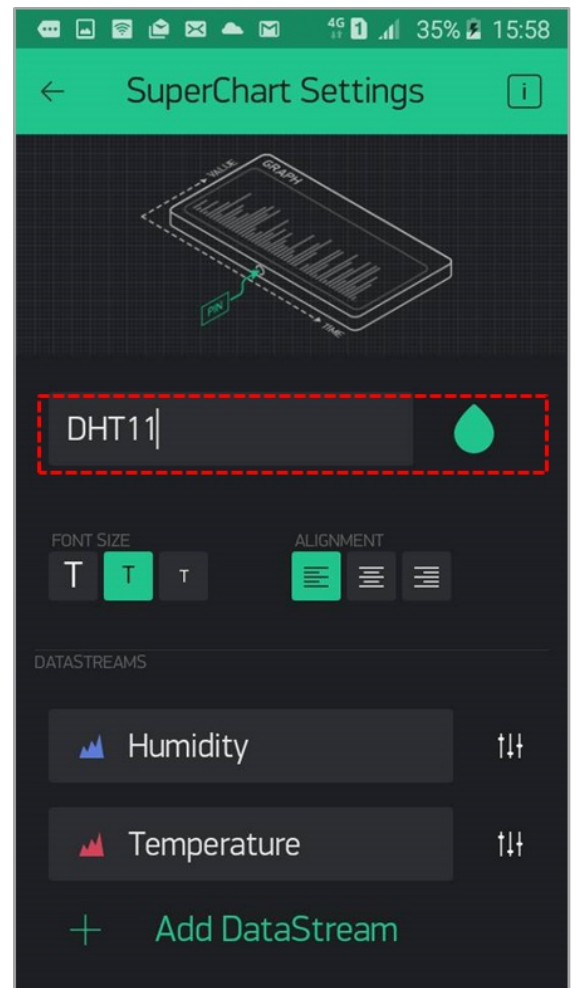


ระบุ IP Server และพอร์ต

EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

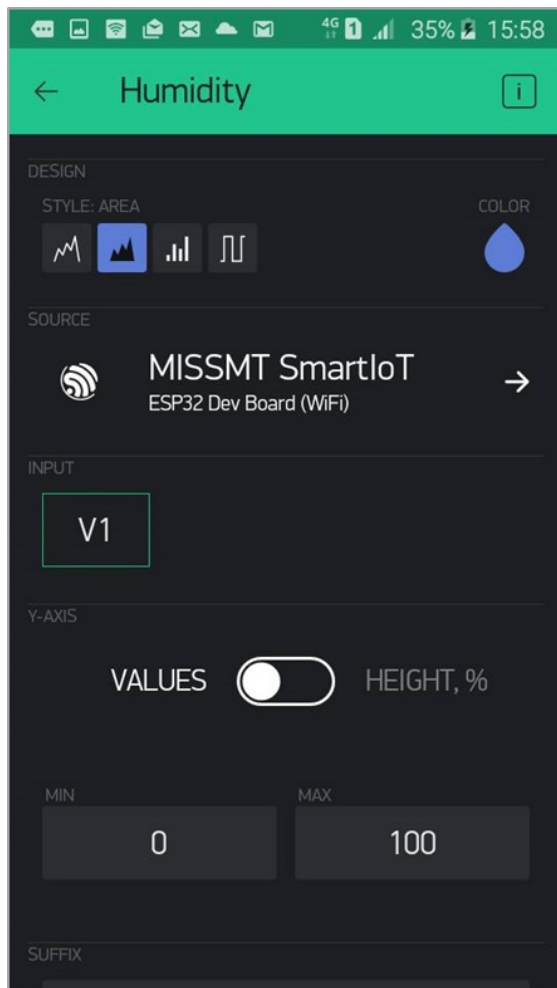


เพิ่มเครื่องมือ

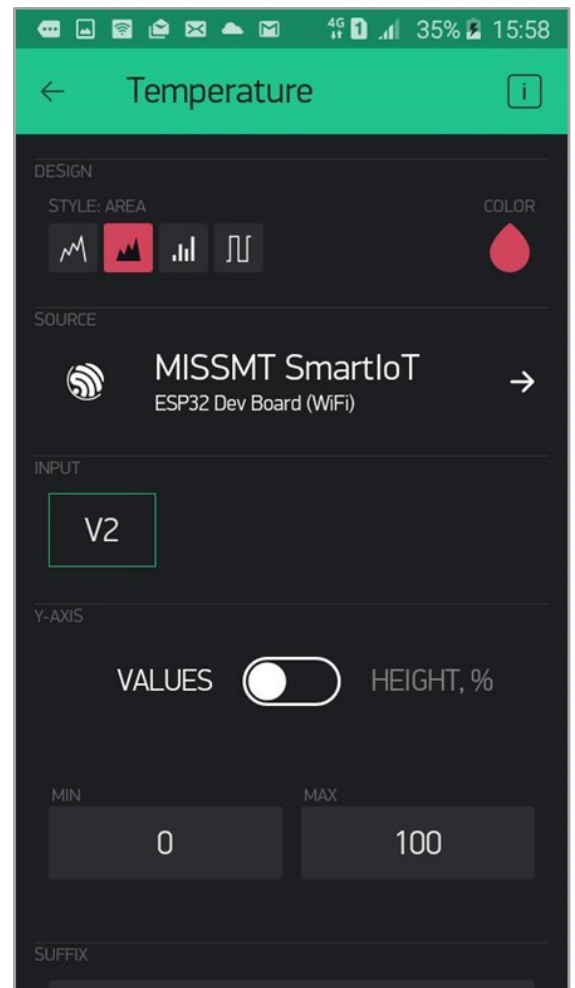


ตั้งค่าตามนิภาพ

EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

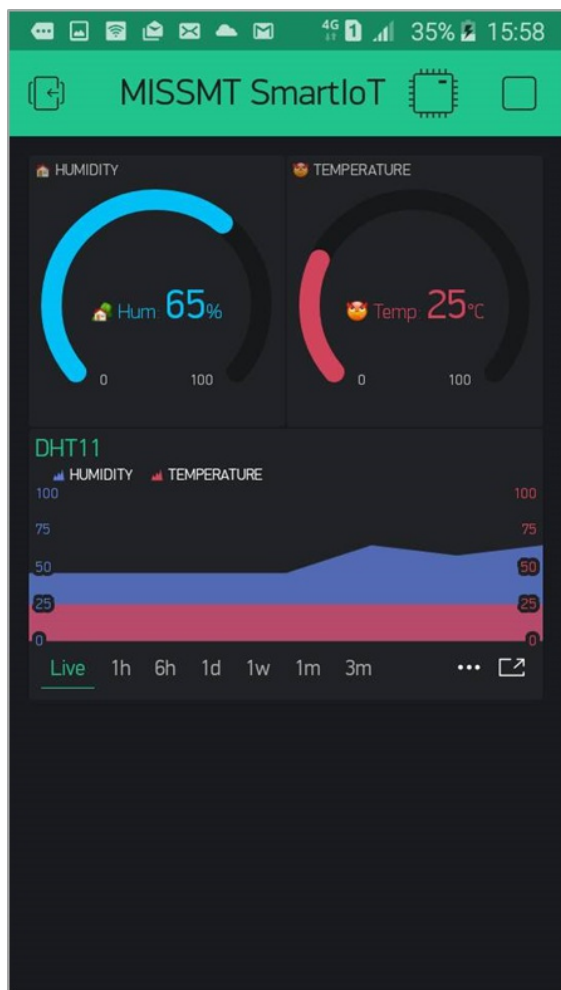


Add DataStream



Add DataStream

EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

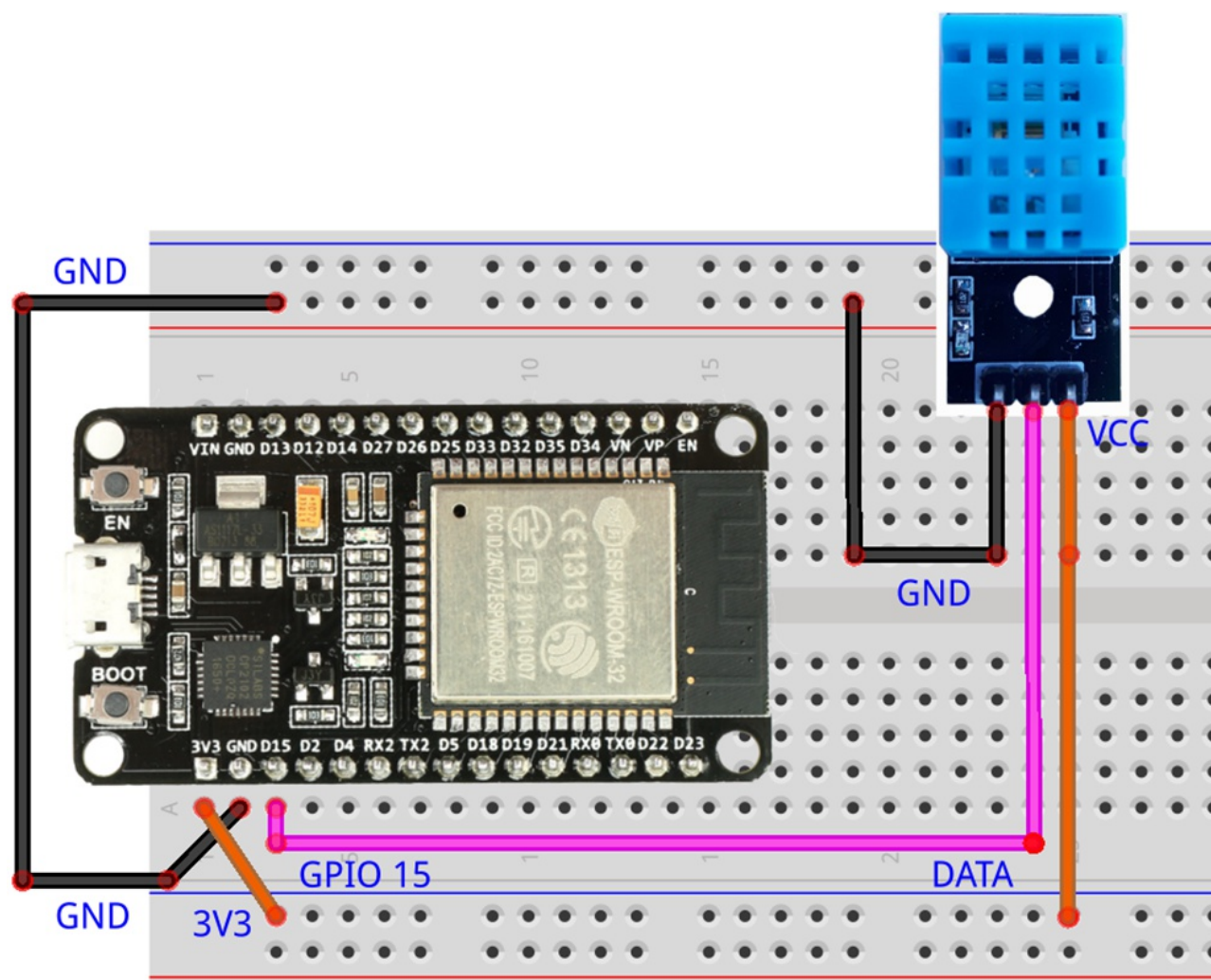


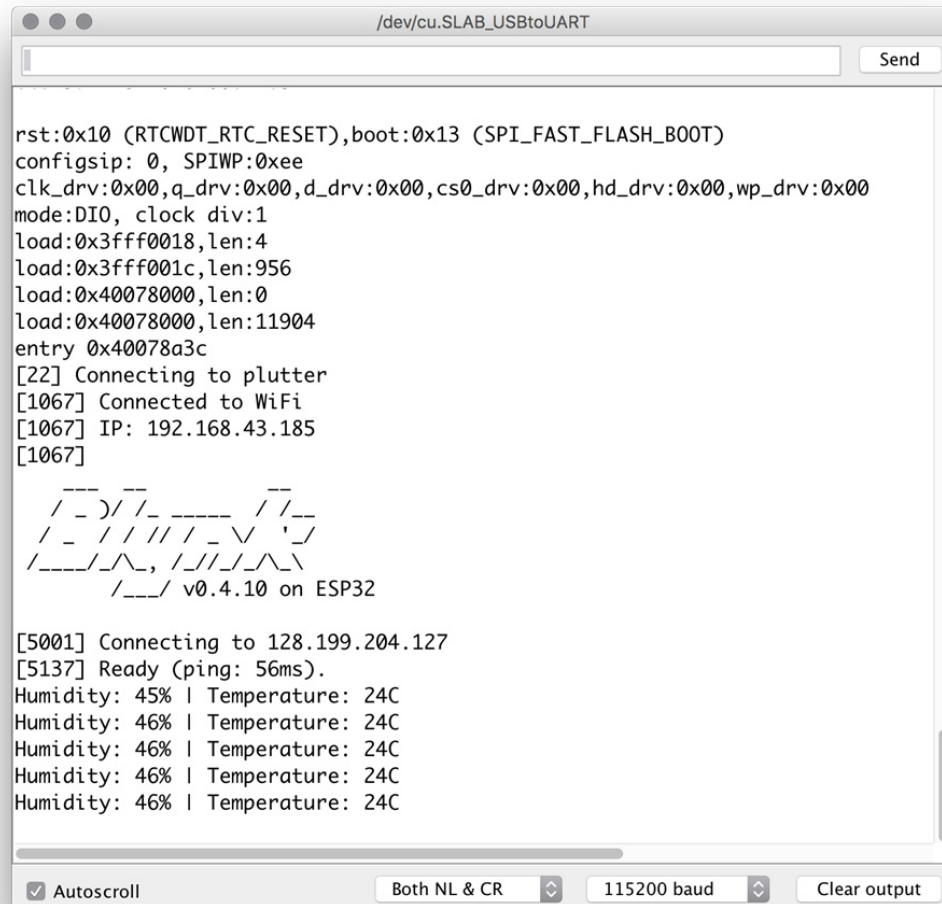
SuperChart



SuperChart

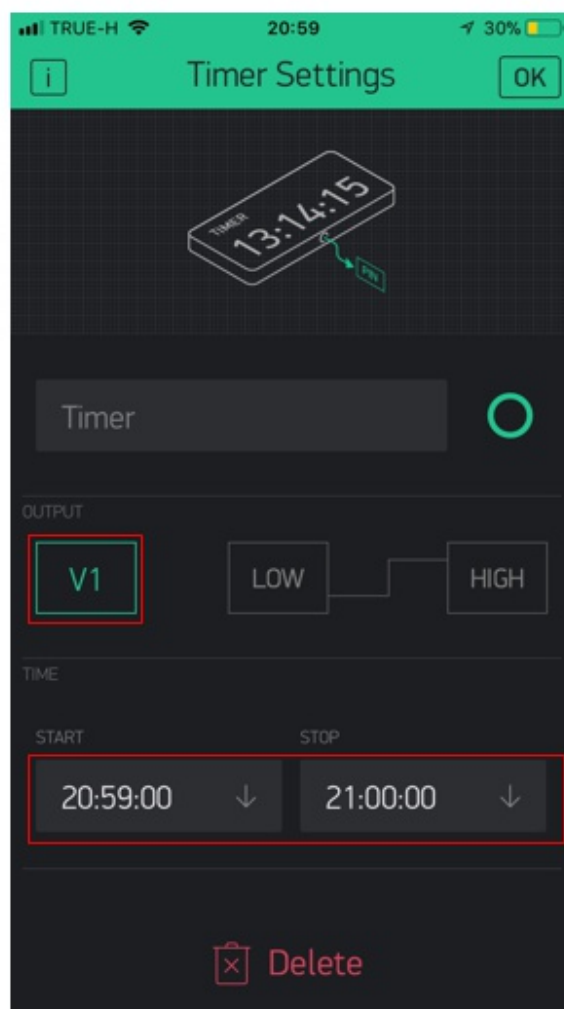
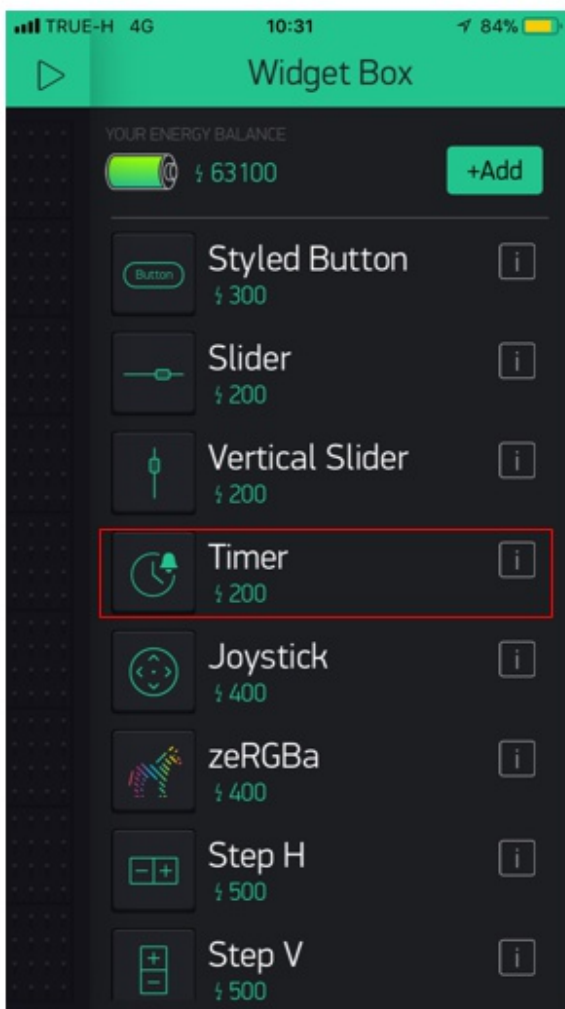
EX15 รายงานอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

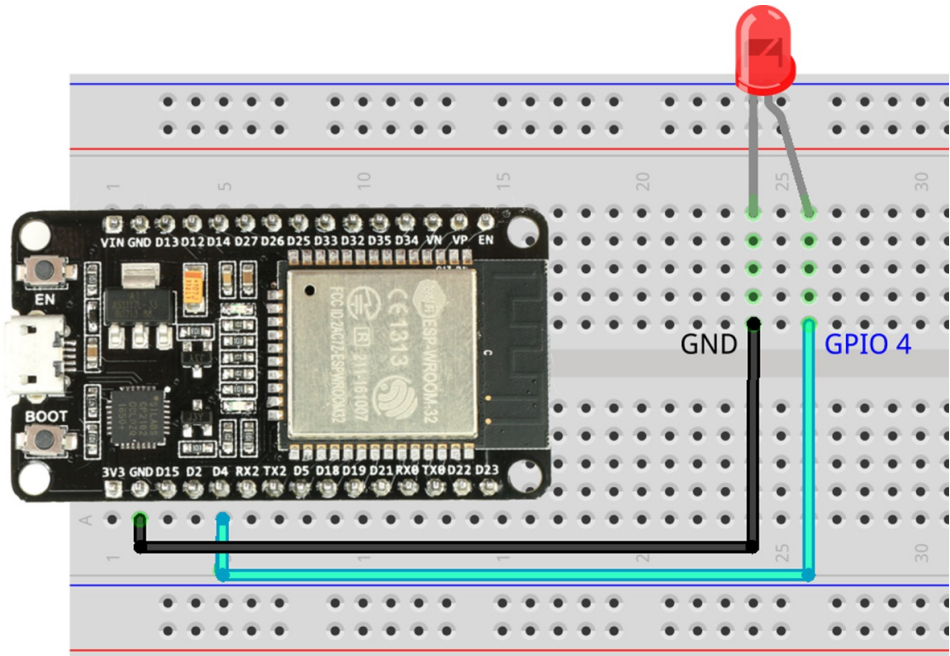
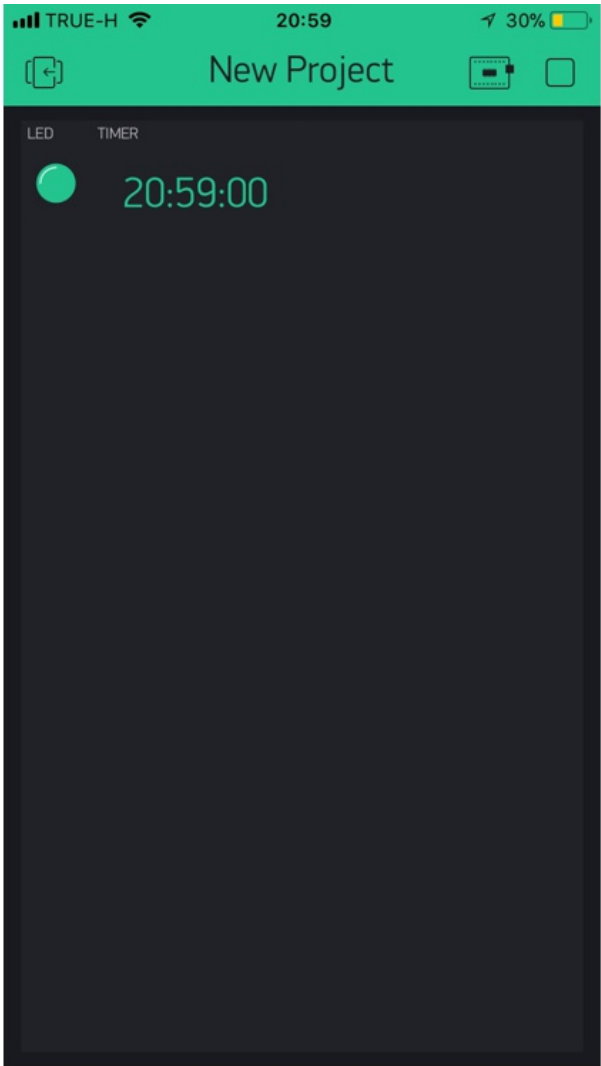
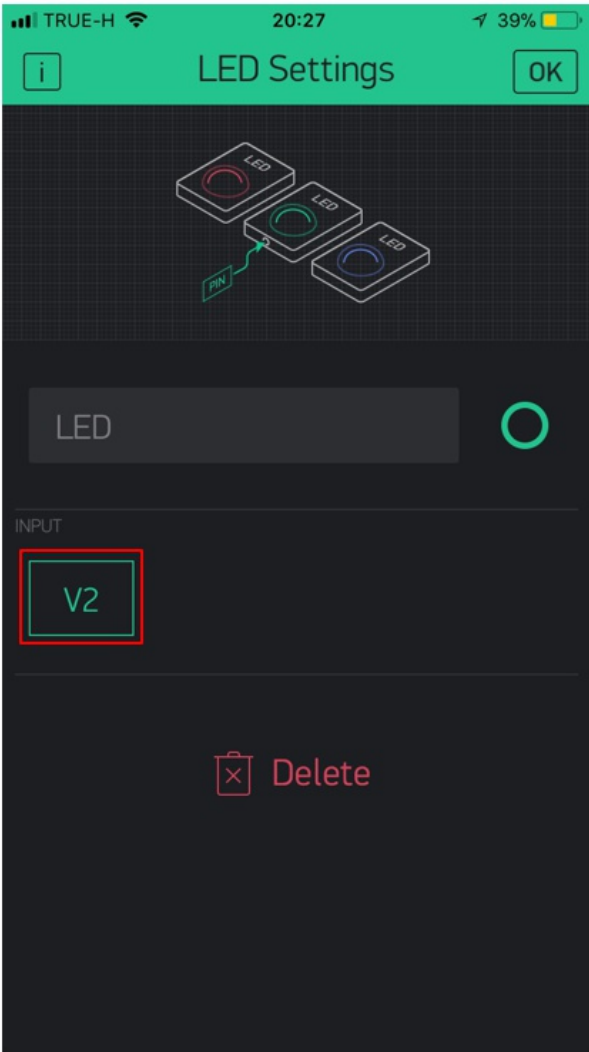




EX15-1 Blynk Timer

ตั้งเวลาให้ LED เปิด และ ปิด ตามเวลาที่กำหนด





```
EX15-1_Timer

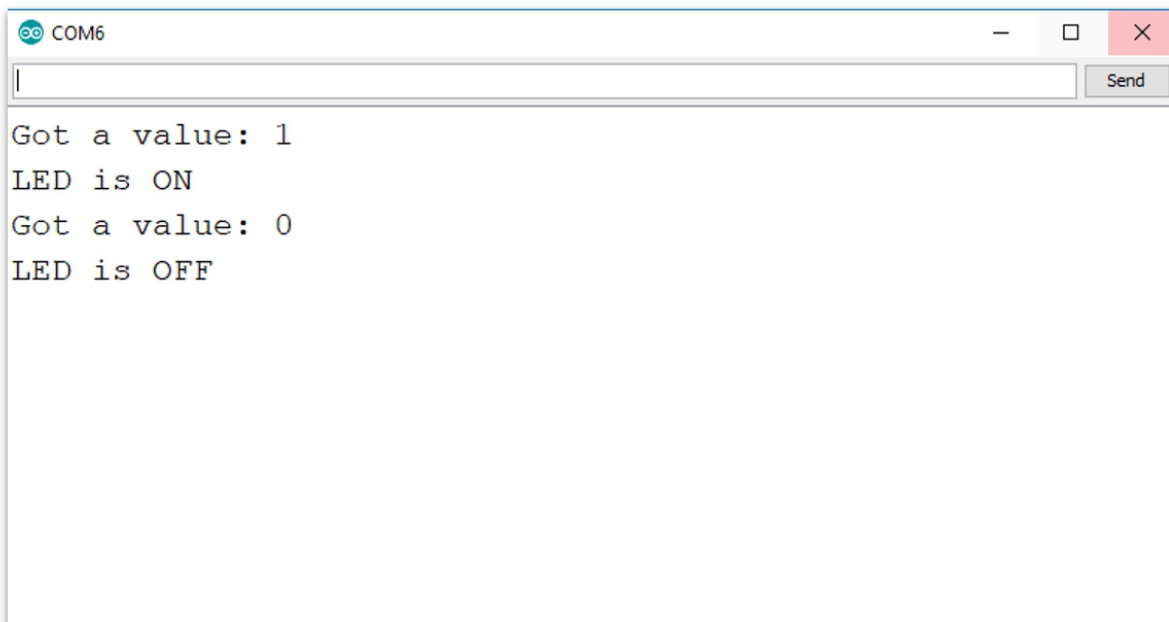
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiClientSecure.h>
#include <BlynkSimpleEsp32.h>

#define TestLED 4 // on board LED pin assignment

char auth[] = "b1343c84b1204752b8d4681e28156aa1";
char ssid[] = "Kerdsawas 2.4G";
char pass[] = "aoyaomaum";

BlynkTimer timer;
WidgetLED ledStatus(V2);

BLYNK_WRITE(V1)
{
    // You'll get HIGH/1 at startTime and LOW/0 at stopTime.
    // this method will be triggered every day
    // until you remove widget or stop project or
    // clean stop/start fields of widget
    Serial.print("Got a value: ");
    Serial.println(param.asInt());
    int data = param.asInt();
    if(data == 1)
    {
        digitalWrite(TestLED, HIGH);
    }
}
```

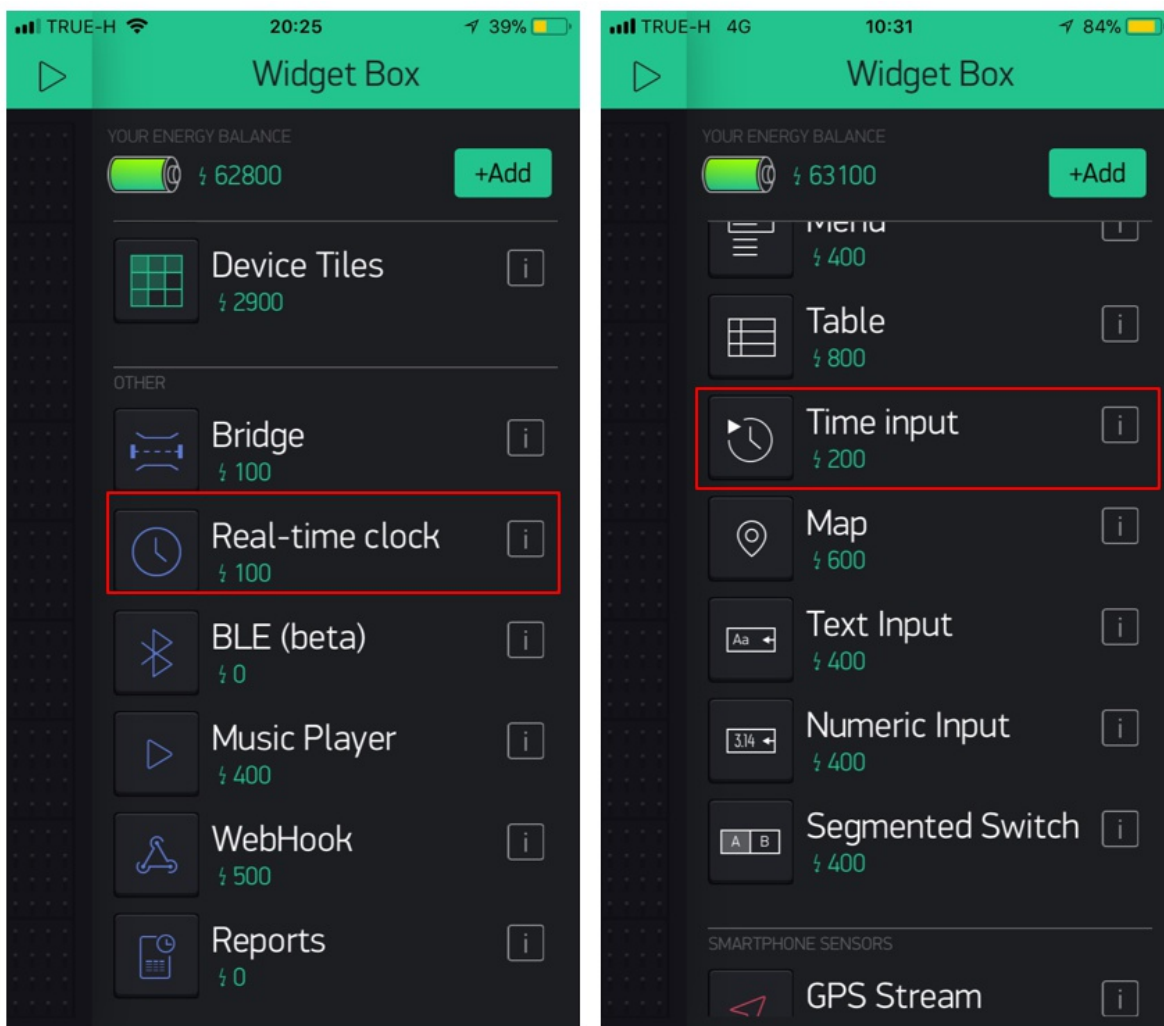


```
COM6

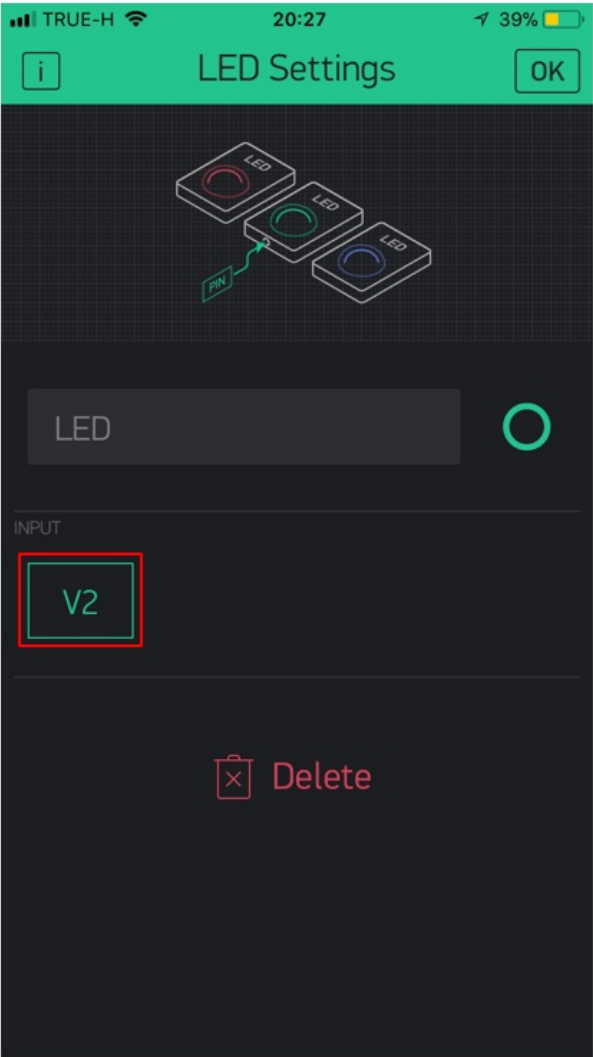
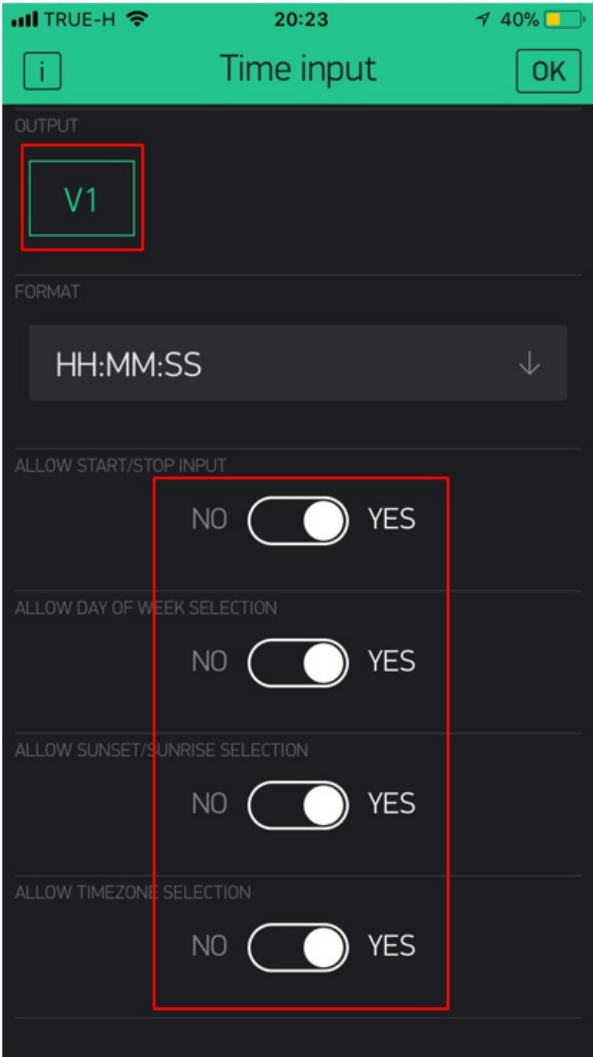
Got a value: 1
LED is ON
Got a value: 0
LED is OFF
```


EX15-2 BlynkTime Input

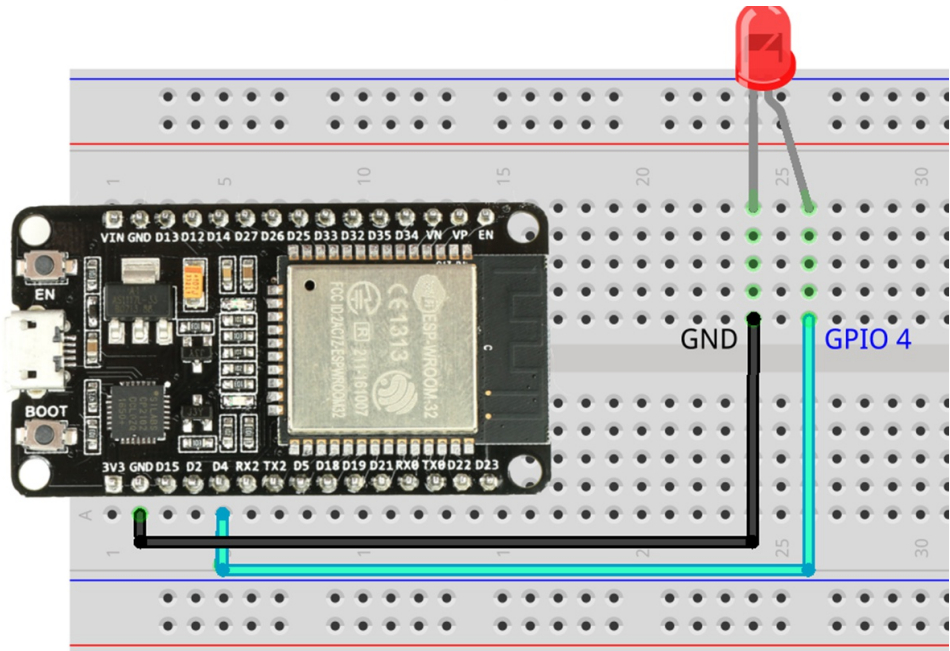
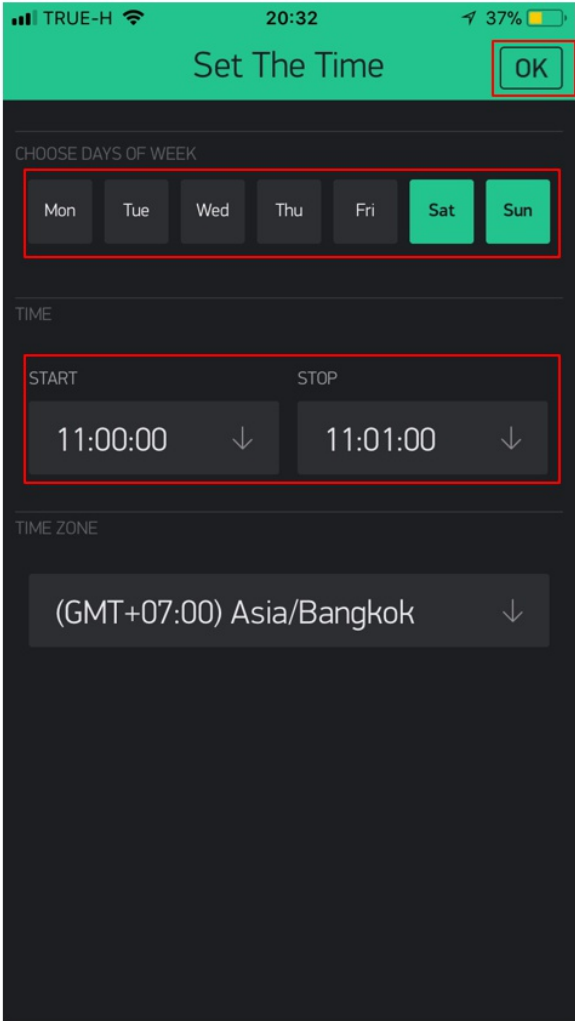
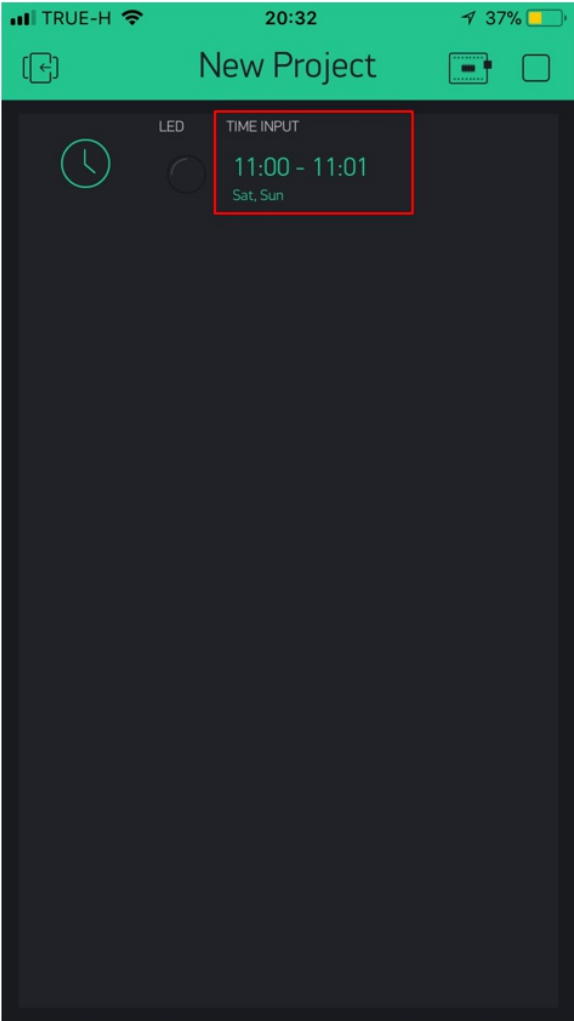
ทดสอบตั้งวัน เวลา ที่จะให้ Led เปิด และ ปิด ตามวันเวลาที่กำหนด



EX15-2 Time Input



EX15-2 Time Input



```

EX15-2_TimeInput$
BlynkTimer timer;

WidgetRTC rtc;
WidgetLED ledStatus(V2);
//BLYNK_ATTACH_WIDGET(rtc, V0);

#define TestLED 4 // on board LED pin assignment
char Date[16];
char Time[16];
char auth[] = "2fe2bdd3339e40f5a7991b1b2414c167";
char ssid[] = "Kerdsawas 2.4G";
char pass[] = "aoyaomaum";
long startsecondswd; // weekday start time in seconds
long stopsecondswd; // weekday stop time in seconds
long nowseconds; // time now in seconds

void setup() {

  pinMode(TestLED, OUTPUT);
  digitalWrite(TestLED, LOW); // set LED OFF
  Serial.begin(115200);
  Serial.println("\nStarting");
  Blynk.begin(auth, ssid, pass, IPAddress(128,199,204,127), 8080);

  int mytimeout = millis() / 1000;
  while (Blynk.connect() == false) { // try to connect to server for 10 seconds

```

COM6

Send

```

Schedule ACTIVE today
Start: 20:17
Stop : 20:18
Time zone: Asia/Bangkok
Time zone offset: 25200
Day 1 is selected
Day 2 is selected
Day 3 is selected
Day 4 is selected
Day 5 is selected
Day 6 is selected
Day 7 is selected
LED is ON at 20:17
LED is OFF at 20:18

```

Blynk HTTP RESTful API

ทำให้เราสามารถอ่าน เขียนข้อมูล ไปยัง PIN ใน Mobile Application ได้อย่างง่ายดาย ดังนั้นเราสามารถคอนโทรลบอร์ด nodeMCU บน web browser ได้ด้วยอย่างง่ายๆ โดยใช้ช่องทางของ Blynk API เช่นสามารถสั่งปิดเปิด, ดาวน์โหลด History อุณหภูมิในรูปแบบ CSV ไฟล์, เช็คสถานะ PIN, เรียกดูค่าของ Sensor ที่ต่อ, ดูรายละเอียด Widget ทั้งหมดของโปรเจกต์ใน Blynk และอื่นๆ ในรูปแบบข้อมูลแบบ JSON

- Get pin value

128.199.204.127:8080/#token#/get/V1

- Write pin value via GET

128.199.204.127:8080/#token#/update/V1?value=1

- Hardware network status

128.199.204.127:8080/#token#/isHardwareConnected

- Application network status

128.199.204.127:8080/#token#/isAppConnected

- Pin history data

128.199.204.127:8080/#token#/data/V1

- Get project

128.199.204.127:8080/#token#/project

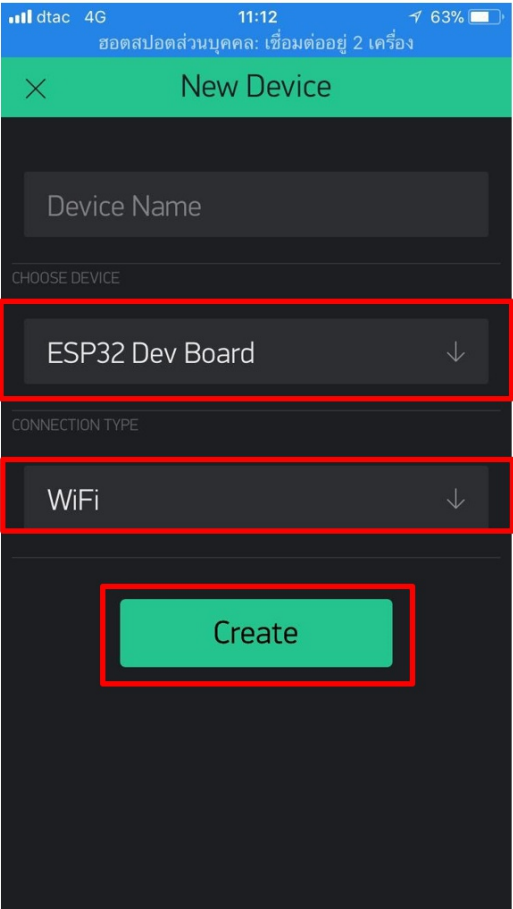
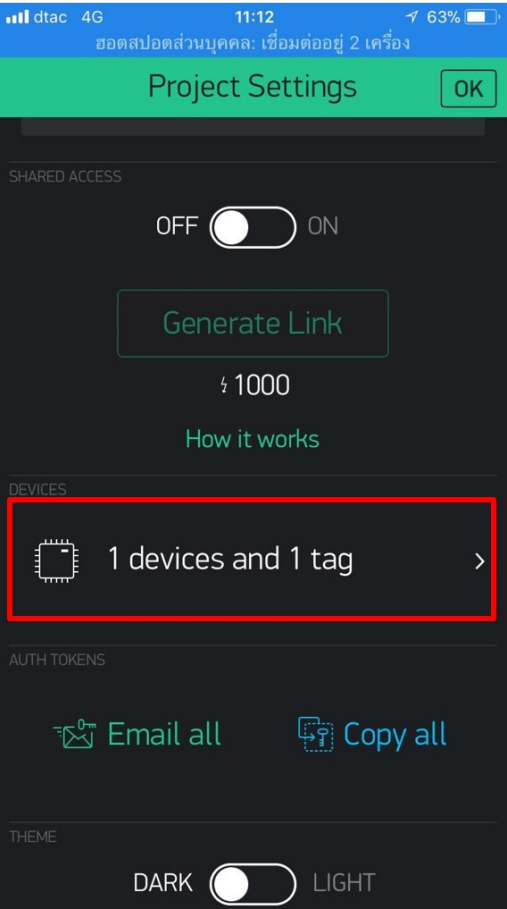
- QR for project cloning

128.199.204.127:8080/#token#/qr

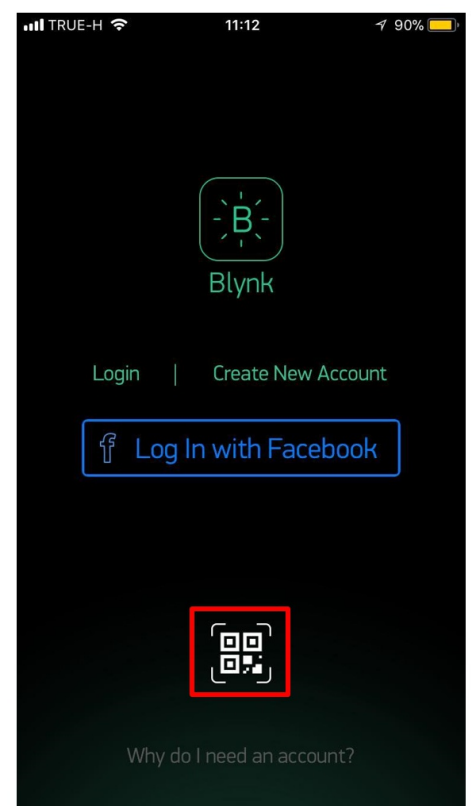
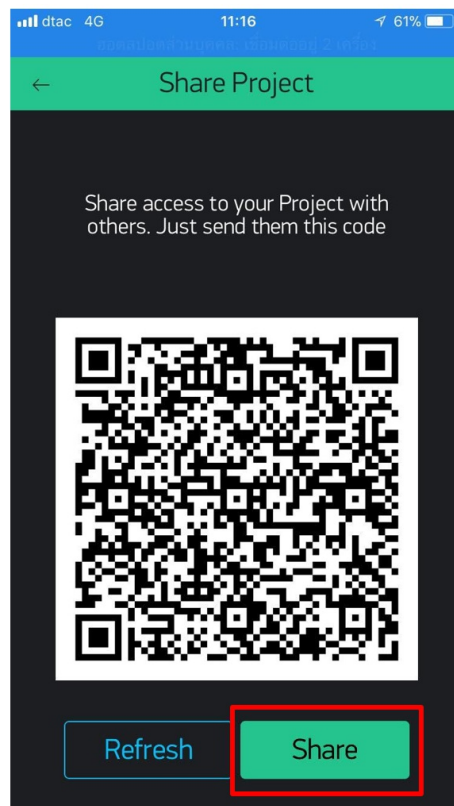
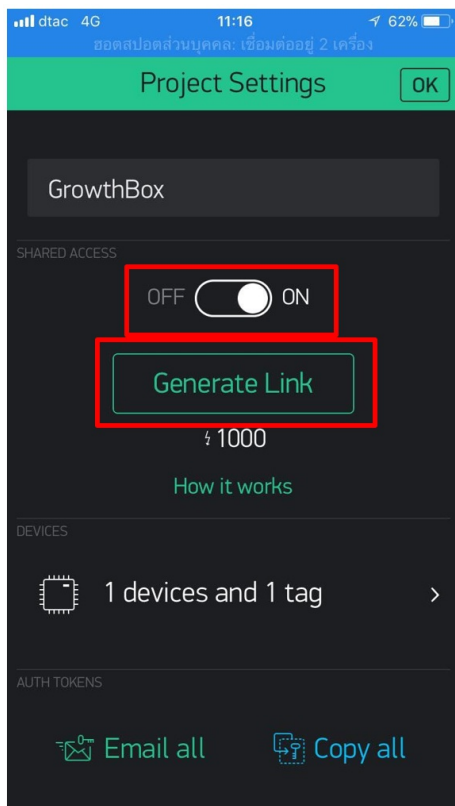
ดูรายละเอียดเพิ่มเติม

<https://blynkapi.docs.apiary.io/#>

Blynk Add Device



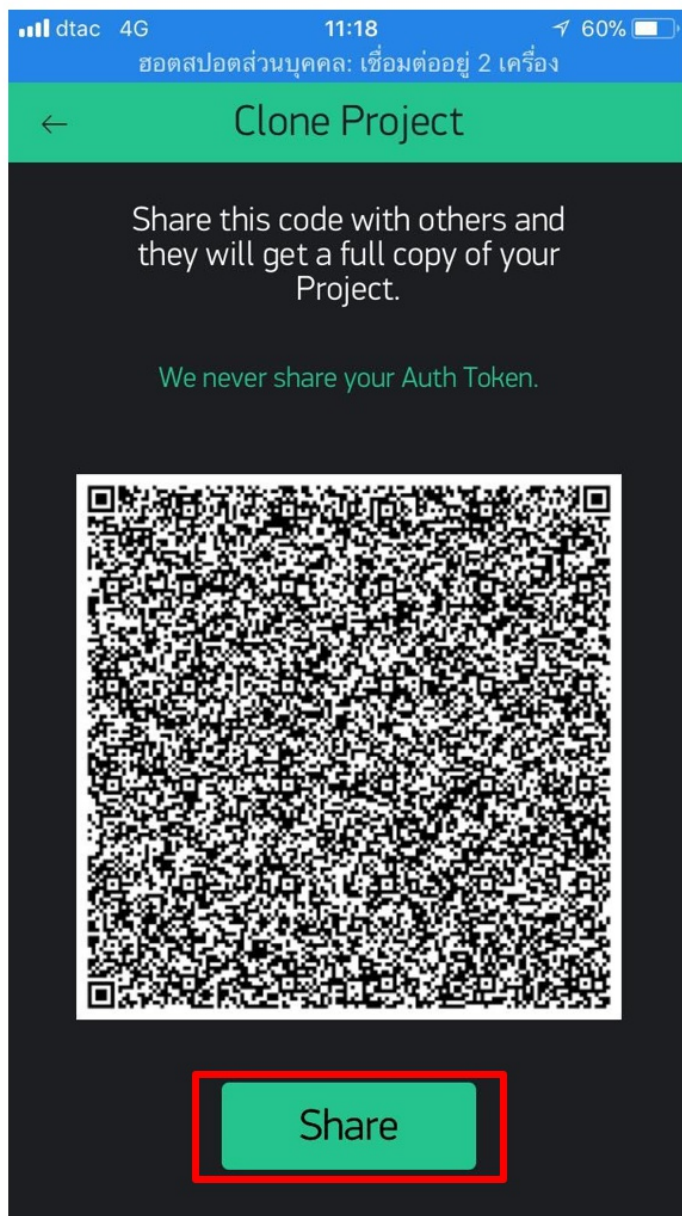
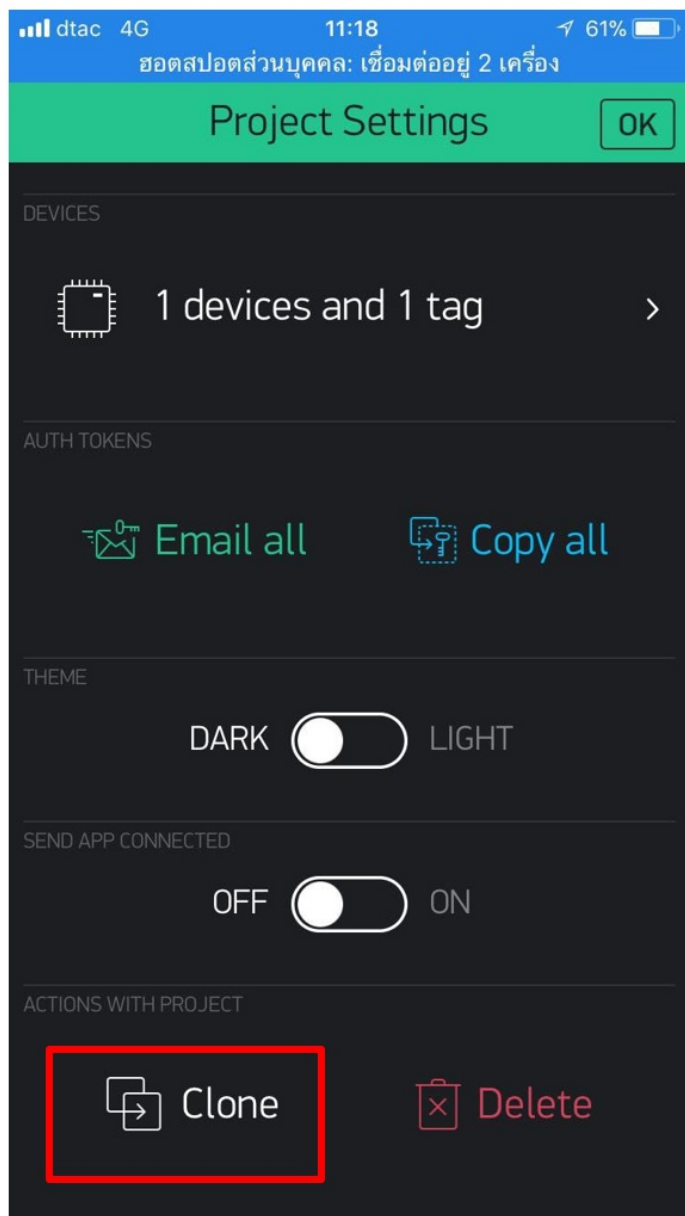
Blynk Share Controller



Blynk Clone Controller



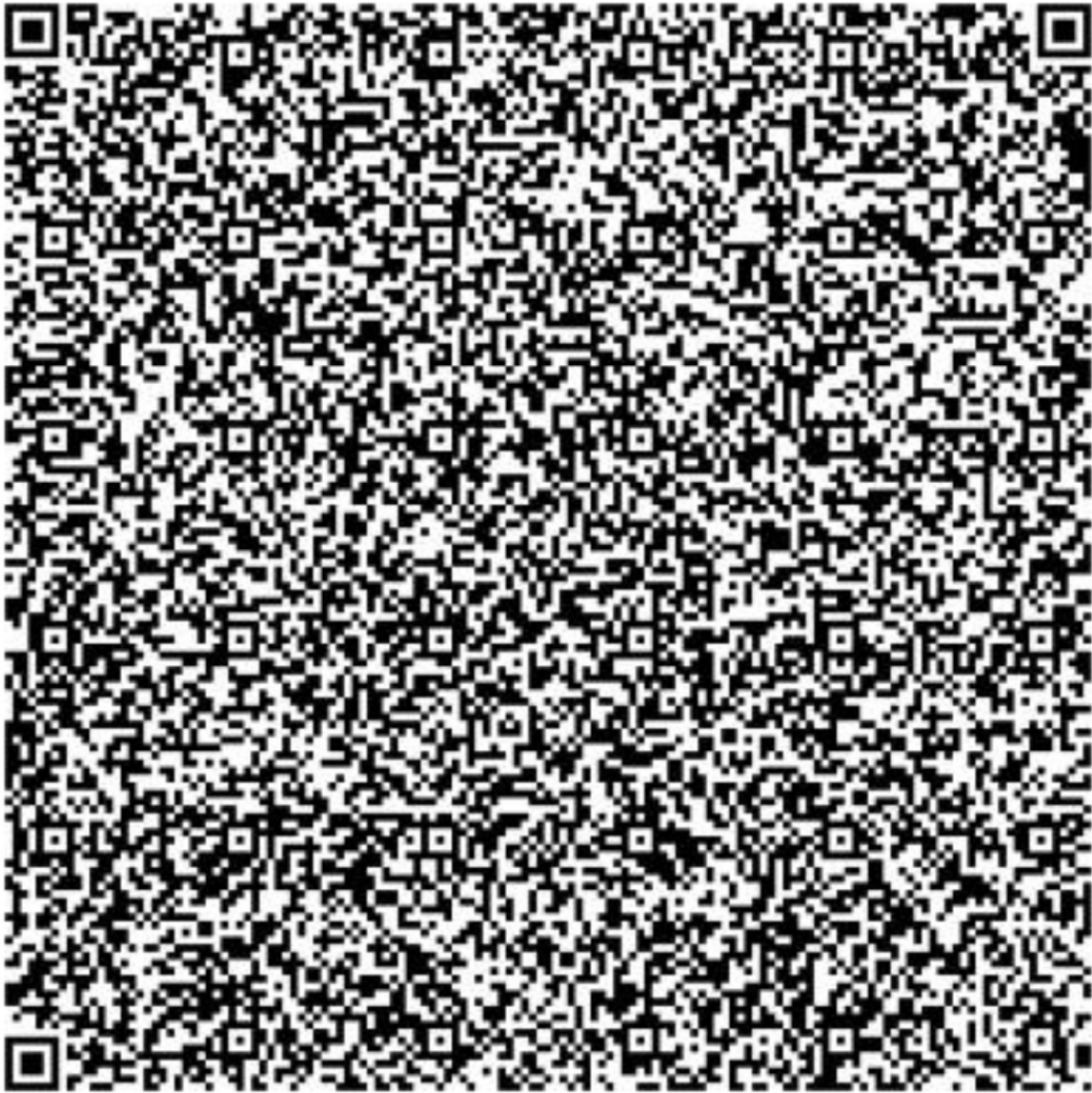
Blynk Clone Controller



Blynk Clone Controller



QR Smart Farm Project



ส่งการแจ้งเตือนไป LINE

ด้วย LINE API



EX16_LINE_NOTIFY

ตัวอย่างทดสอบ การเชื่อมต่อ ESP32 กับ SOCIAL MEDIA เพื่อส่งข้อมูลข่าวสาร ในที่นี้จะส่งเข้าบริการ Line Notify ซึ่งเป็นบริการฟรี ใช้งานง่ายและได้รับความนิยม

ผลลัพธ์ที่ได้คือ จะสามารถส่งข้อความจาก ESP32 ไปแสดงผลที่ Line Notify ได้ เราสามารถดูข้อความผ่านทางแอปพลิเคชันบนสมาร์ตโฟนได้

ขาที่ต่อทดสอบ คือ

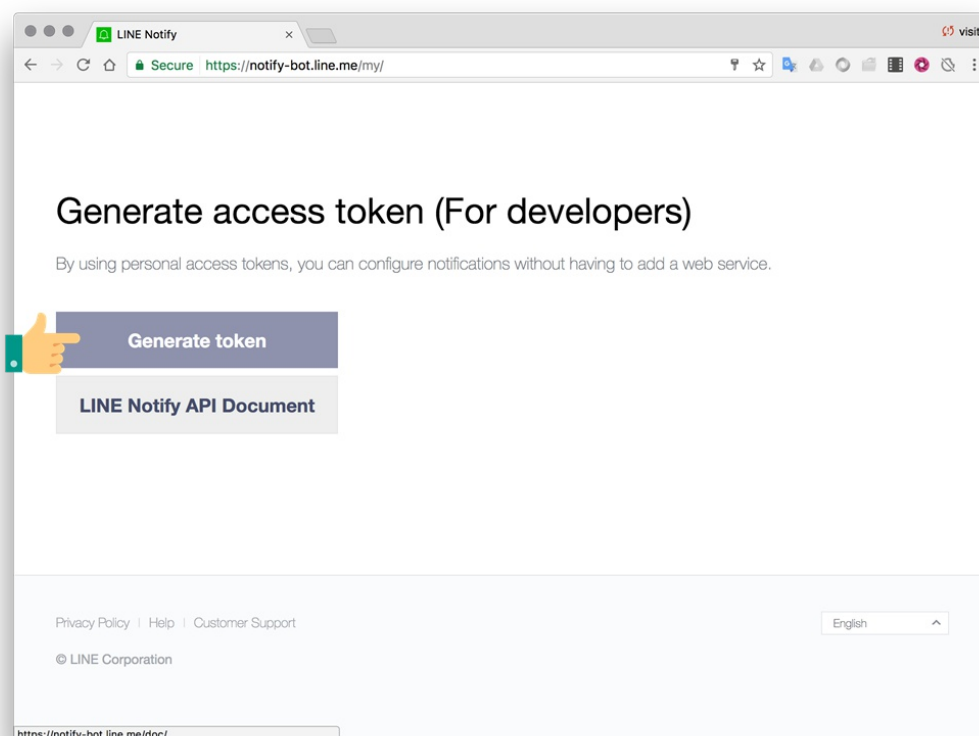
GPIO5 //Switch button

เพิ่ม LINE Notify เป็นเพื่อน

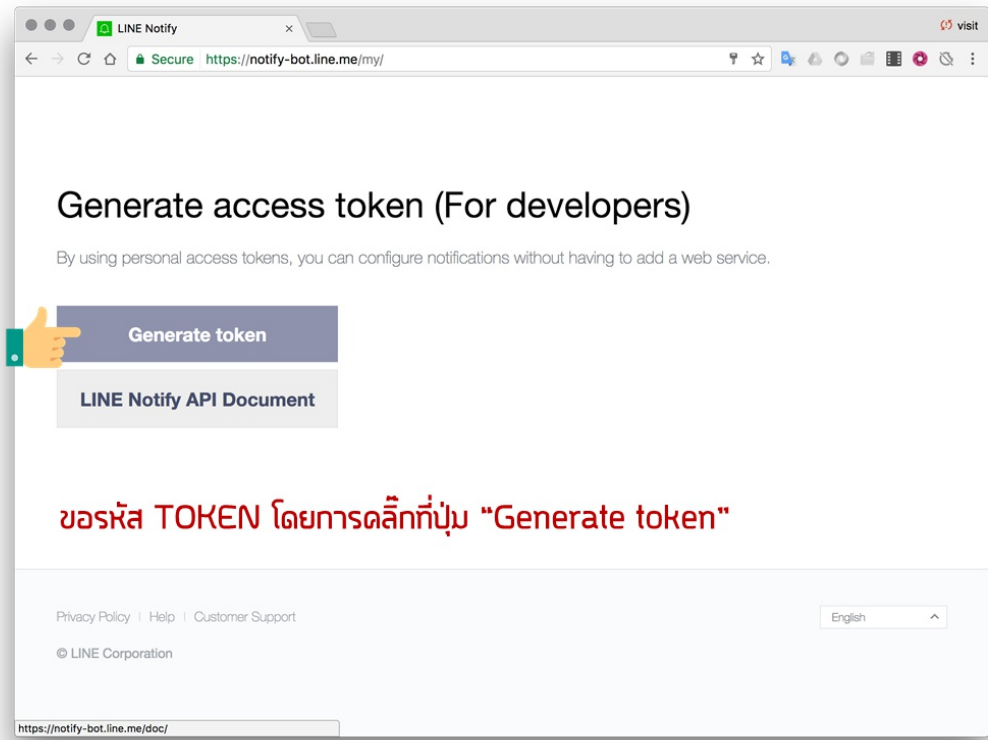


ลงทะเบียนขอรหัส Token

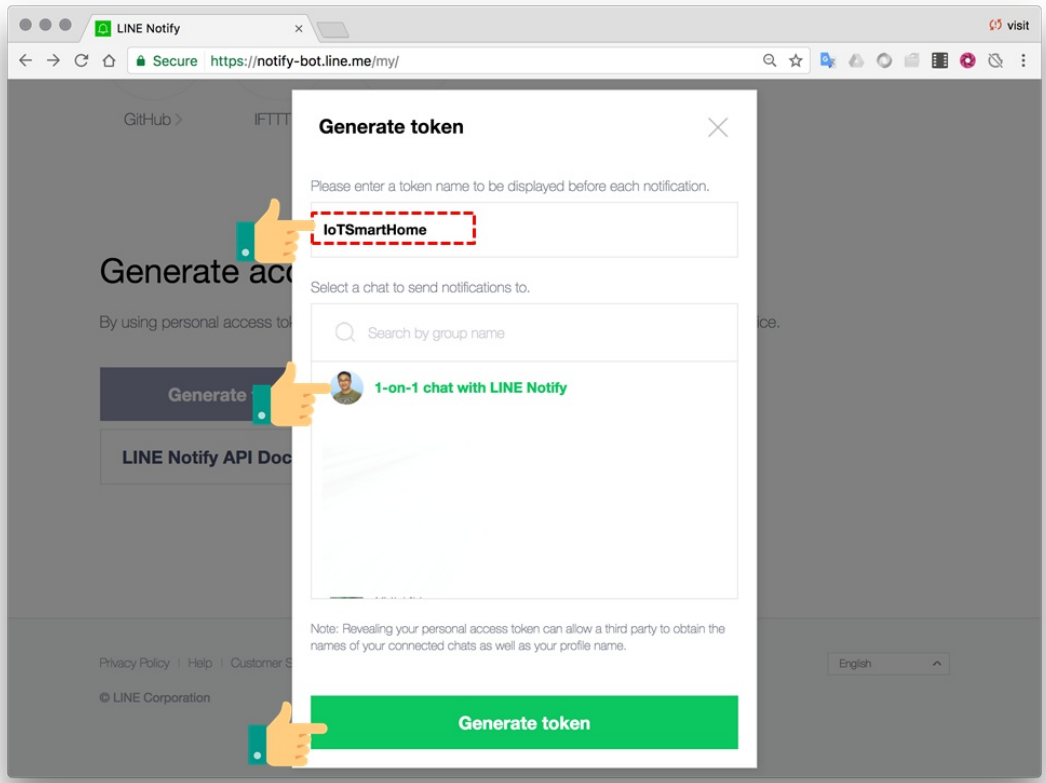
<https://notify-bot.line.me/my/>



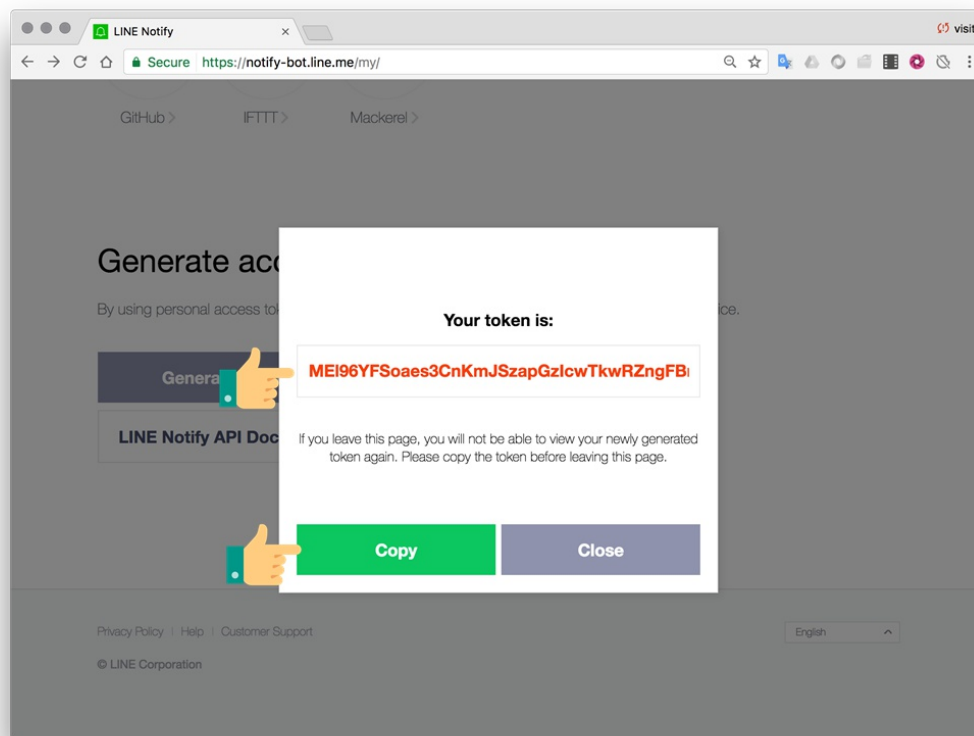
ลงทะเบียนขอรหัส Token



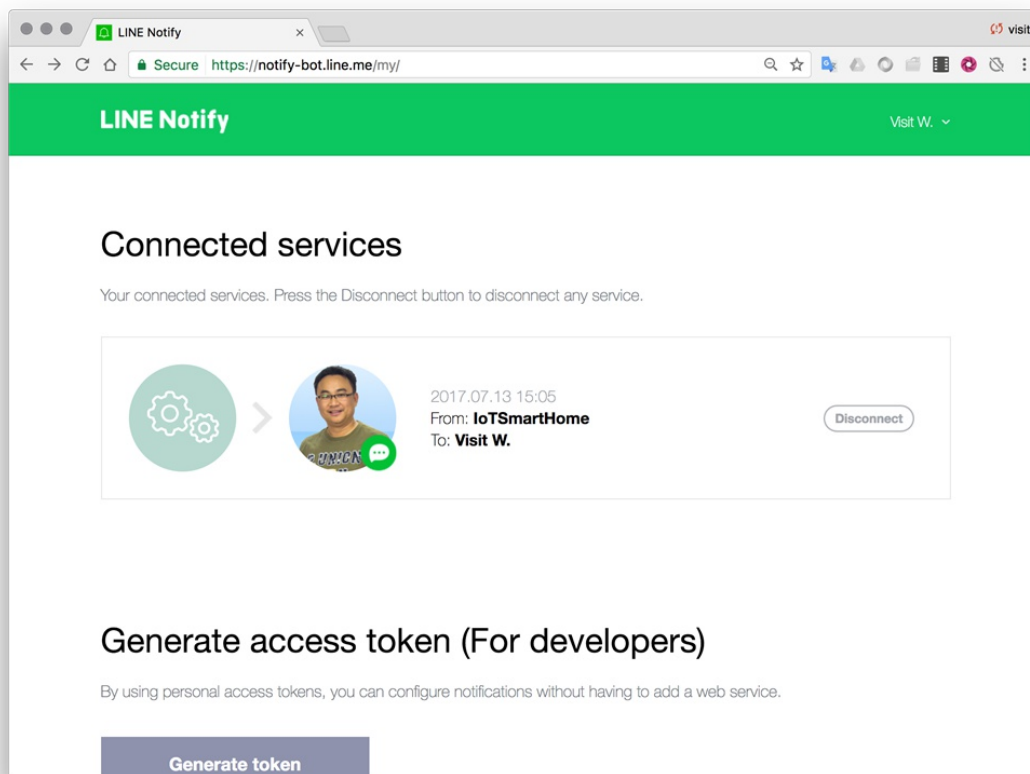
ตั้งชื่อ TOKEN และเลือกช่องทางที่จะส่งการแจ้งเตือน



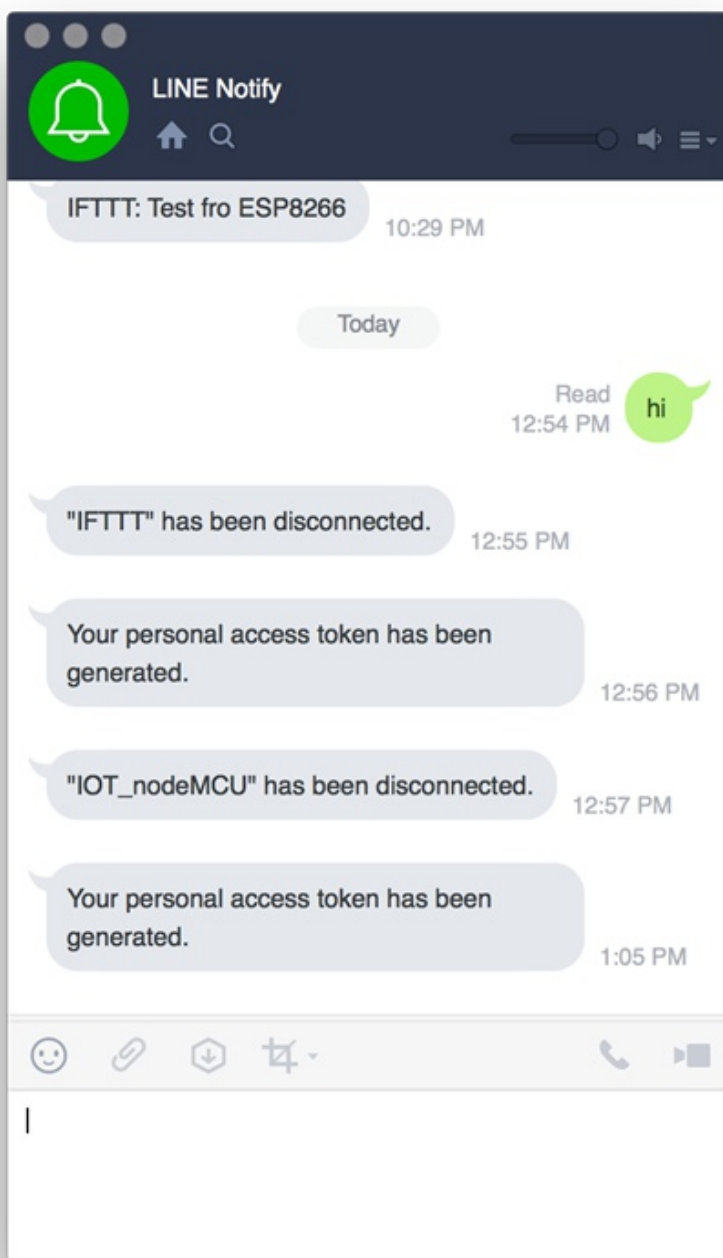
LINE จะออกรหัส TOKEN ให้ จากนั้นให้คลิกที่ปุ่ม Copy



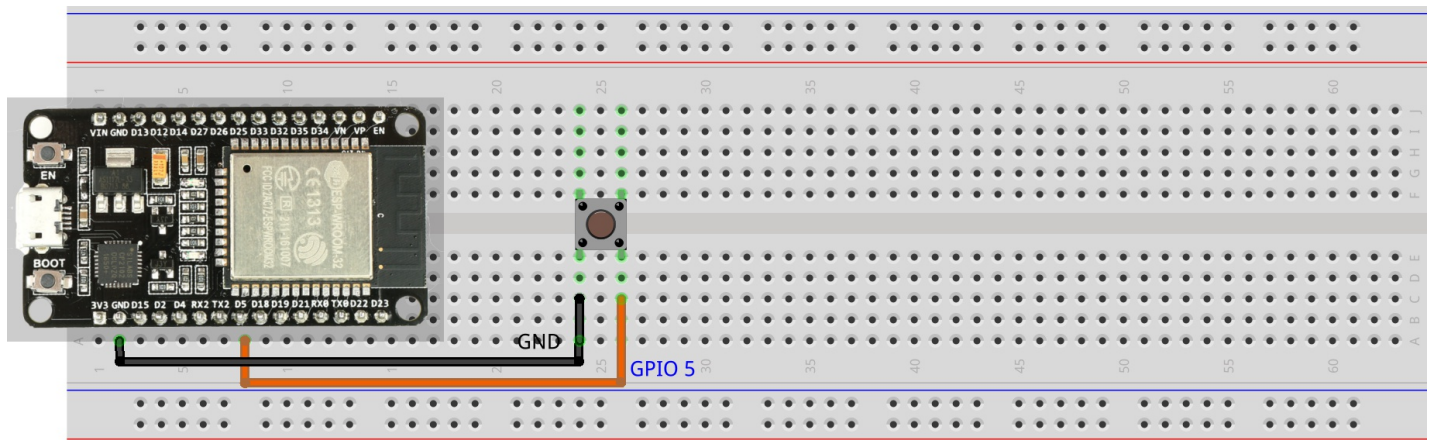
LINE ออกรหัส TOKEN ให้เรียบร้อยแล้ว



LINE ส่งข้อความแจ้งทำการออก TOKEN เสร็จเรียบร้อยแล้ว



EX16 ส่งการแจ้งเตือนไป LINE



fritzing

```

EX16_LineNotify | Arduino 1.8.5

EX16_LineNotify
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <BlynkSimpleEsp32.h>

//##### CONFIGURATION : MODIFY HERE #####

char ssid[] = "plutter"; //ชื่อวายฟาย SSID
char pass[] = "visit012"; //รหัสผ่านวายฟาย

String txt = "Switch pressed"; //ข้อความที่จะแสดงใน Line
#define TokenLine "Ynzzovc8dway8SpRSJKNvtUIAzlkPnvBPs9MUpaBWRA" //รหัสเชื่อมต่อ Line
#define SWPIN 5 //กำหนดขาสวิตช์ที่เชื่อมต่อกับสวิตช์

//#####

boolean oldState;
boolean data;

void setup() {

  pinMode(SWPIN, INPUT_PULLUP); //กำหนดให้ขาดีจิตอล SWPIN รับข้อมูลจากสวิตช์
  Serial.begin(115200); //ตั้งค่าคอนโซล
  WiFi.begin(ssid, pass); //เริ่มการเชื่อมต่อกับวายฟายแม่ข่าย
  Serial.print("WiFi Connecting"); //แสดง "WiFi Connecting" ในคอนโซล

  //ตรวจสอบสถานะการเชื่อมต่อวายฟาย ถ้าไม่สำเร็จให้แสดง "." ในคอนโซลจนกว่าจะเชื่อมต่อได้
  while (WiFi.status() != WL_CONNECTED) {
    .
  }
}

Done Saving.
Leaving...
Hard resetting...

75 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

```

<https://www.urlencoder.org>

Encode to URL encoded format

Simply use the form below

ยินดีต้อนรับสู่การอบรมเชิงปฏิบัติการ IoT

> ENCODE <

☐ Live mode OFF

Encodes while you type or paste.

☐ UPLOAD FILE

Encodes an entire file (max. 10MB).

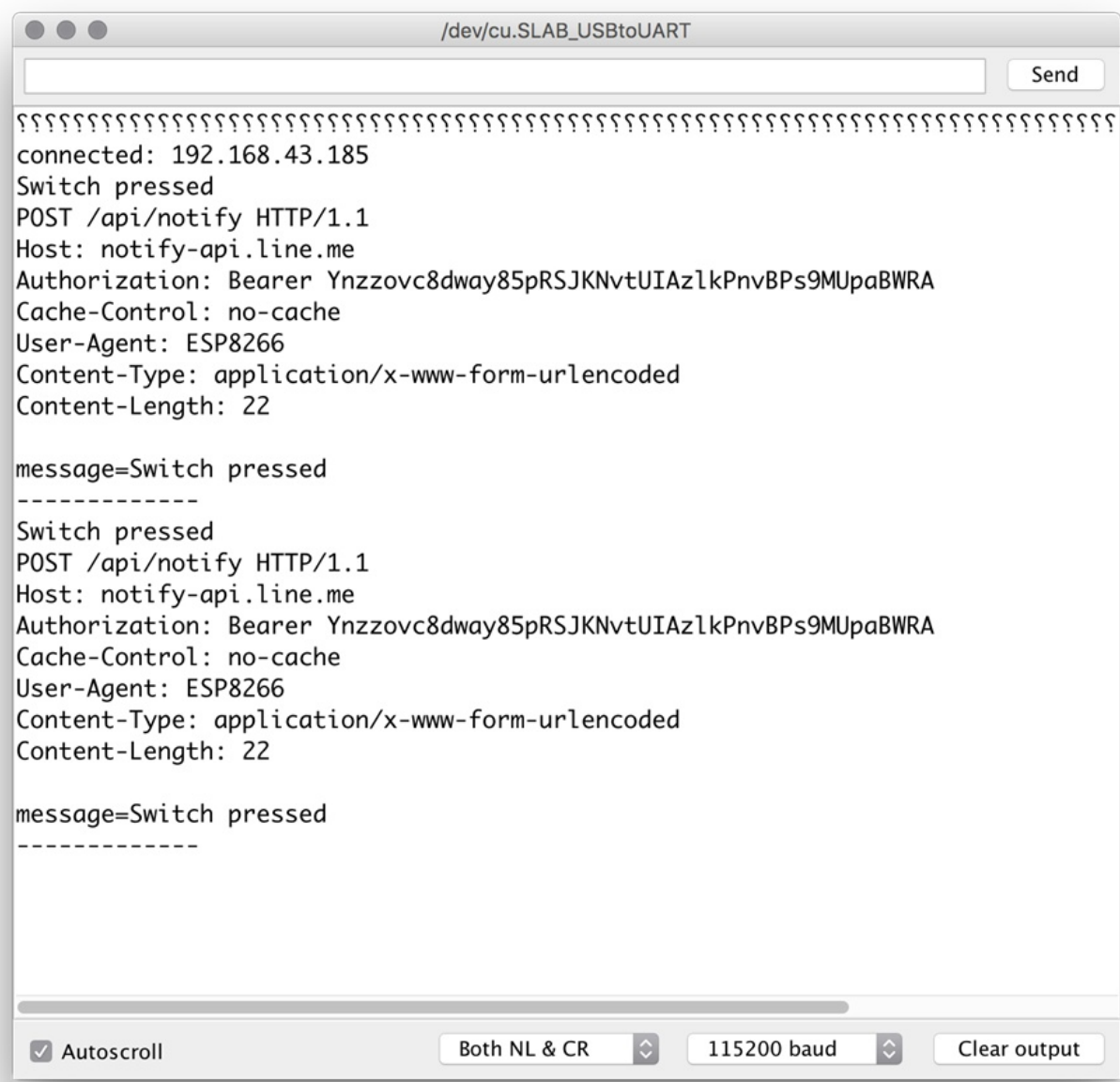


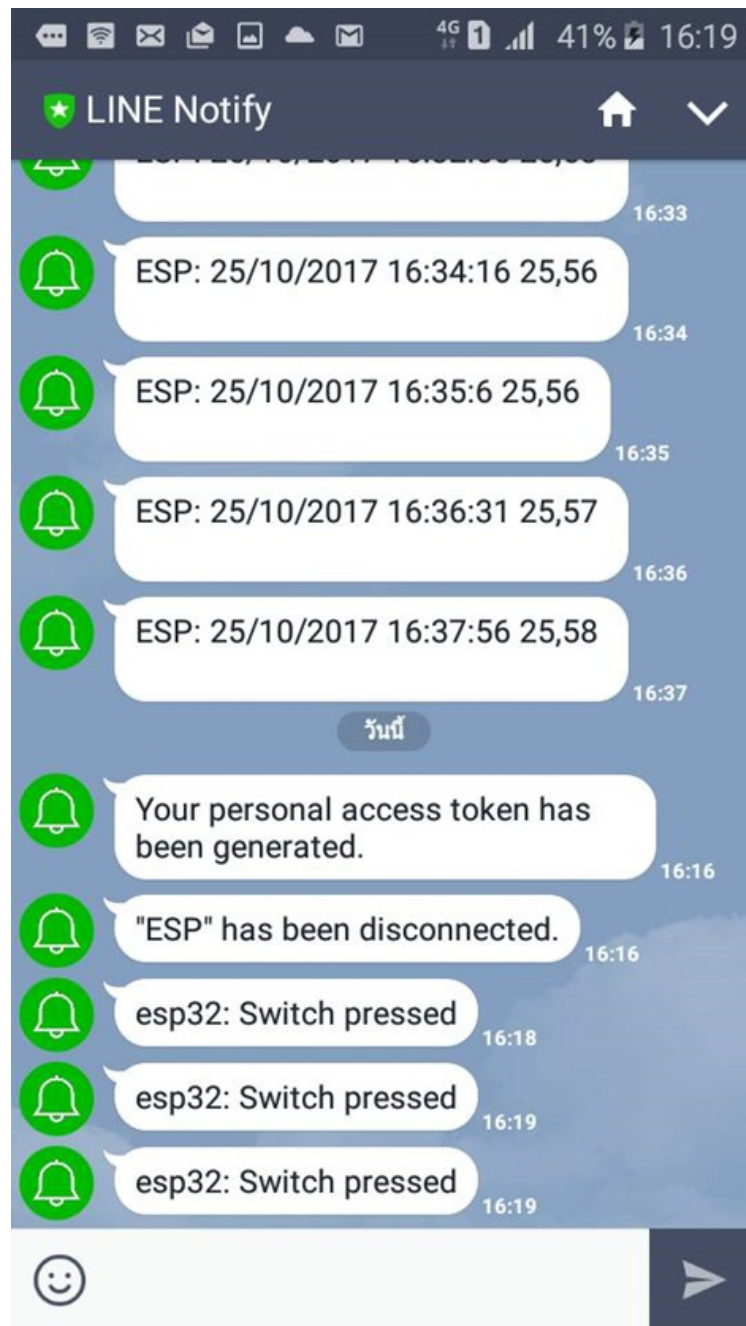
Google Cloud Platform.



%E0%B8%A2%E0%B8%B4%E0%B8%99%E0%B8%94%E0%B8%B5%E0%B8%95%E0%B9%89
 %E0%B8%AD%E0%B8%99%E0%B8%A3%E0%B8%B1%E0%B8%9A%E0%B8%AA%E0%B8%B
 9%E0%B9%88%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%AD%E0%B8%9A%E0%B8%
 A3%E0%B8%A1%E0%B9%80%E0%B8%8A%E0%B8%B4%E0%B8%87%E0%B8%9B%E0%B8%
 8F%E0%B8%B4%E0%B8%9A%E0%B8%B1%E0%B8%95%E0%B8%B4%E0%B8%81%E0%B8%
 B2%E0%B8%A3%20IoT

EX16 ส่งการแจ้งเตือนไป LINE





โหมดประหยัดพลังงาน

Deep Sleep on ESP32



Table: Functionalities Depending on the Power Modes

Power mode	Active	Modem-sleep	Light-sleep	Deep-sleep	Hibernation
Sleep pattern	Association sleep pattern			ULP sensor-monitored pattern	-
CPU	ON	PAUSE	ON	OFF	OFF
Wi-Fi/BT base-band and radio	ON	OFF	OFF	OFF	OFF
RTC	ON	ON	ON	ON	OFF
ULP co-processor	ON	ON	ON	ON/OFF	OFF

Sleep Pattern

- Association sleep pattern: The power mode switches between the active mode and Modem-sleep/ Light- sleep mode during this sleep pattern. The CPU, Wi-Fi, Bluetooth, and radio are woken up at predetermined intervals to keep Wi-Fi/BT connections alive.
- ULP sensor-monitored pattern: The main CPU is in the Deep-sleep mode. The ULP co-processor does sensor measurements and wakes up the main system, based on the measured data from sensors.

Power mode

- Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.
- Modem-sleep mode: The CPU is operational and the clock is configurable. The Wi-Fi/Bluetooth base- band and radio are disabled.
- Light-sleep mode: The CPU is paused. The RTC and ULP-coprocessor are running. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
- Deep-sleep mode: Only RTC is powered on. Wi-Fi and Bluetooth connection data are stored in RTC memory. The ULP-coprocessor can work.
- Hibernation mode: The internal 8MHz oscillator and ULP-coprocessor are disabled. The RTC recovery memory are power-down. Only one RTC timer on the slow clock and some RTC GPIOs are active. The RTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.

Table: Power Consumption by Power Modes

Power mode	Description	Power consumption
Active (RF working)	Wi-Fi Tx packet 13 dBm ~ 21 dBm	160 ~ 260 mA
	Wi-Fi / BT Tx packet 0 dBm	120 mA
	Wi-Fi / BT Rx and listening	80 ~ 90 mA
	Association sleep pattern (by Light-sleep)	0.9 mA@DTIM3, 1.2 mA@DTIM1
Modem-sleep	The CPU is powered on.	Max speed: 20 mA
		Normal speed: 5 ~ 10 mA
		Slow speed: 3 mA
Light-sleep	-	0.8 mA
Deep-sleep	The ULP co-processor is powered on.	0.5 mA
	ULP sensor-monitored pattern	25 μ A @1% duty
	RTC timer + RTC memory	10 μ A
Hibernation	RTC timer only	2.5 μ A

Table: RF Power Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	225	-	mA
Transmit 802.11b, CCK 11 Mbps, POUT = +18.5 dBm	-	205	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	160	-	mA
Transmit 802.11n, MCS7, POUT = +14 dBm	-	152	-	mA
Receive 802.11b, packet length = 1024 bytes, -80 dBm	-	85	-	mA
Receive 802.11g, packet length = 1024 bytes, -70 dBm	-	85	-	mA
Receive 802.11n, packet length = 1024 bytes, -65 dBm	-	80	-	mA
Receive 802.11n HT40, packet length = 1024 bytes, -65 dBm	-	80	-	mA

The current consumption measurements are conducted with 3.0 V supply and 25°C ambient, at antenna port. All the transmitters' measurements are based on 90% duty cycle and continuous transmit mode.

ความหมายของหน่วยทางไฟฟ้า

mV มิลลิโวลต์ คือหน่วยที่ใช้เรียกขนาดของแรงดันไฟฟ้าหนึ่งในพันของโวลต์ เช่น 1mV หมายถึงขนาดของแรงดันไฟฟ้า 1 มิลลิโวลต์ เท่ากับ 0.001 โวลต์ ขนาดของแรงดันไฟฟ้า 1000mV มีค่าเท่ากับ 1 โวลต์

V (Volt) โวลต์ คือหน่วยที่ใช้เรียกขนาดของแรงดันไฟฟ้า เช่น 220V หมายถึงขนาดของแรงดันไฟฟ้า 220 โวลต์

mA มิลลิแอมป์ คือหน่วยที่ใช้เรียกปริมาณของกระแสไฟฟ้าที่ไหลผ่านหนึ่งในพันของแอมป์ เช่น 1mA หมายถึงปริมาณของกระแสไฟฟ้าที่ไหลผ่านมีขนาด 1 มิลลิแอมป์ เท่ากับ 0.001 แอมป์ ปริมาณของกระแสไฟฟ้าที่ไหลผ่าน 1000mA มีค่าเท่ากับ 1 แอมป์

A (Ampere) แอมป์ หรือ แอมแปร์ คือหน่วยที่ใช้เรียกปริมาณของกระแสไฟฟ้าที่ไหลผ่าน เช่น 10A หมายถึงปริมาณกระแสไฟฟ้าที่ไหลผ่าน 10 แอมป์

ตารางแปลง มิลลิวินาที เป็น วินาที

1 มิลลิวินาที = 0.001 วินาที	10 มิลลิวินาที = 0.01 วินาที	2500 มิลลิวินาที = 2.5 วินาที
2 มิลลิวินาที = 0.002 วินาที	20 มิลลิวินาที = 0.02 วินาที	5000 มิลลิวินาที = 5 วินาที
3 มิลลิวินาที = 0.003 วินาที	30 มิลลิวินาที = 0.03 วินาที	10000 มิลลิวินาที = 10 วินาที
4 มิลลิวินาที = 0.004 วินาที	40 มิลลิวินาที = 0.04 วินาที	25000 มิลลิวินาที = 25 วินาที
5 มิลลิวินาที = 0.005 วินาที	50 มิลลิวินาที = 0.05 วินาที	50000 มิลลิวินาที = 50 วินาที
6 มิลลิวินาที = 0.006 วินาที	100 มิลลิวินาที = 0.1 วินาที	100000 มิลลิวินาที = 100 วินาที
7 มิลลิวินาที = 0.007 วินาที	250 มิลลิวินาที = 0.25 วินาที	250000 มิลลิวินาที = 250 วินาที
8 มิลลิวินาที = 0.008 วินาที	500 มิลลิวินาที = 0.5 วินาที	500000 มิลลิวินาที = 500 วินาที
9 มิลลิวินาที = 0.009 วินาที	1000 มิลลิวินาที = 1 วินาที	1000000 มิลลิวินาที = 1000 วินาที

การออกจากโหมด deep sleep ของ ESP8266 และ ESP32

ESP8266 Wake-Up Source:

- Reset the chip from an external source.
- Connect GPIO16 to the reset pin for a timed wake-up.
- With the timer, we were only able to sleep for about one hour.

The ESP32 has four different wake-up sources:

- The touch pads.
- The timer.
- Two possibilities of external Wake-up.
- ULP coprocessor wake-up.

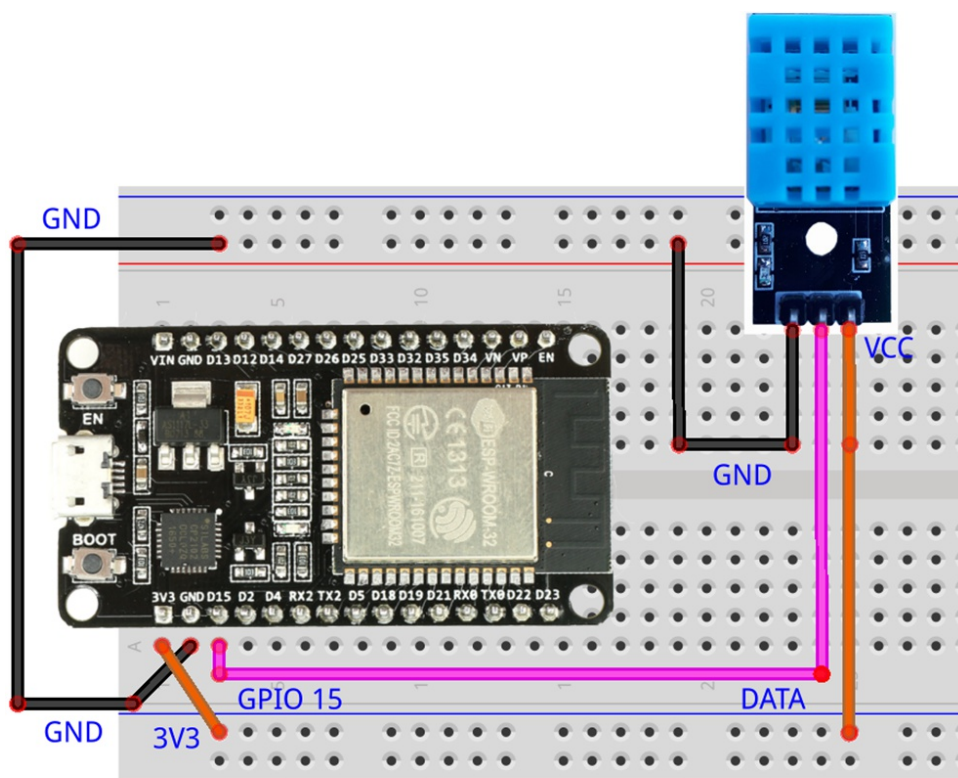
EX17_DHT11_LINE_NOTIFY_SLEEPMODE

ตัวอย่างทดสอบ การเชื่อมต่อ ESP32 กับ SOCIAL MEDIA เพื่อส่งข้อมูลข่าวสาร ในที่นี้จะส่งเข้าบริการ Line Notify ซึ่งเป็นบริการฟรี ใช้งานง่ายและได้รับความนิยม

ผลลัพธ์ที่ได้คือ จะสามารถส่งค่าอุณหภูมิและความชื้นจากเซ็นเซอร์ DHT11 ใน ESP32 ไปแสดงผลที่ Line Notify ได้ โดยจะส่งข้อความทุกๆ 1 นาที จากนั้นจะเข้าสู่ Sleep โหมดเพื่อประหยัดพลังงาน เราสามารถดูข้อความผ่านทางแอปพลิเคชันบนสมาร์ตโฟนได้

ขาที่ต่อทดสอบ คือ

GPIO15 //DHT11 sensor



EX17 แจ้งเตือนอุณหภูมิความชื้นสัมพัทธ์ออนไลน์

```

EX17_DHT11_LINE_NOTIFY | Arduino 1.8.5

#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "esp_deep_sleep.h"
#include <SimpleDHT.h> //โหลดไลบรารี DHT11

##### CONFIGURATION : MODIFY HERE #####

char ssid[] = "plutter"; //ชื่อสายฟาย SSID
char pass[] = "visit012"; //รหัสผ่านสายฟาย

#define TokenLine "Ynzzovc8dway85pRSJKNvtUIAzlkPnvBP9MUpaBWRA" //รหัสเชื่อมต่อ Line
#define SWPIN 5 //กำหนดขาสวิตช์ที่เชื่อมต่อกับสวิตช์
int DHTSENSOR = 15; //GPIO15 (ADC2_CH3)

#####

SimpleDHT11 dht11; //ระบุรุ่นเซ็นเซอร์รุ่น DHT11
unsigned long last1, last2, last3;

void setup() {

  Serial.println("Wake up.");
  esp_deep_sleep_enable_timer_wakeup(1 * 60 * 1000 * 1000); // กำหนดให้ตื่นเมื่อครบ 1 นาทีข้างหน้า

  pinMode(SWPIN, INPUT_PULLUP); //กำหนดให้ขาดีจิตอล SWPIN รับข้อมูลจากสวิตช์
  Serial.begin(115200); //ตั้งค่าคอนโซล
  WiFi.begin(ssid, pass); //เริ่มการเชื่อมต่อกับสายฟายแม่ข่าย
}

Done Saving.
Leaving...
Hard resetting...

17 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

```

```

/dev/cu.SLAB_USBtoUART

-----
Temperature: 24C | Humidity: 57RH
Go to sleep mode.
ets Jun  8 2016 00:22:57

rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:956
load:0x40078000,len:0
load:0x40078000,len:11904
entry 0x40078a3c
WiFi Connecting..
connected: 192.168.43.185
POST /api/notify HTTP/1.1
Host: notify-api.line.me
Authorization: Bearer Ynzzovc8dway85pRSJKNvtUIAzlkPnvBP9MUpaBWRA
Cache-Control: no-cache
User-Agent: ESP8266
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

message=Temperature: 24C | Humidity: 60RH
-----
Temperature: 24C | Humidity: 60RH
Go to sleep mode.

[Autoscroll] [Both NL & CR] [115200 baud] [Clear output]

```

EX17 แจ้งเตือนอุณหภูมิความชื้นสัมพัทธ์ออนไลน์



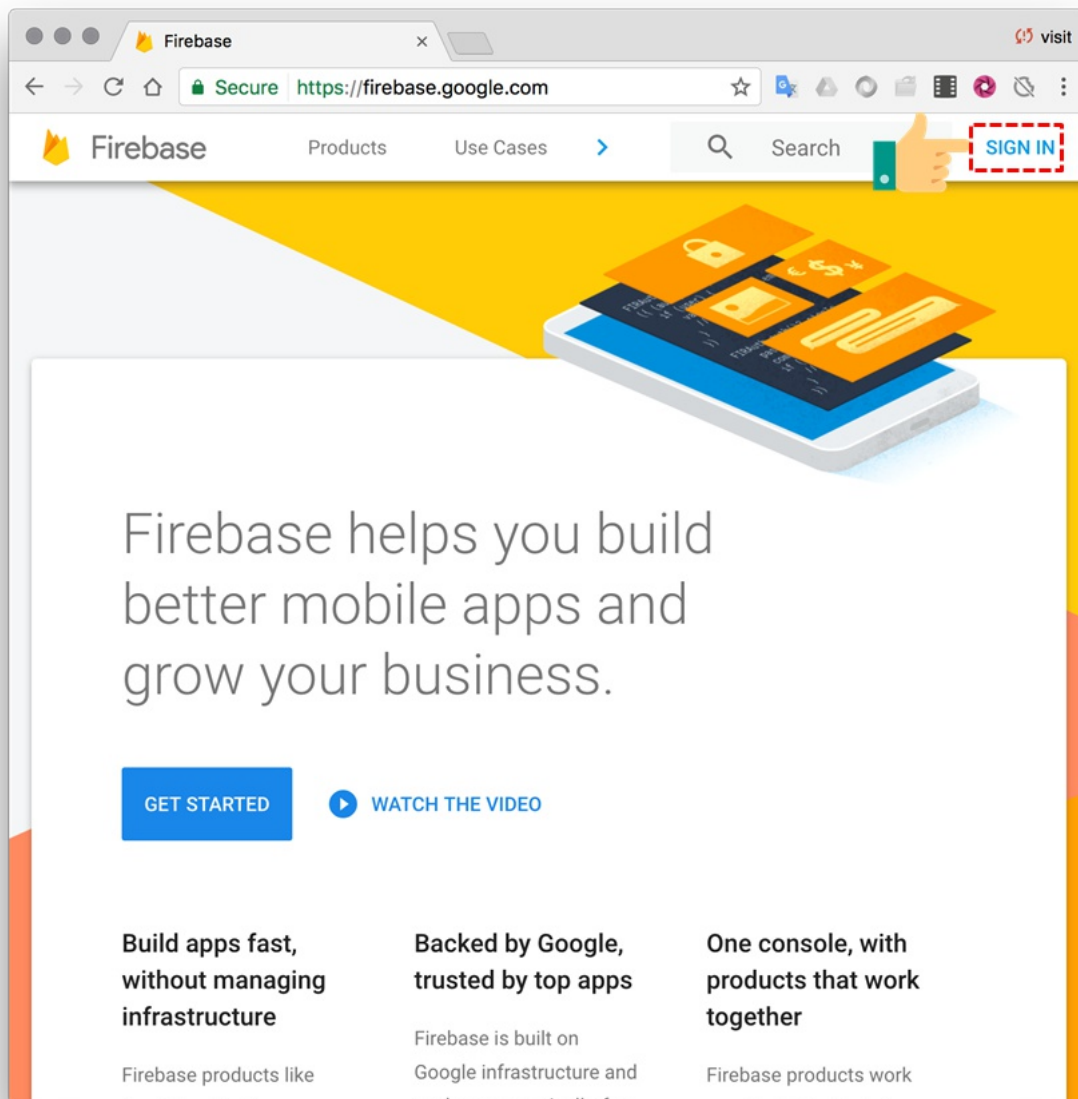
Firestore

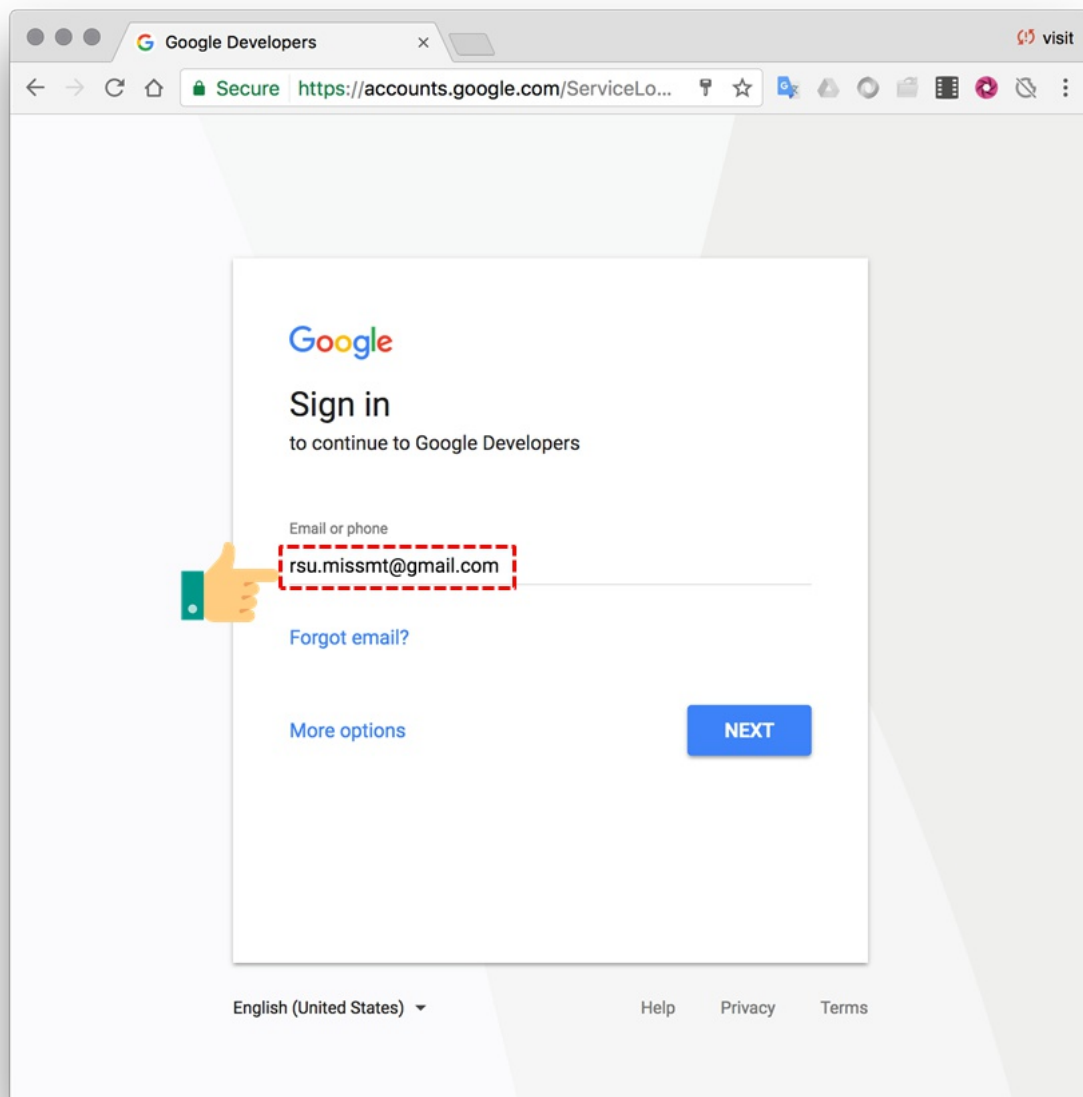
iot realtime database

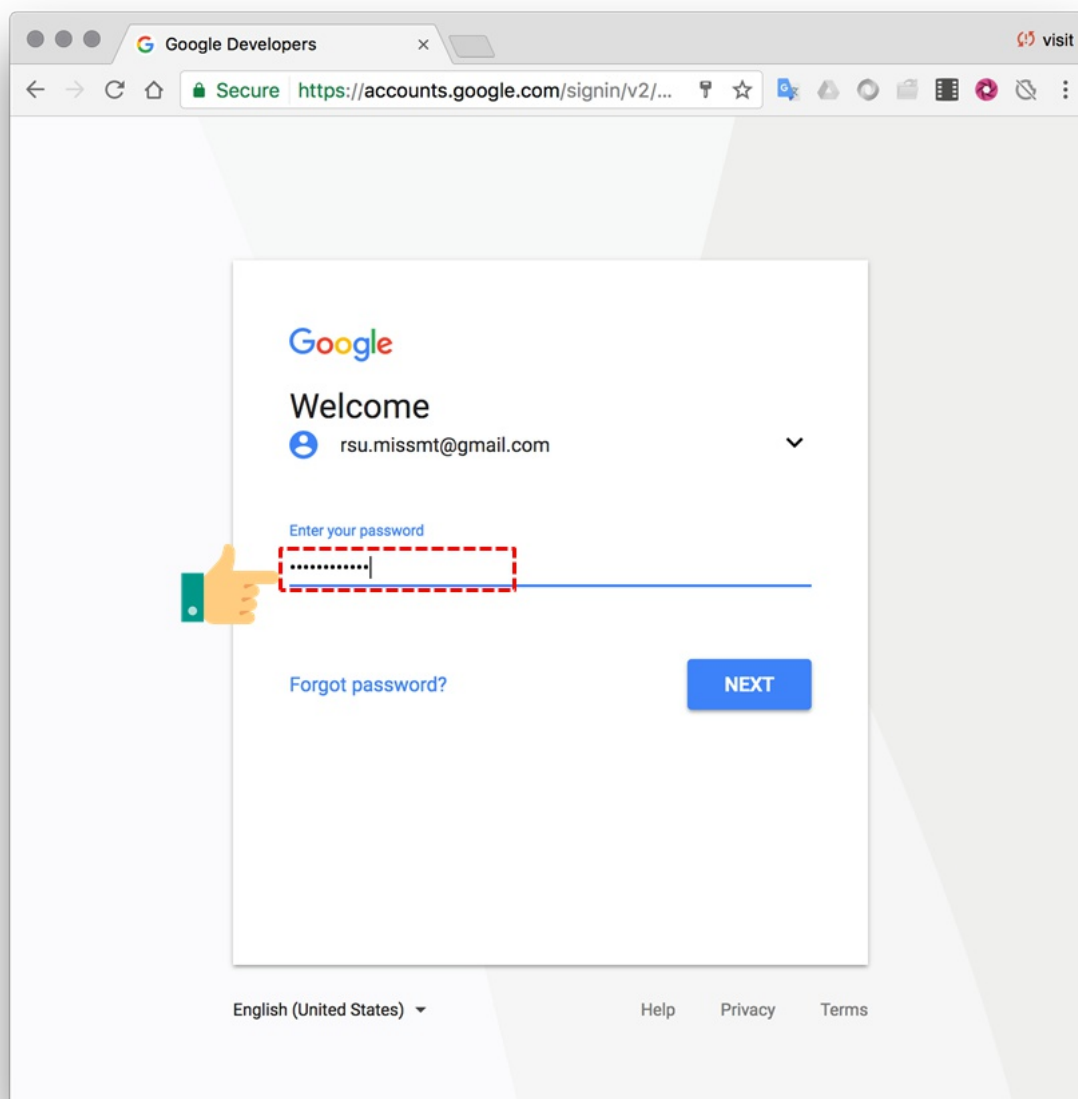
การเชื่อมต่อ ESP32 กับ Firestore

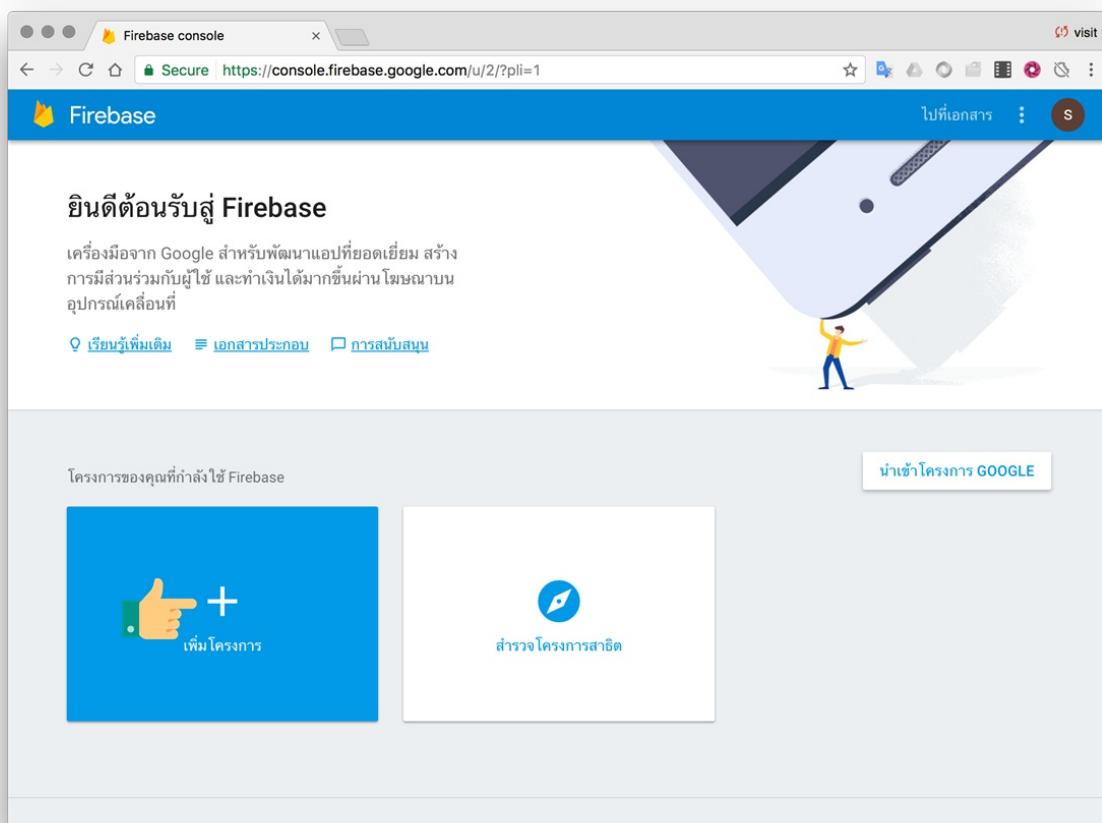
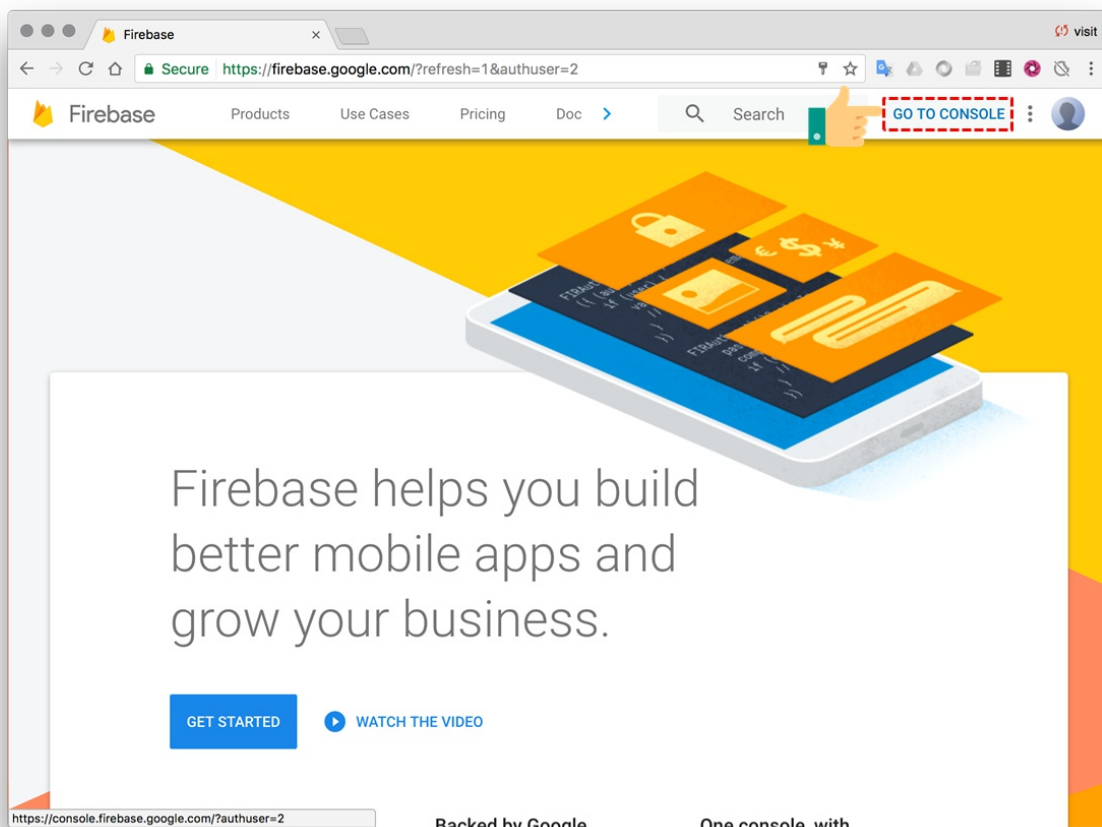


ลงทะเบียนใช้งาน Firebase









เพิ่มโครงการ

ชื่อโครงการ

iot-project

รหัสโครงการ

iot-project-8bb54

ประเทศ/ภูมิภาค

ไทย

☒ ใช้การตั้งค่าเริ่มต้นในการแชร์ข้อมูล Google Analytics สำหรับ Firebase

✓ แชร์ข้อมูล Analytics กับ Google เพื่อช่วยปรับปรุงผลิตภัณฑ์และบริการของ Google

✓ แชร์ข้อมูล Analytics กับ Google เพื่อเปิดใช้การสนับสนุนด้านเทคนิค

✓ แชร์ข้อมูล Analytics กับ Google เพื่อเปิดใช้การเปรียบเทียบ

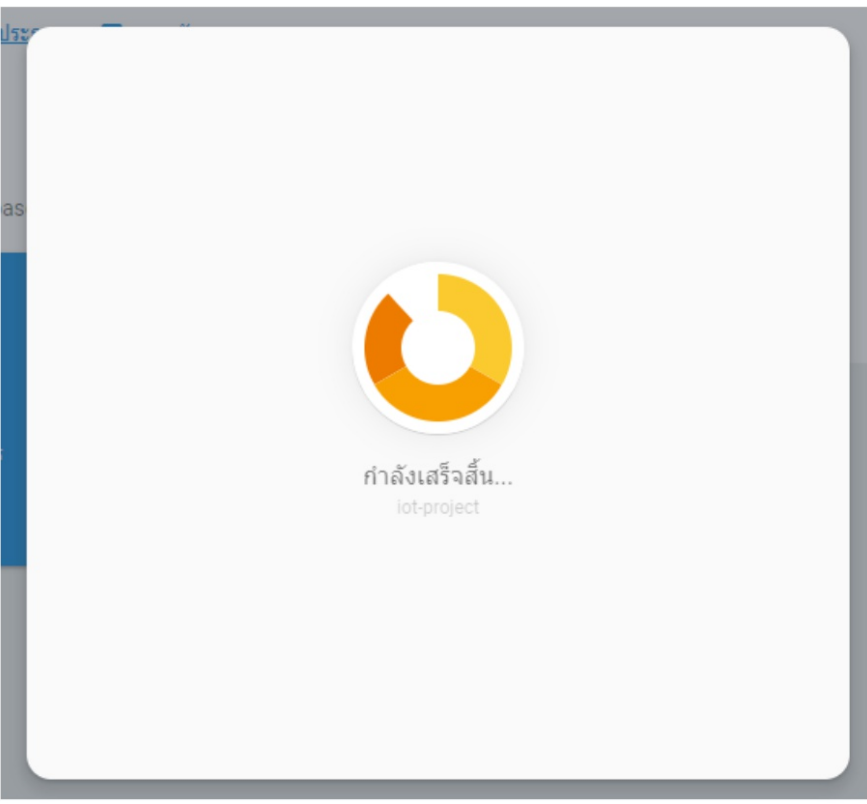
✓ แชร์ข้อมูล Analytics กับผู้เชี่ยวชาญปัญหาคี Google

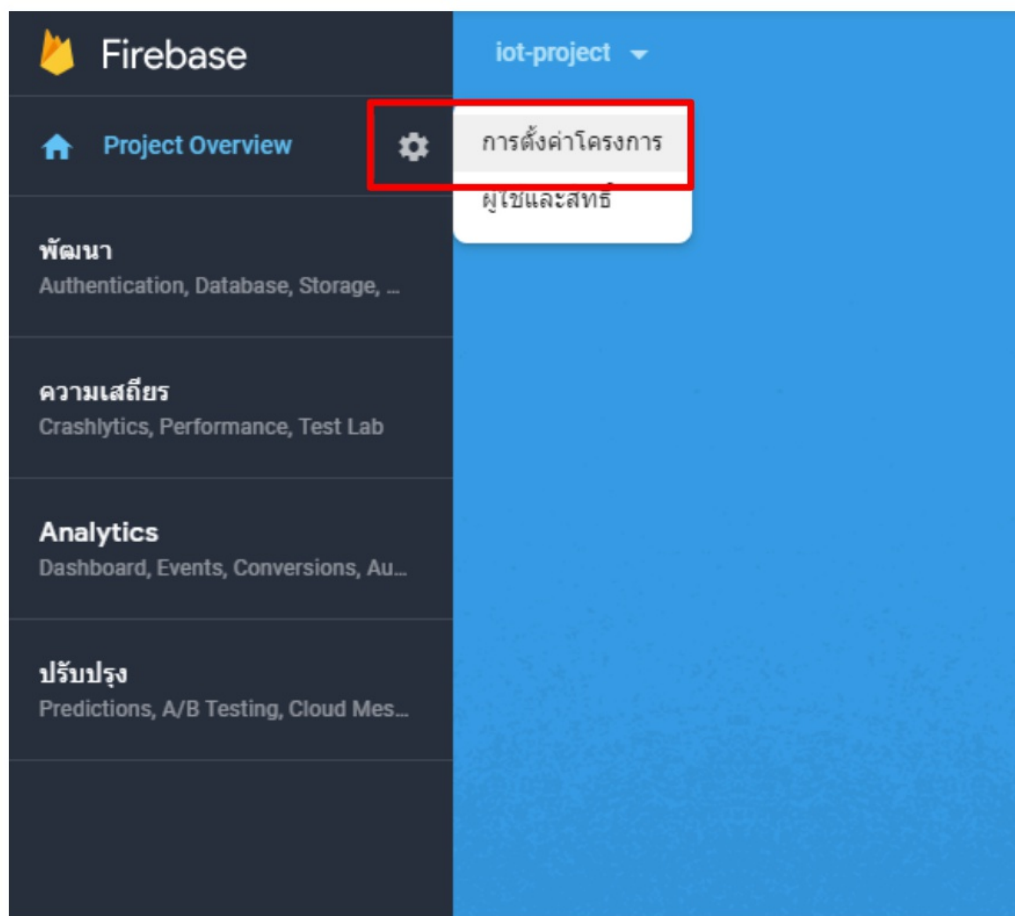
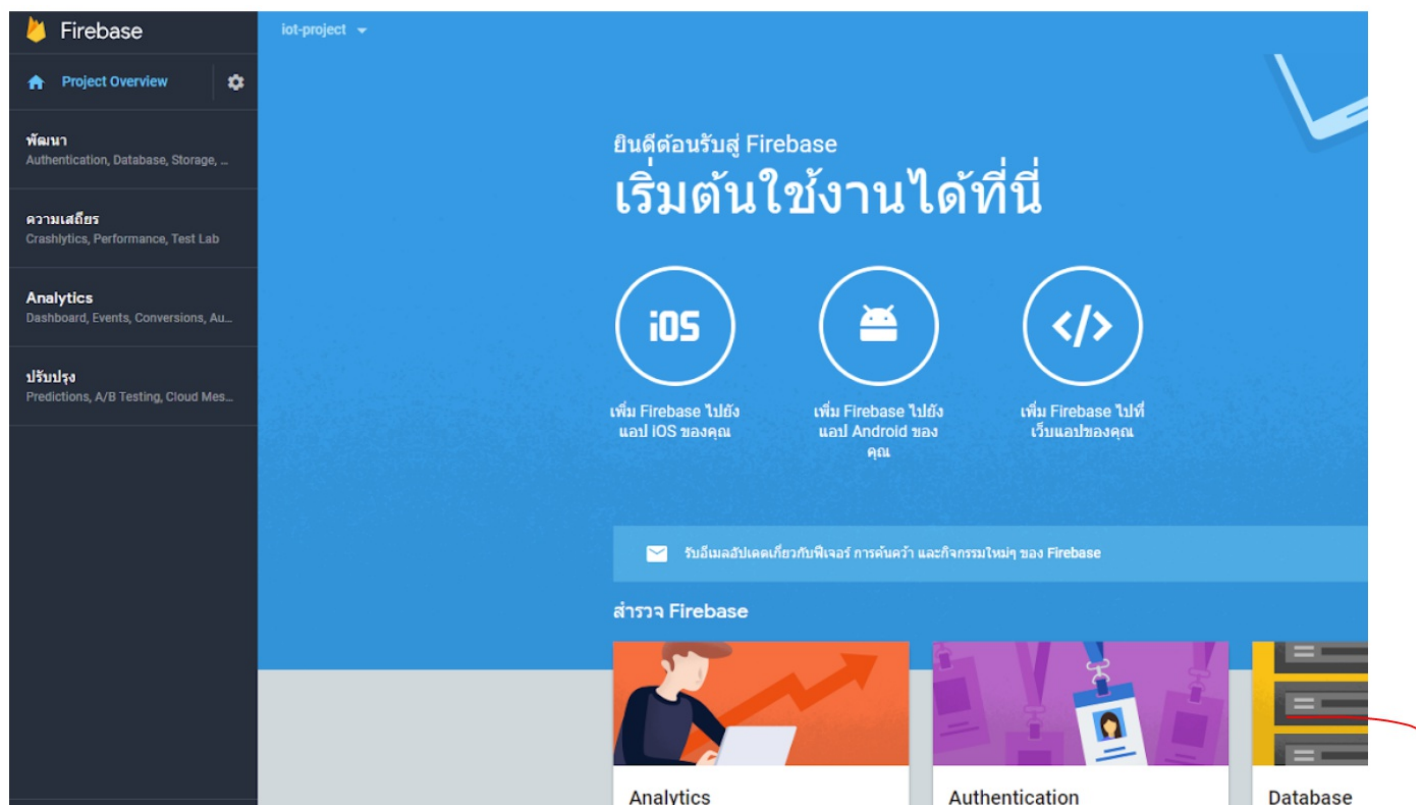
☒ ฉันยอมรับ [ข้อกำหนดด้านการคุ้มครองข้อมูลส่วนตัว](#) โดยจำเป็นต้องมีข้อกำหนดนี้เมื่อแชร์ข้อมูล Analytics เพื่อปรับปรุงผลิตภัณฑ์และบริการของ Google [ดูข้อมูลเพิ่มเติม](#)

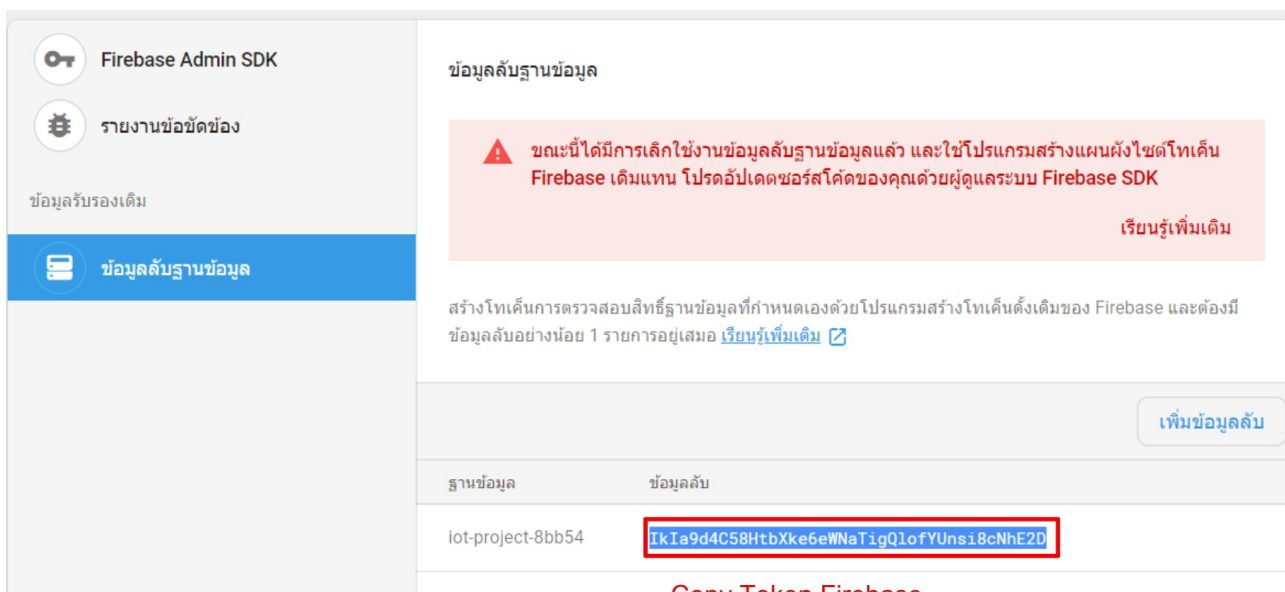
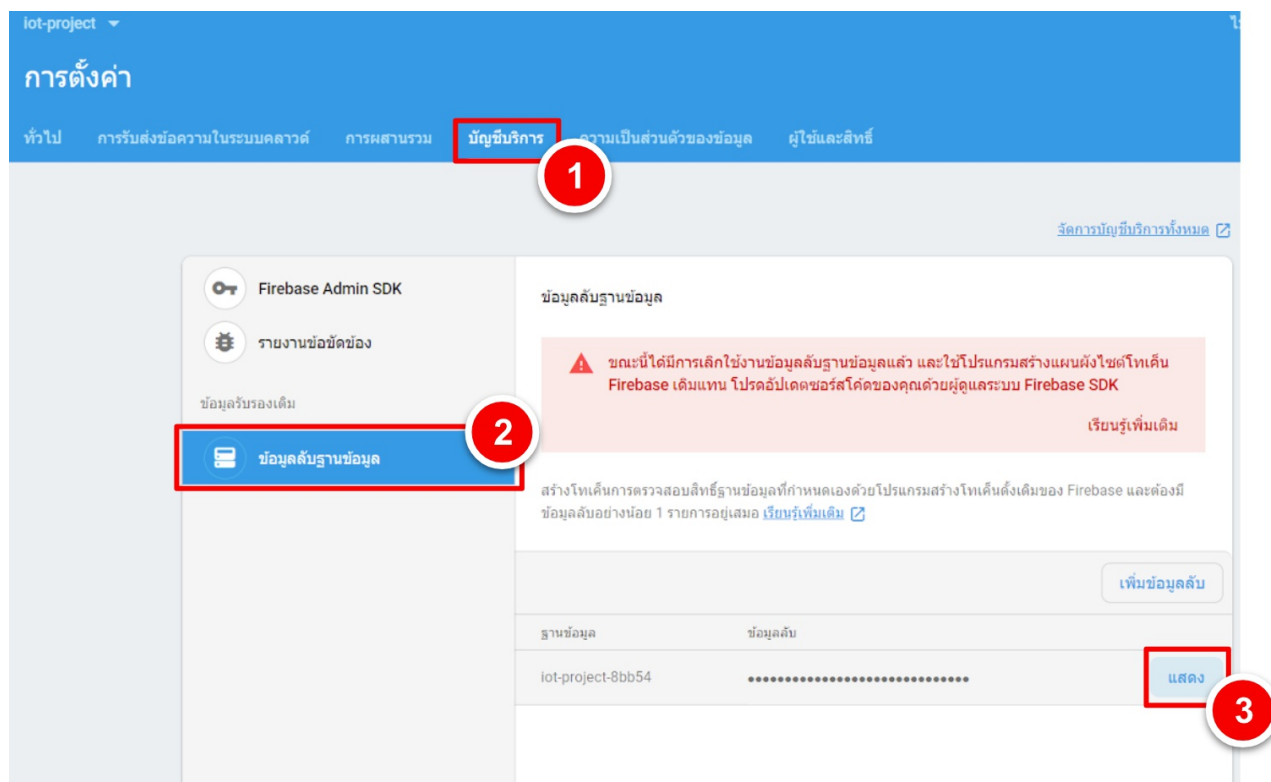
☒ ฉันยอมรับว่าฉันกำลังใช้บริการ Firebase ในแอปของฉัน และยอมรับ [ข้อกำหนดที่เกี่ยวข้อง](#)

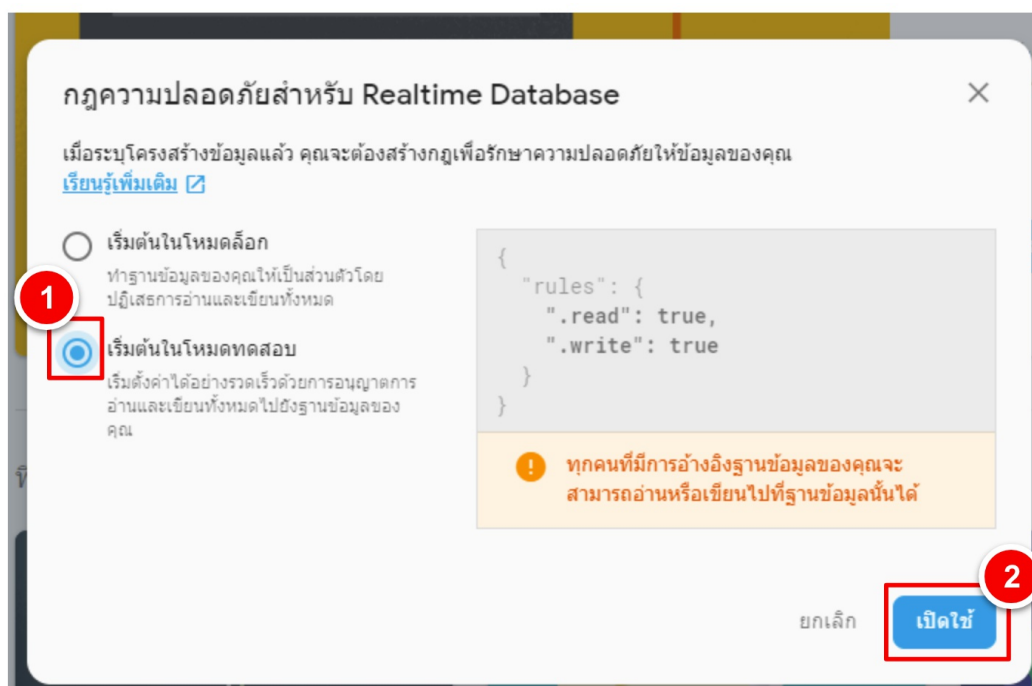
ยกเลิก

สร้างโครงการ









1

2

3

Copy URL Firebase

<https://iot-project-8bb54.firebaseio.com/>

กฎความปลอดภัยได้กำหนดเป็นสาธารณะ ดังนั้นทุกคนจะขโมย แก้ไข หรือลบข้อมูลใน

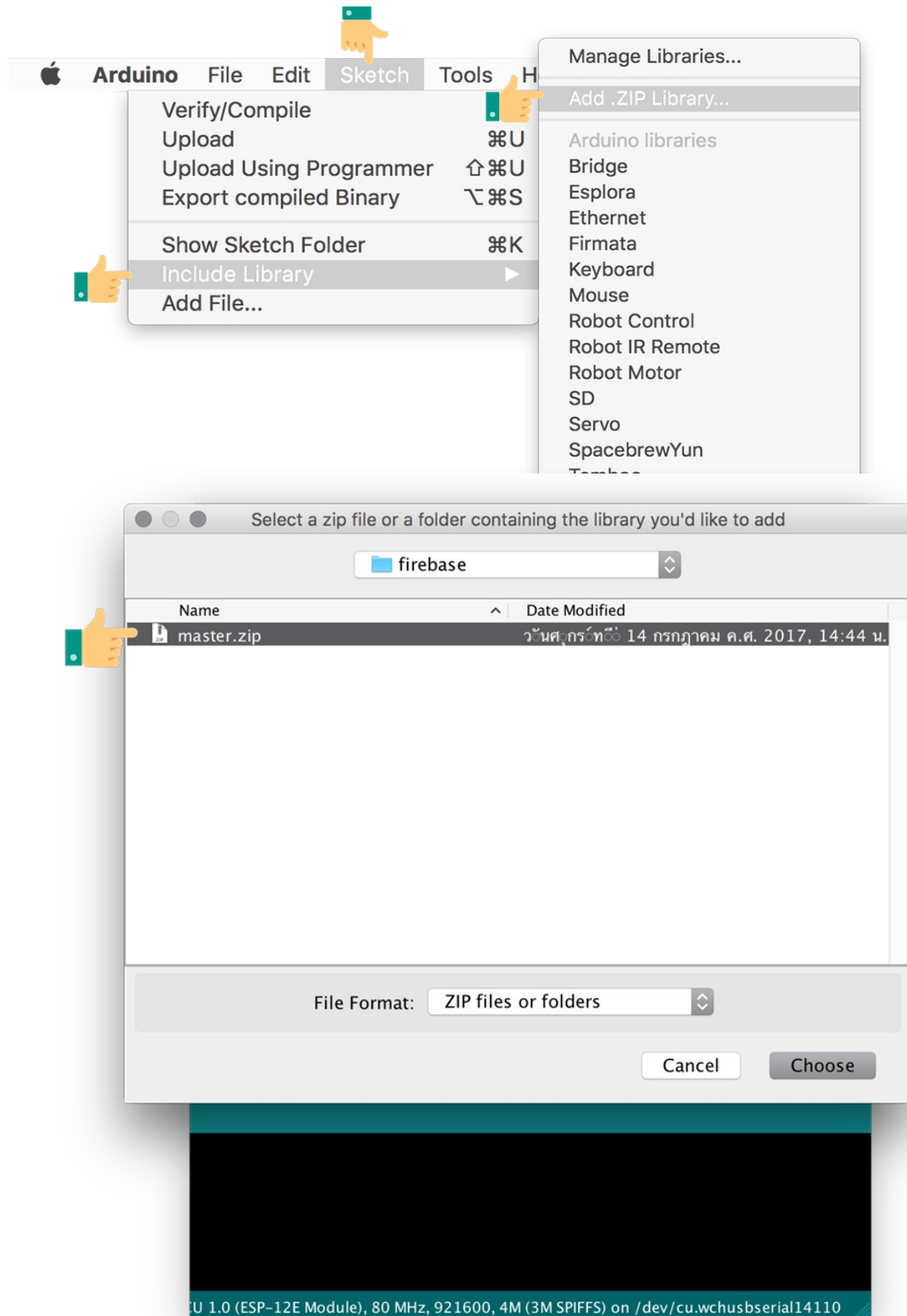
iot-project-8bb54: null

library setup

ดาวน์โหลดไลบรารี Firebase สำหรับ arduino

<https://github.com/googlesamples/firebase-arduino/archive/master.zip>

เพิ่มไลบรารี Firebase ใน arduino IDE



EX18_FIREBASE_COUNTER

ตัวอย่างทดสอบ การเชื่อมต่อ ESP32 เข้ากับฐานข้อมูลออนไลน์ Google Firebase ซึ่งเป็นบริการแบบเรียลไทม์ที่กำลังได้รับความนิยมเป็นอย่างมากในปัจจุบัน

ผลลัพธ์ที่ได้คือ จะแสดงข้อความทักทายบน Firebase และแสดงค่าตัวเลขเพิ่มขึ้นไปที่ละ+1 บนคอนโซลของโปรแกรม arduino และ Firebase แบบเรียลไทม์



```
EX18_FIREBASE_COUNTER | Arduino 1.8.5

EX18_FIREBASE_COUNTER

#include <WiFi.h>
#include <IOXhop_FirebaseESP32.h>

/* ##### */
//Firebase config here
#define FIREBASE_HOST "smtled-f6bc8.firebaseio.com"
#define FIREBASE_AUTH "u7j7XhFN57gnDpmIfUrE5bj9tiy4RVKm3bnNHAQc"
#define WIFI_SSID "plutter"
#define WIFI_PASSWORD "visit012"
/* ##### */

int i = 0;
void setup() {

  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  Serial.println();
  Serial.print("connected: ");

  Done Saving.
  Leaving...
  Hard resetting...

55 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART
```

EX18 แสดงเลข counter บน Firebase แบบเรียลไทม์

The image shows two side-by-side windows. The left window is a serial monitor titled "/dev/cu.SLAB_USBtoUART" with a "Send" button. It displays a list of commands: "set /CounterNumber to 8" through "set /CounterNumber to 35". The last command, "set /CounterNumber to 35", is highlighted with a red dashed box. The right window is the Firebase Realtime Database console for the project "smtled-f6bc8". It shows the database structure with a "CounterNumber" field containing the value "35" (highlighted with a red dashed box) and a "message" field containing the text "Welcome to IoT training." The URL bar shows "https://smtled-f6bc8.firebaseio.com/".

Serial Monitor Output:

```
set /CounterNumber to 8
set /CounterNumber to 9
set /CounterNumber to 10
set /CounterNumber to 11
set /CounterNumber to 12
set /CounterNumber to 13
set /CounterNumber to 14
set /CounterNumber to 15
set /CounterNumber to 16
set /CounterNumber to 17
set /CounterNumber to 18
set /CounterNumber to 19
set /CounterNumber to 20
set /CounterNumber to 21
set /CounterNumber to 22
set /CounterNumber to 23
set /CounterNumber to 24
set /CounterNumber to 25
set /CounterNumber to 26
set /CounterNumber to 27
set /CounterNumber to 28
set /CounterNumber to 29
set /CounterNumber to 30
set /CounterNumber to 31
set /CounterNumber to 32
set /CounterNumber to 33
set /CounterNumber to 34
set /CounterNumber to 35
```

Firebase Realtime Database Console:

Database: smtLED

URL: https://smtled-f6bc8.firebaseio.com/

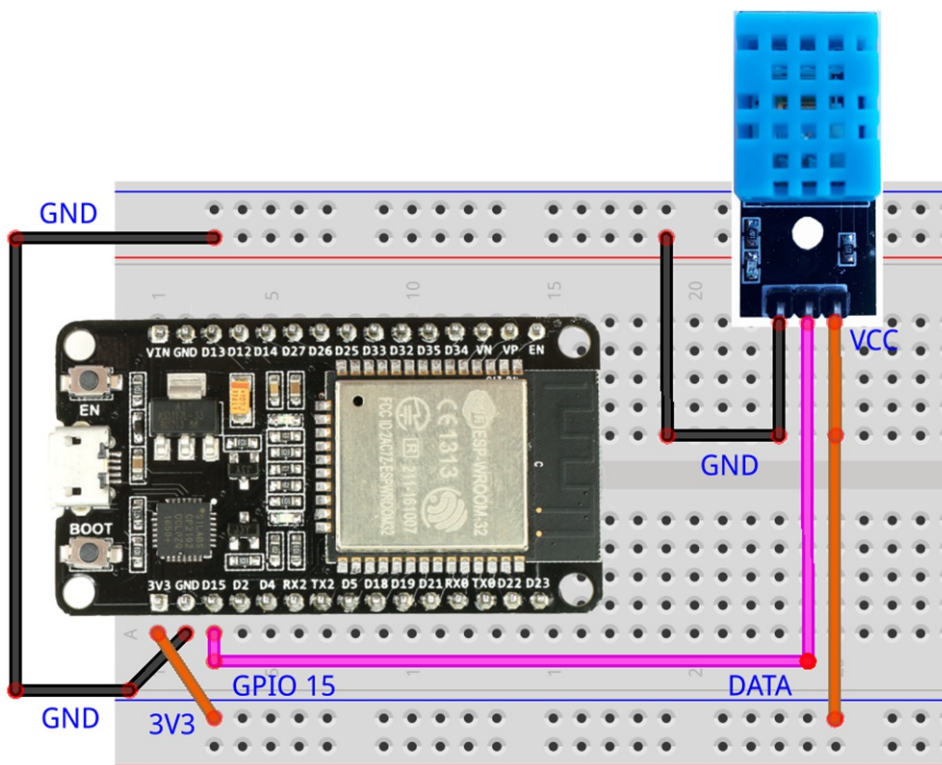
Database Structure:

```
{
  CounterNumber: 35
  message: "Welcome to IoT training."
}
```

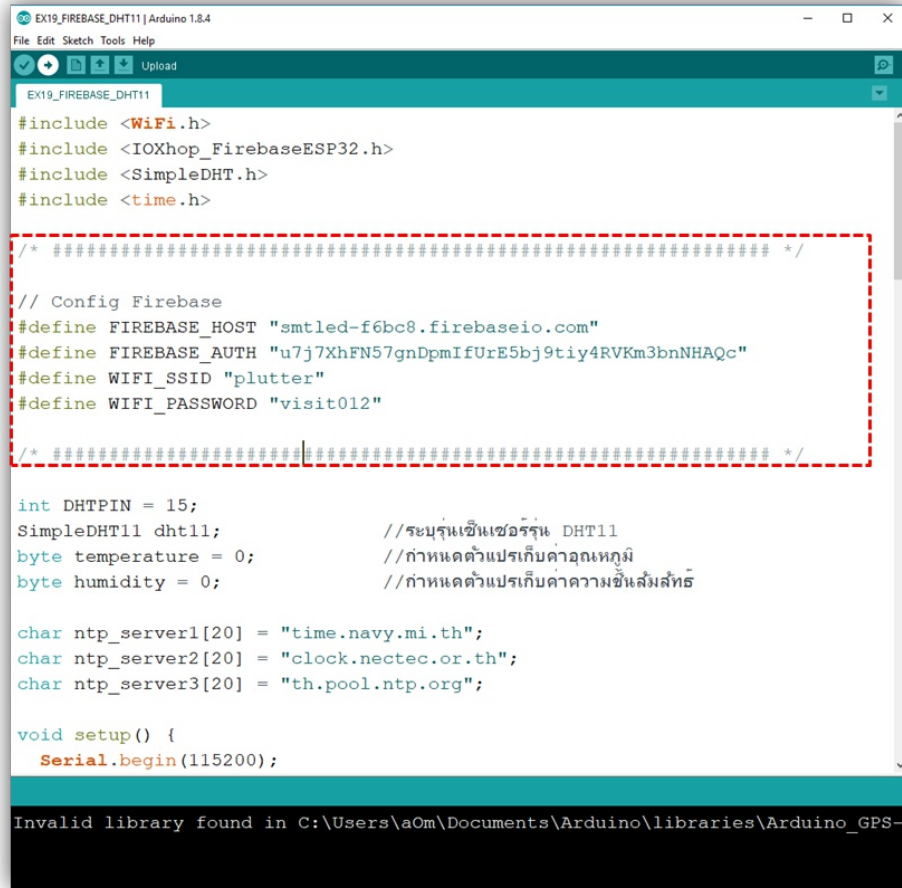
EX19_FIREBASE_DHT11

ตัวอย่างทดสอบ การเชื่อมต่อ ESP32 เข้ากับฐานข้อมูลออนไลน์ Google Firebase ซึ่งเป็นบริการแบบเรียลไทม์ที่กำลังได้รับความนิยมเป็นอย่างมากในปัจจุบัน

ผลลัพธ์ที่ได้คือ จะส่งค่าอุณหภูมิและความชื้นไปเก็บบนฐานข้อมูลออนไลน์ Firebase และแสดงผลบน dashboard ในไฟล์ html แบบเรียลไทม์ สามารถดูได้ผ่านโปรแกรม web browser ทั่วไปได้เช่น google chrome



EX19 แสดงข้อมูลอุณหภูมิและความชื้นบน Firebase



```

EX19_FIREBASE_DHT11 | Arduino 1.8.4
File Edit Sketch Tools Help

EX19_FIREBASE_DHT11

#include <WiFi.h>
#include <IOXhop_FirebaseESP32.h>
#include <SimpleDHT.h>
#include <time.h>

/* ##### */

// Config Firebase
#define FIREBASE_HOST "smtled-f6bc8.firebaseio.com"
#define FIREBASE_AUTH "u7j7XhFN57gnDpmIfUrE5bj9tiy4RVKm3bnNHAQc"
#define WIFI_SSID "plutter"
#define WIFI_PASSWORD "visit012"

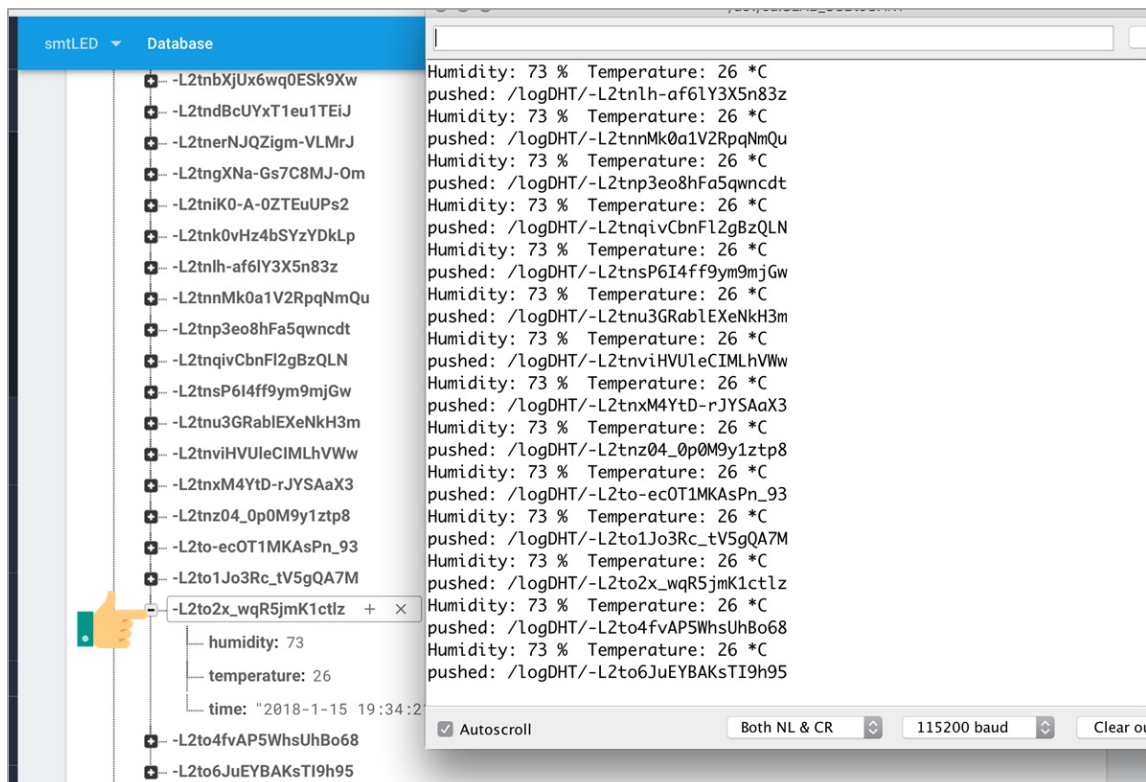
/* ##### */

int DHTPIN = 15;
SimpleDHT11 dht11;           // ระบุพินเซ็นเซอร์ DHT11
byte temperature = 0;        // กำหนดตัวแปรเก็บค่าอุณหภูมิ
byte humidity = 0;           // กำหนดตัวแปรเก็บค่าความชื้นสัมพัทธ์

char ntp_server1[20] = "time.navy.mi.th";
char ntp_server2[20] = "clock.nectec.or.th";
char ntp_server3[20] = "th.pool.ntp.org";

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

  Invalid library found in C:\Users\Aom\Documents\Arduino\libraries\Arduino_GPS-
  
```



Database

- L2tnbXjUx6wq0ESk9Xw
- L2tndBcUYxT1eu1TEiJ
- L2tnerNJQZigm-VLMrJ
- L2tngXNa-Gs7C8MJ-Om
- L2tniK0-A-0ZTEuUPs2
- L2tnk0vHz4bSYzYDkLp
- L2tnlh-af6lY3X5n83z
- L2tnnMk0a1V2RpqNmQu
- L2tnp3eo8hFa5qwncdt
- L2tnqivCbnFl2gBzQLN
- L2tnsP6I4ff9ym9mjGw
- L2tnu3GRablEXeNkH3m
- L2tnviHVUleCIMLhVWw
- L2tnxM4YtD-rJYSAaX3
- L2tnz04_0p0M9y1ztp8
- L2to-ecOT1MKAsPn_93
- L2to1Jo3Rc_tV5gQA7M
- L2to2x_wqR5jmK1ctlz
- L2to4fvAP5WhsUhBo68
- L2to6JuEYBAKsTI9h95

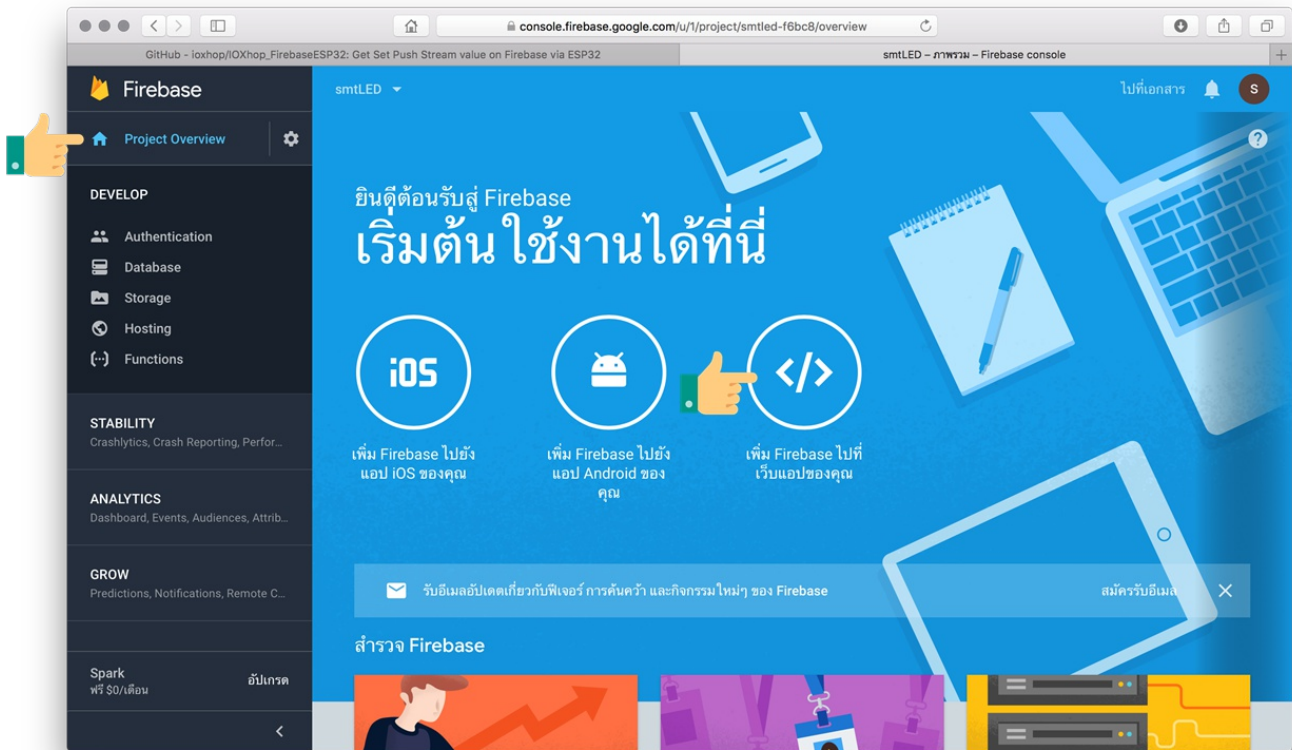
Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnlh-af6lY3X5n83z
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnnMk0a1V2RpqNmQu
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnp3eo8hFa5qwncdt
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnqivCbnFl2gBzQLN
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnsP6I4ff9ym9mjGw
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnu3GRablEXeNkH3m
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnviHVUleCIMLhVWw
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnxM4YtD-rJYSAaX3
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2tnz04_0p0M9y1ztp8
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2to-ecOT1MKAsPn_93
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2to1Jo3Rc_tV5gQA7M
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2to2x_wqR5jmK1ctlz
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2to4fvAP5WhsUhBo68
 Humidity: 73 % Temperature: 26 *C
 pushed: /logDHT/-L2to6JuEYBAKsTI9h95

humidity: 73
 temperature: 26
 time: "2018-11-15 19:34:2"

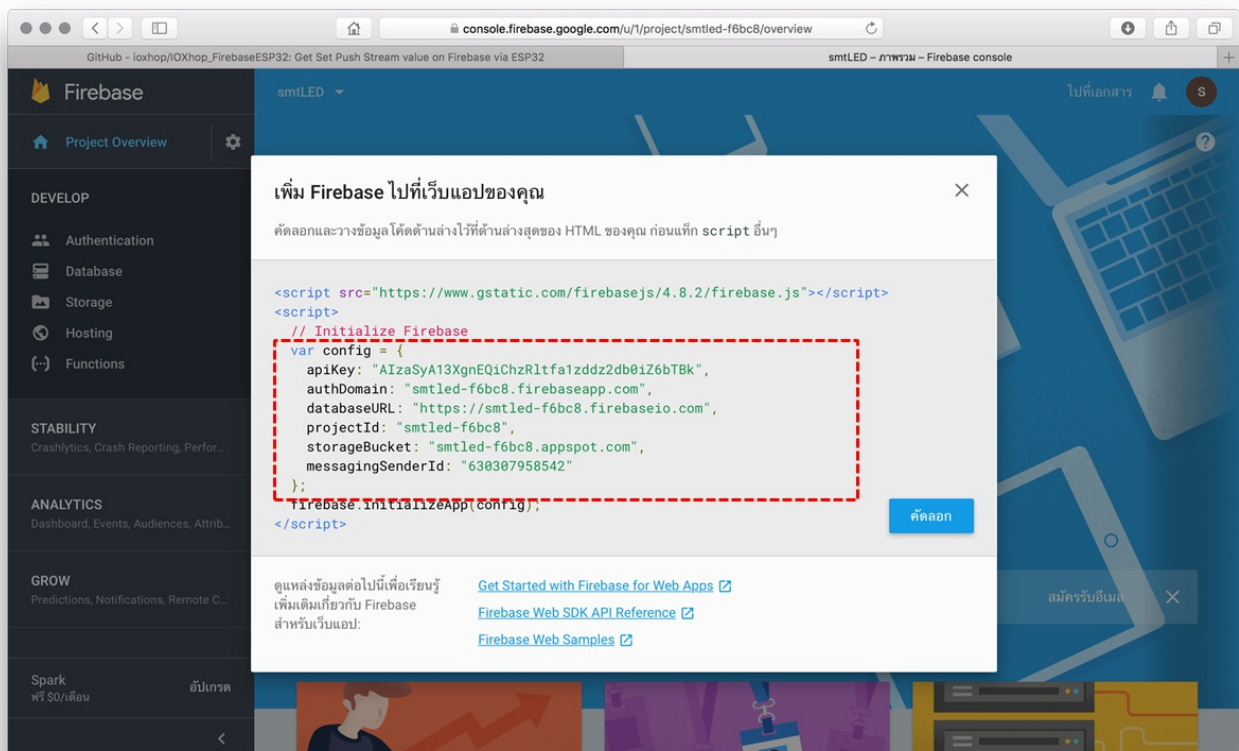
Autoscroll Both NL & CR 115200 baud Clear

EX19 แสดงข้อมูลอุณหภูมิและความชื้นบน Firebase

ดูรายละเอียด API เพื่อนำมาสร้าง dashboard

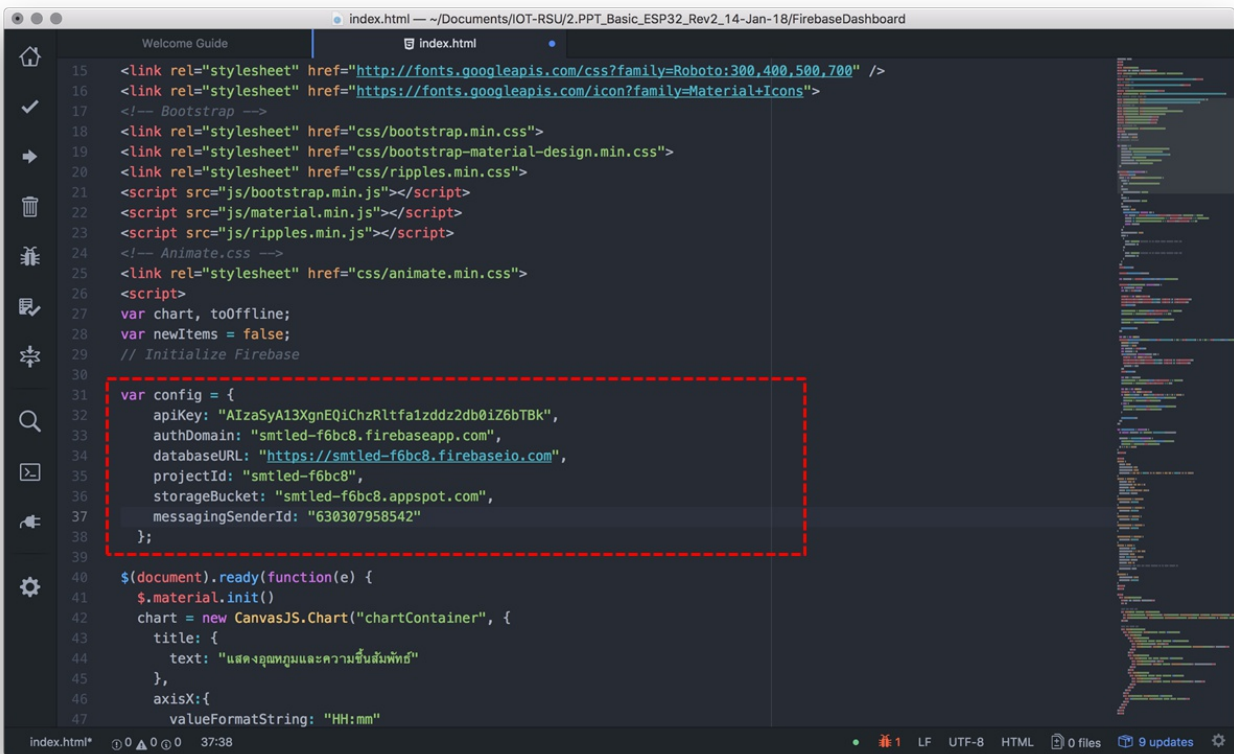


คัดลอกโค้ดจาก Firebase



EX19 แสดงข้อมูลอุณหภูมิและความชื้นบน Firebase

นำโค้ดที่ได้จาก Firebase มาเขียนที่นี่

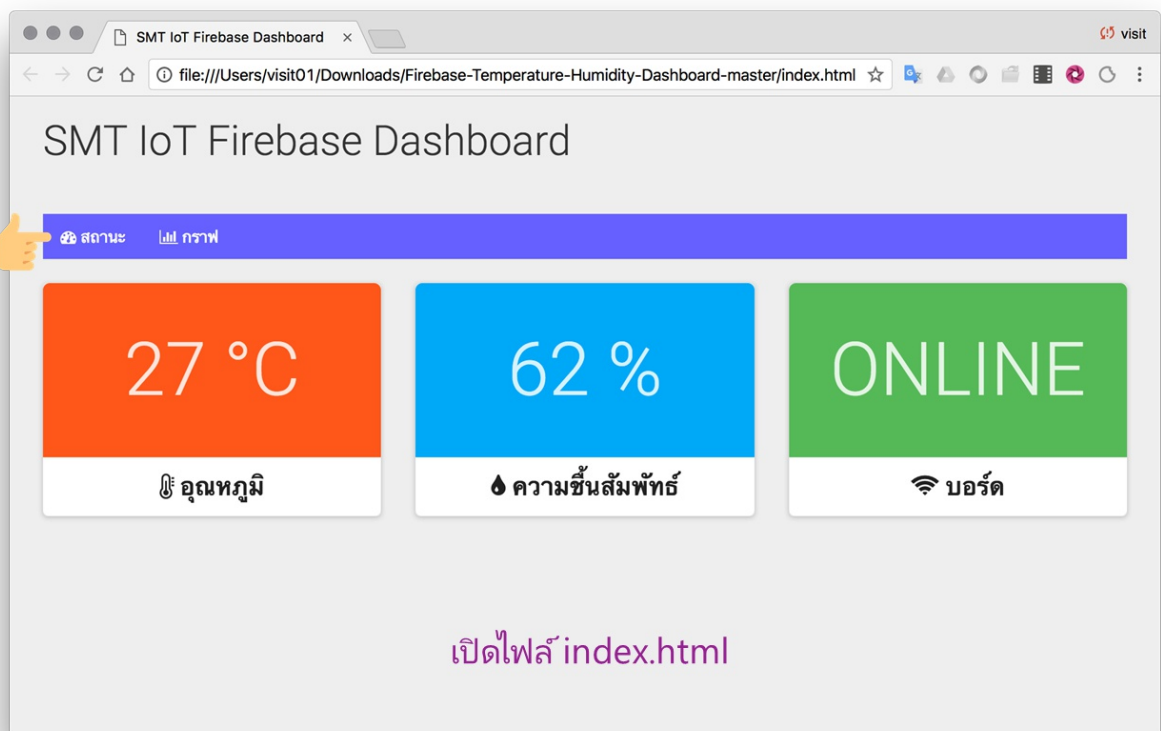


```

15 <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto:300,400,500,700" />
16 <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
17 <!-- Bootstrap -->
18 <link rel="stylesheet" href="css/bootstrap.min.css">
19 <link rel="stylesheet" href="css/bootstrap-material-design.min.css">
20 <link rel="stylesheet" href="css/ripples.min.css">
21 <script src="js/bootstrap.min.js"></script>
22 <script src="js/material.min.js"></script>
23 <script src="js/ripples.min.js"></script>
24 <!-- Animate.css -->
25 <link rel="stylesheet" href="css/animate.min.css">
26 <script>
27   var chart, toOffline;
28   var newItems = false;
29   // Initialize Firebase
30
31   var config = {
32     apiKey: "AIzaSyA13XgnEQiChzRltfa1zddz2db0iZ6bTBk",
33     authDomain: "smtled-f6bc8.firebaseio.com",
34     databaseURL: "https://smtled-f6bc8.firebaseio.com",
35     projectId: "smtled-f6bc8",
36     storageBucket: "smtled-f6bc8.appspot.com",
37     messagingSenderId: "630307958542"
38   };
39
40   $(document).ready(function(e) {
41     $.material.init()
42     chart = new CanvasJS.Chart("chartContainer", {
43       title: {
44         text: "แสดงอุณหภูมิและความชื้นสัมพัทธ์"
45       },
46       axisX: {
47         valueFormatString: "HH:mm"
48       }
49     });
50   });

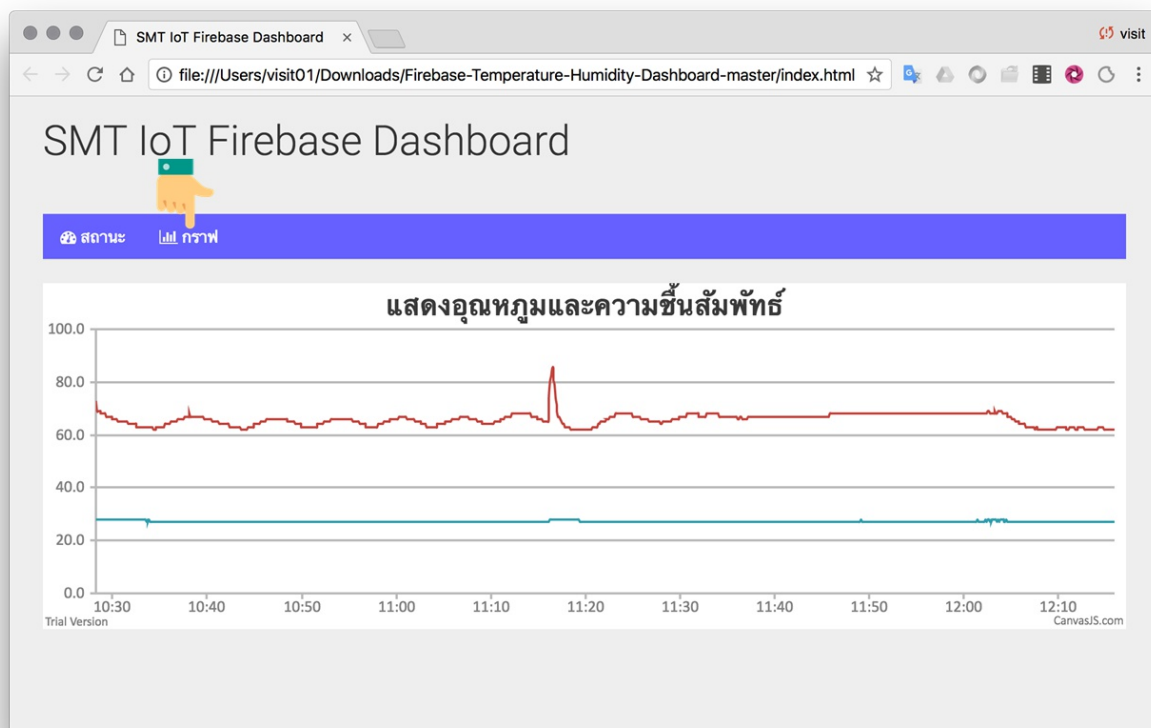
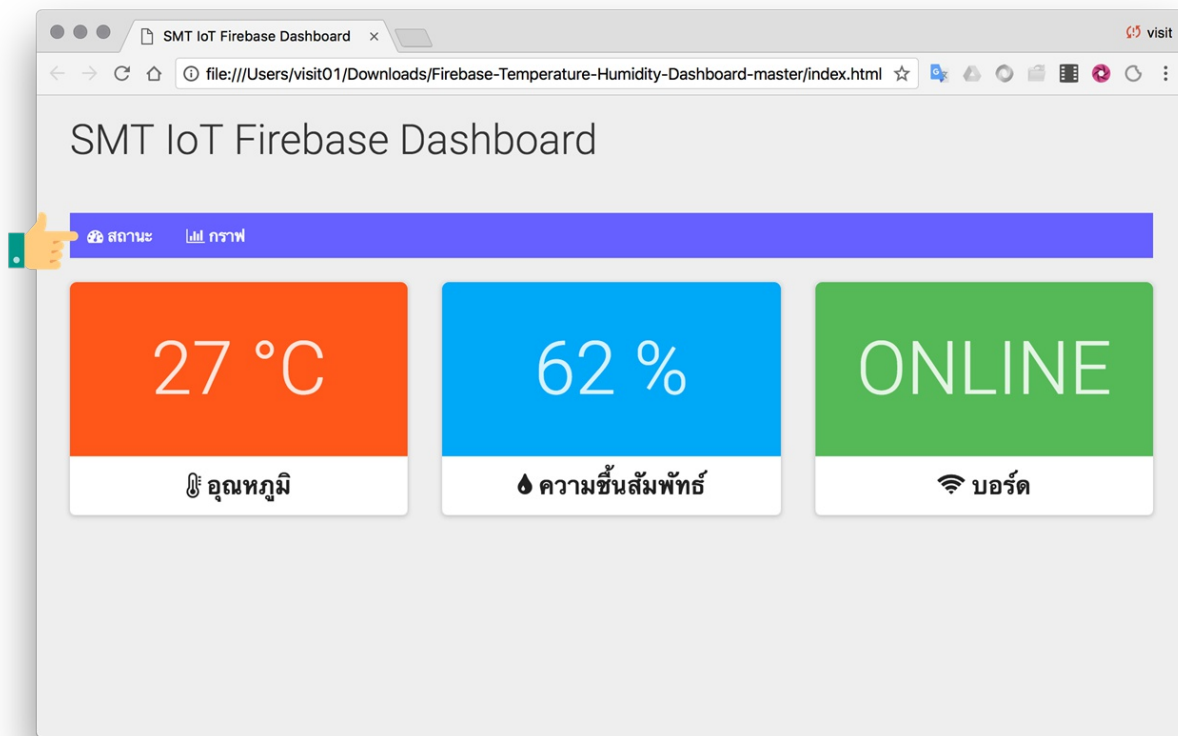
```

ไฟล์ประกอบ FirebaseDashboard.zip (index.html)



EX19 แสดงข้อมูลอุณหภูมิและความชื้นบน Firebase

เปิดไฟล์ index.html



EX20_FIREBASE_LED_MULTITASK

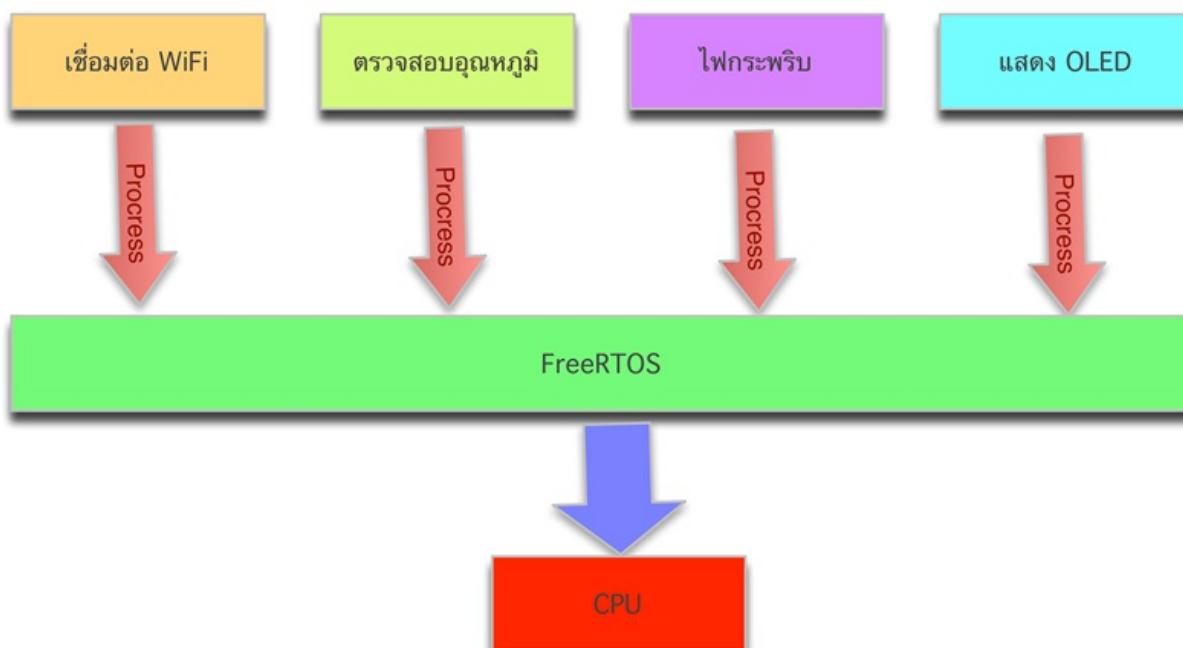
ตัวอย่างทดสอบ การเชื่อมต่อ ESP32 เข้ากับฐานข้อมูลออนไลน์ Google Firebase ซึ่งเป็นบริการแบบเรียลไทม์ที่กำลังได้รับความนิยมเป็นอย่างมากในปัจจุบัน

ผลลัพธ์ที่ได้คือ สามารถควบคุมเปิด ปิดหลอดไฟ LED โดยการกดปุ่มสวิทช์ที่ต่อไว้ สถานะของหลอดไฟ LED จะถูกส่งไปอัปเดตที่ฐานข้อมูล Google Firebase แบบเรียลไทม์ นอกจากนี้ยังสามารถเปลี่ยนสถานะ “LED1” ที่หน้าคอนโซลของ Firebase เพื่อควบคุมหลอดไฟ LED1 ได้อีกด้วย

ตัวอย่างนี้จะสาธิตใช้วิธีการทำงาน CPU แบบ Multitask แท้ ซึ่ง ESP32 สามารถจัดการทำงานได้อย่างดีเยี่ยมผ่าน FreeRTOS

การทำงานแบบมัลติทาสกิ้ง

มัลติทาสกิ้ง (multitasking) ใน ESP32 คือ การทำงานหลายส่วนขนานกันไป อาทิเช่นการตรวจสอบสถานะของแสงเพื่อเปิดปิดไฟในขณะที่ต้องตรวจสอบสถานะการกดปุ่มสวิทช์ไปด้วยในเวลาเดียวกัน หรือการทำไฟกระพริบเป็นจังหวะเพื่อแสดงให้เห็นว่าอุปกรณ์กำลังทำงานในขณะที่ต้องคอยตรวจสอบข้อมูลที่มาจากเซนเซอร์วัดอุณหภูมิ ความชื้น เป็นต้น



MultiTasking FreeRTOS

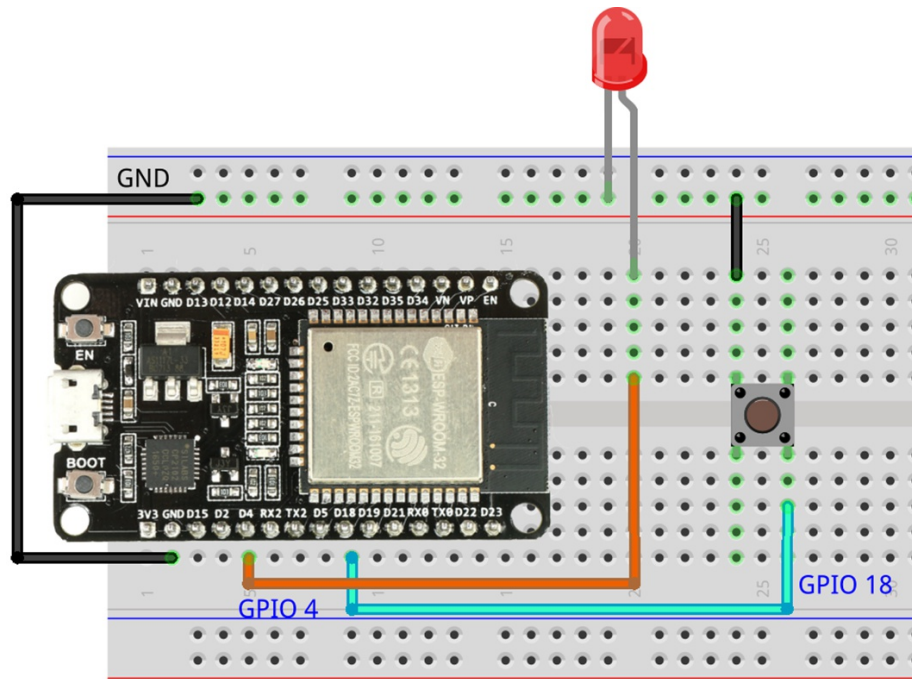
เป็นการทำงานแบบหลายกระบวนการในเวลาเดียวกัน และยังสามารถกำหนดระดับความสำคัญ (Priority) ได้ ความสามารถนี้ ESP32 จะใช้ระบบ FreeRTOS เข้ามาช่วยจัดการ โดยทำการการแบ่งแยกคอร์ของ MCU ควบคุมการทำงานของแต่ละ task เรียกว่าการทำงานในลักษณะนี้ว่ามัลติทาสกิ้ง โดยแต่ละ Task จะแยกการทำงานออกจากกัน อย่างอิสระ

ฟังก์ชัน xTaskCreate() สำหรับการสร้าง task

/* Create the task, storing the handle. */

```
xReturned = xTaskCreate(
    vTaskCode,          /* Function that implements the task. */
    "NAME",              /* Text name for the task. */
    STACK_SIZE,         /* Stack size in words, not bytes. */
    ( void * ) 1,        /* Parameter passed into the task. */
    tskIDLE_PRIORITY,    /* Priority at which the task is created. */
    &xHandle ); /* Used to pass out the created task's handle. */
```

EX20 ควบคุมหลอดไฟ LED บน Firebase



```

TESTFIREBASE | Arduino 1.8.5

TESTFIREBASE
#include <WiFi.h>
#include <IOXhop_FirebaseESP32.h>

/* ##### */
// Config Firebase
#define FIREBASE_HOST "smtled-f6bc8.firebaseio.com"
#define FIREBASE_AUTH "u7j7XhFN57gnDpmIfUrE5bj9tiy4RVKm3bnNHAQc"
#define WIFI_SSID "plutter"
#define WIFI_PASSWORD "visit012"
/* ##### */

int LEDPIN = 4; //GPIO4 LED
int SWPIN = 18; //GPIO15 (ADC2_CH3)
boolean state = 0; //กำหนดค่าตัวแปร state เท่ากับ 0 เป็นค่าเริ่มต้น (สถานะปิด)
boolean oldState; //กำหนดค่าตัวแปรเป็นตัวเลข 0 กับ 1
boolean data; //กำหนดค่าตัวแปรเป็นตัวเลข 0 กับ 1

//-----//
void Task1(void *p){

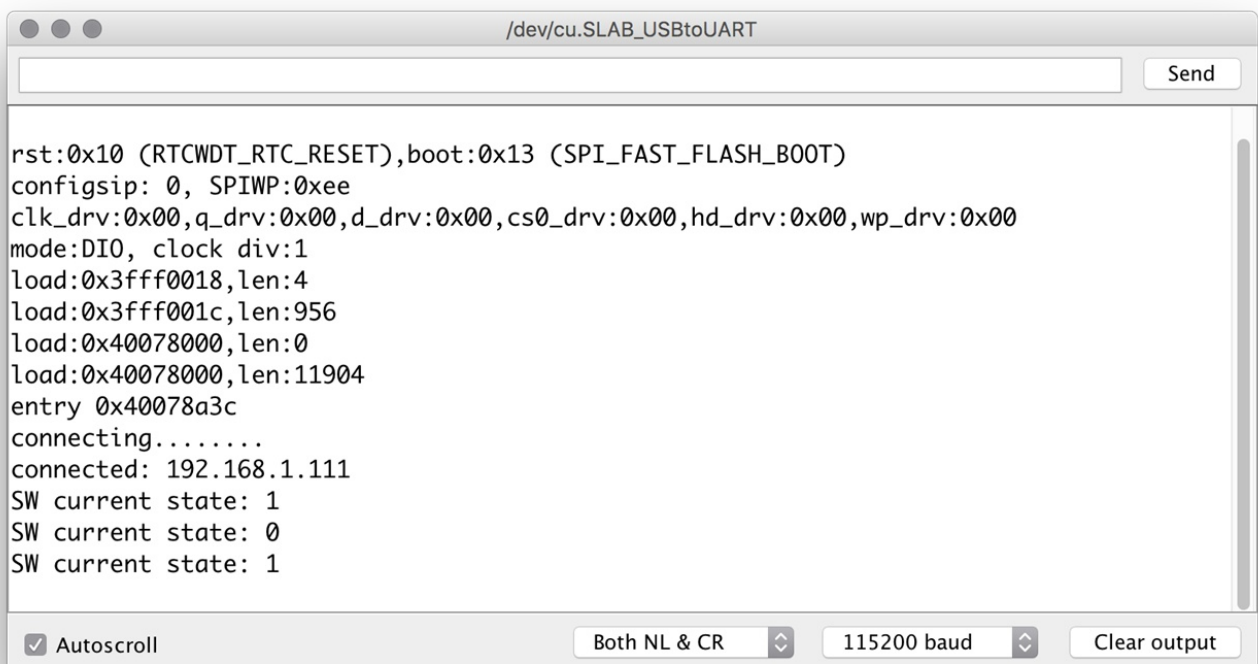
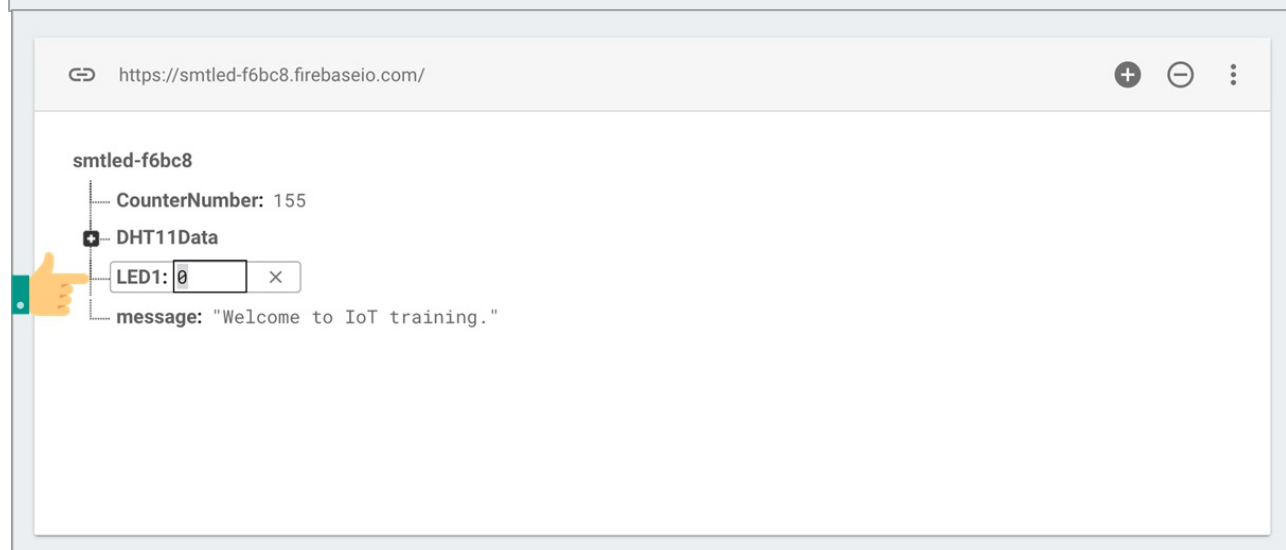
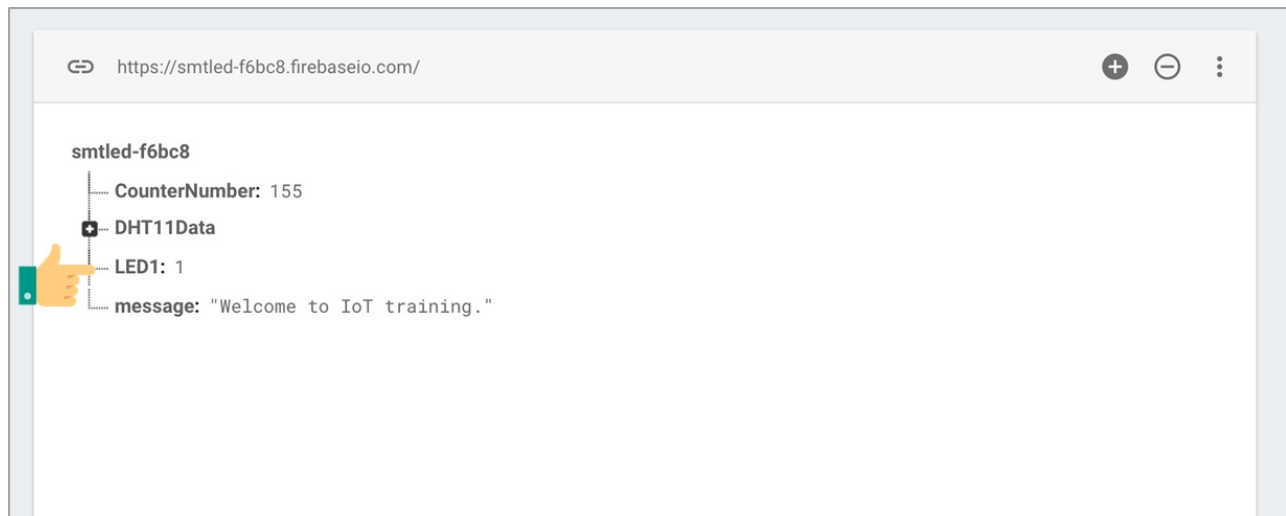
  pinMode(LEDPIN, OUTPUT);
  while(1) {
    digitalWrite(LEDPIN, Firebase.getInt("LED1"));
    vTaskDelay(500 / portTICK_PERIOD_MS);
  }
}

Done Saving.
Leaving...
Hard resetting...

18 ESP32 Dev Module, QIO, 80MHz, 4MB (32Mb), 921600, None on /dev/cu.SLAB_USBtoUART

```


EX20 ควบคุมหลอดไฟ LED บน Firebase



EX21 SMART FARM PROJECT

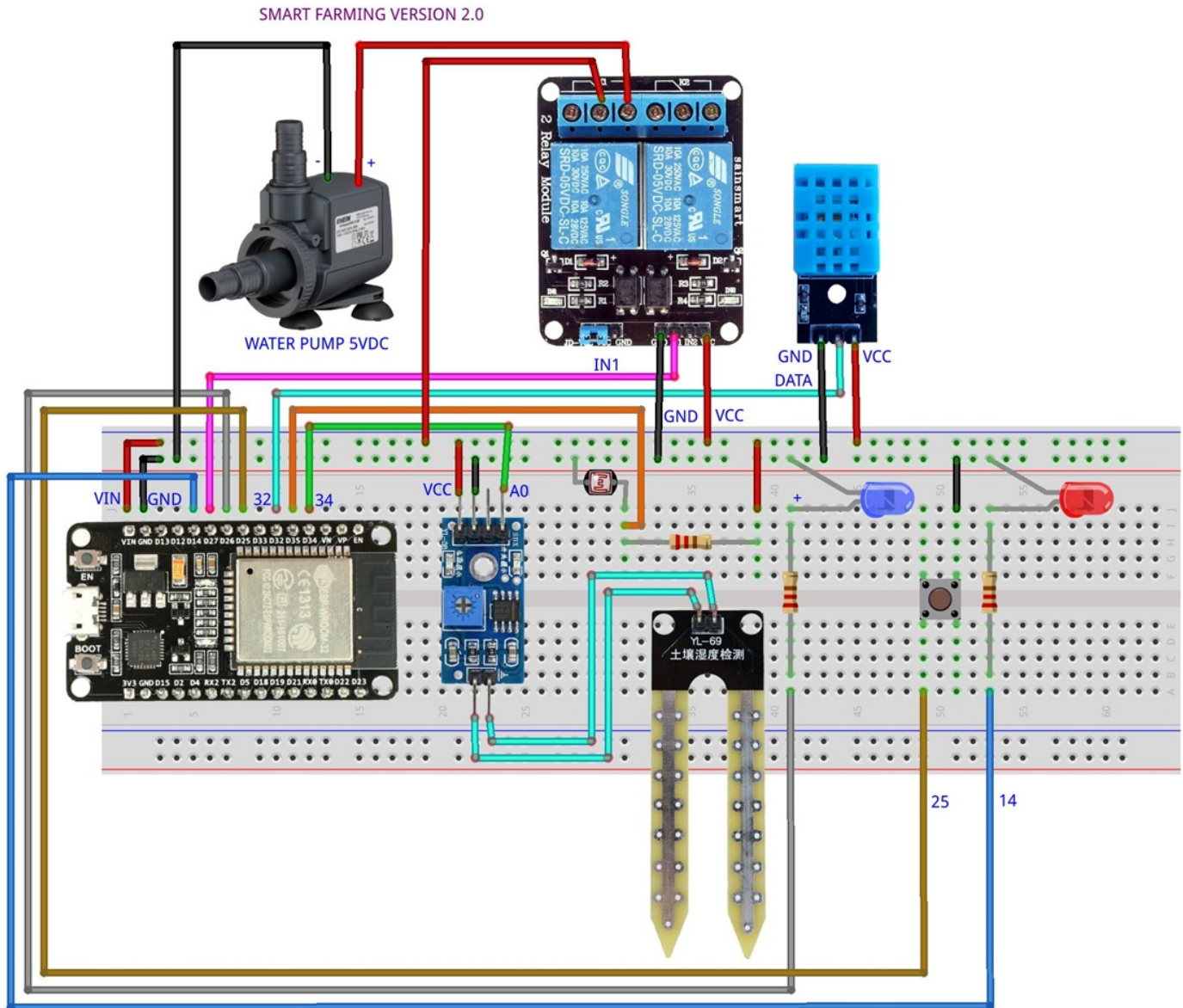


EX21_SMART FARM PROJECT

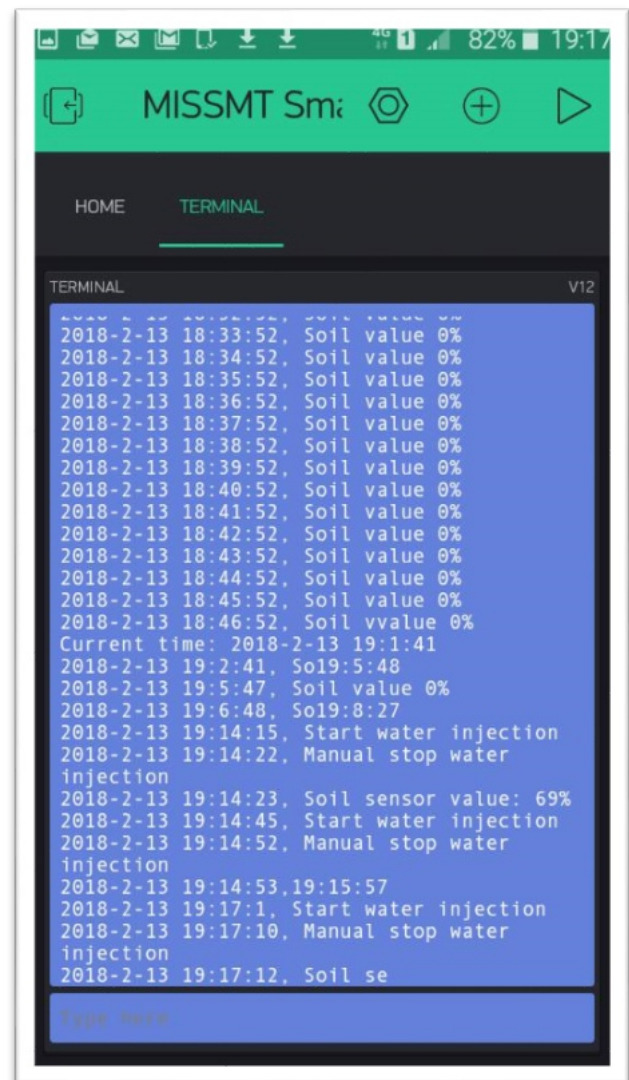
ตัวอย่างทดสอบ การประยุกต์ใช้ ESP32 ควบคุมการปลูกพืชแบบอัจฉริยะแบบ SMART FARM ส่งข้อมูลเข้า CLOUD และสามารถดูการทำงานผ่านทาง dashboard บนสมาร์ตโฟนได้

หลักการทำงาน ถ้าเซ็นเซอร์วัดความชื้นในดินถึงเกณฑ์ที่กำหนด (ดินแห้ง) ระบบจะเปิดหลอดไฟ LED แล้วแจ้งเตือนการเริ่มการทำงานผ่าน Line Notify ระบบสั่งให้ปั๊มน้ำทำงานเพื่อรดน้ำต้นไม้ตามระยะเวลาที่กำหนด ปิดหลอดไฟ LED แล้วแจ้งรายงานการทำงานเสร็จสมบูรณ์ผ่าน Line Notify แต่ถ้าต้องการรดน้ำต้นไม้เอง ก็สามารถทำได้โดยกดปุ่มสวิทช์

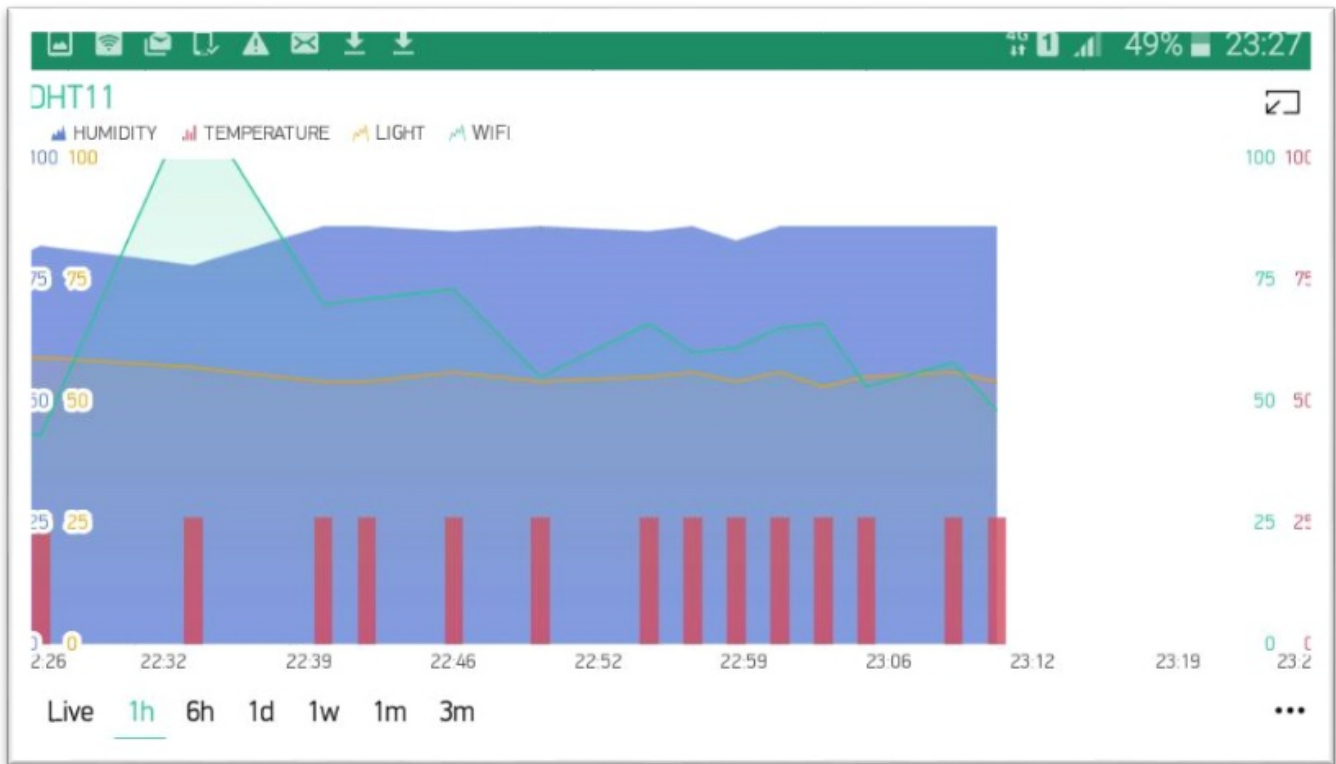
ระบบจะรายงานค่าต่างๆ เข้า Blynk server สามารถดูค่าจากเซ็นเซอร์ต่างๆ เช่น ความเข้มแสง อุณหภูมิ ความชื้นสัมพัทธ์ สถานะของปั๊มน้ำ บน dashboard ในสมาร์ตโฟน เช่น มือถือ ไอแพด ได้



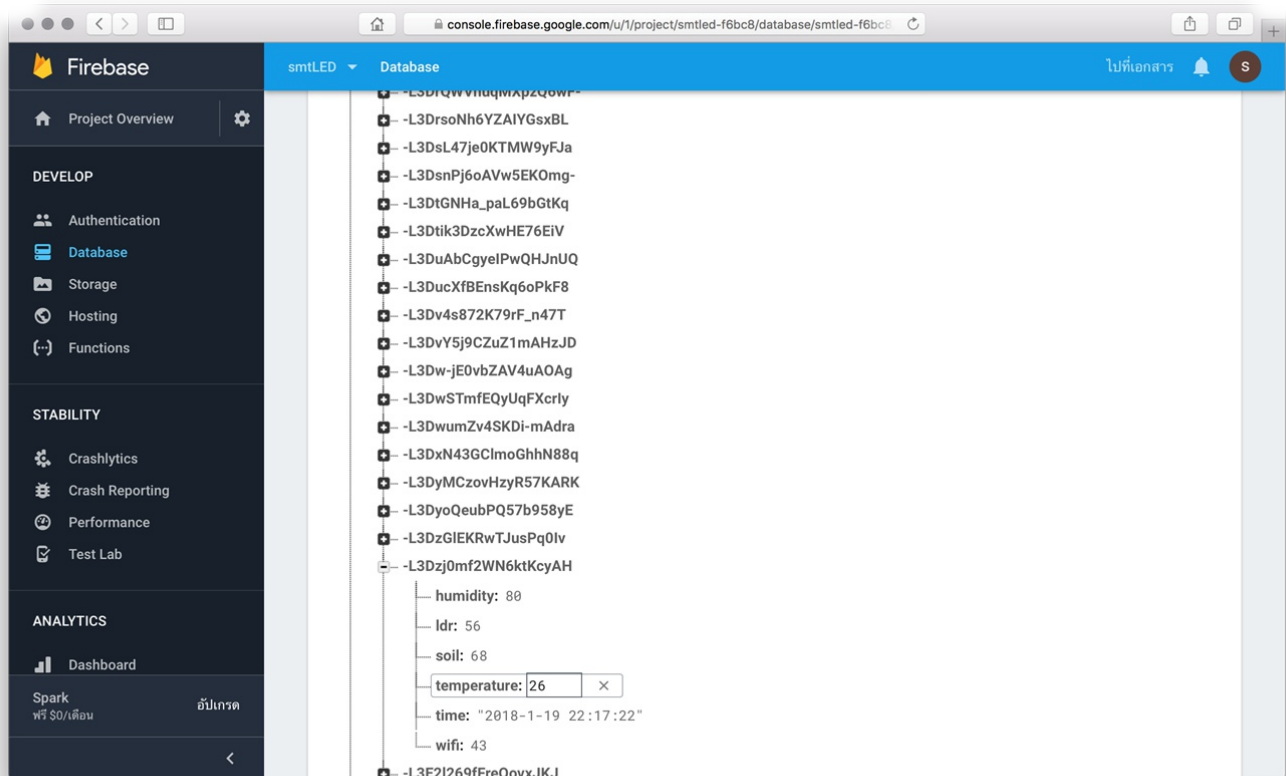
fritzing



หน้าใช้งาน Blynk







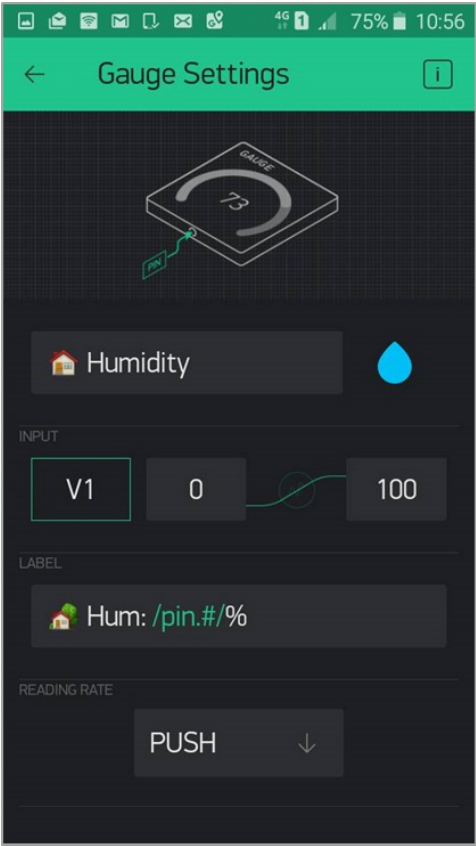
หน้า Firebase

การตั้งค่า Blynk



Widget:

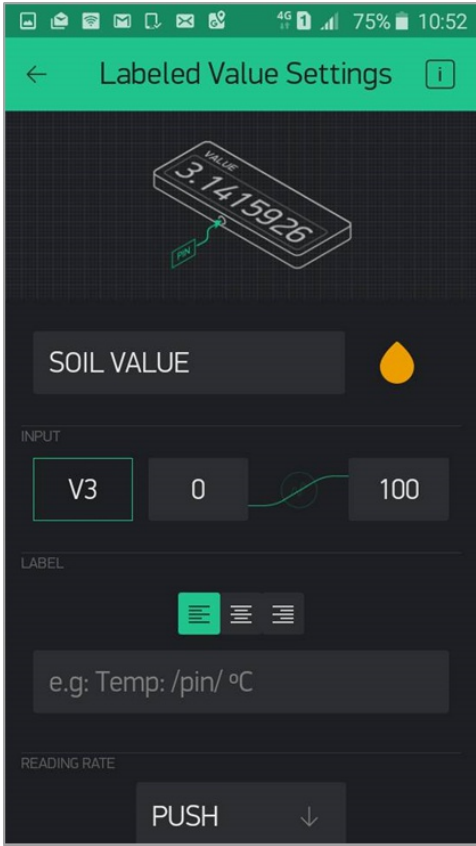
- Widget_Humidity V1
- Widget_Temperature V2
- Widget_SoilValue V3
- Widget_SoilLevel V4
- Widget_LightValue V5
- Widget_LightLevel V6
- Widget_RelayControl V7
- Widget_LedStatus V8
- Widget_RelayStatus V9
- Widget_WifiSignal V10
- Widget_WifiValue V11
- Widget_Terminal V12



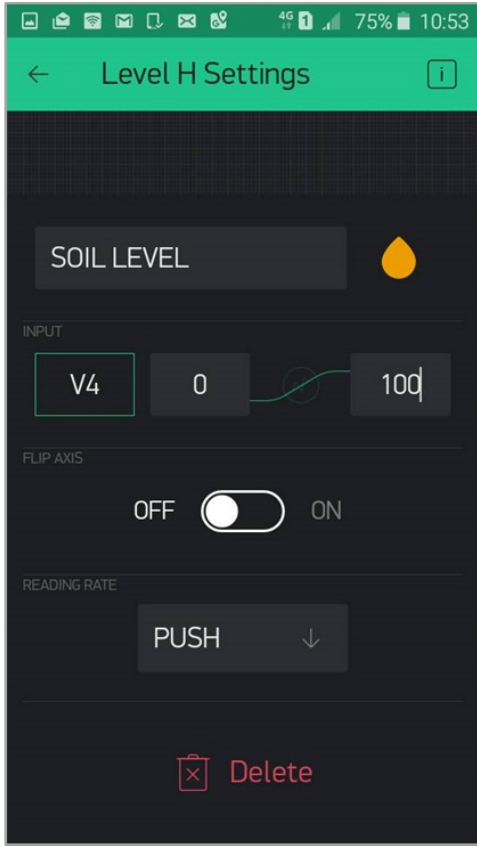
V1 Humidity



V2 Temperature

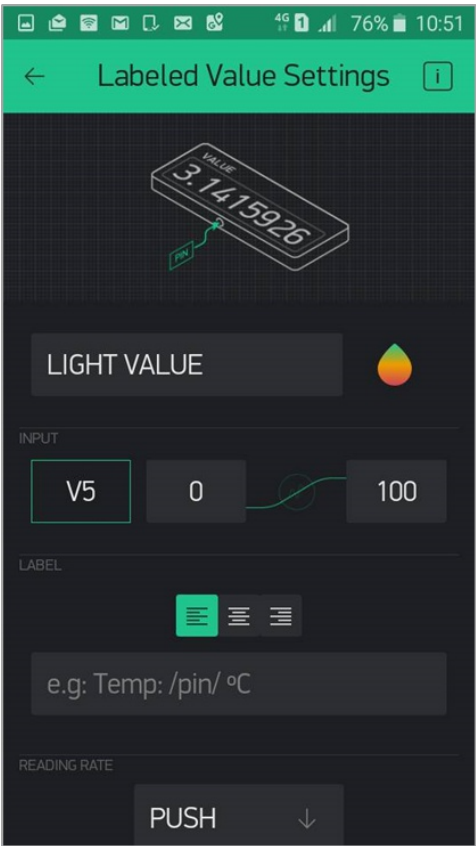


V3 Soil Value

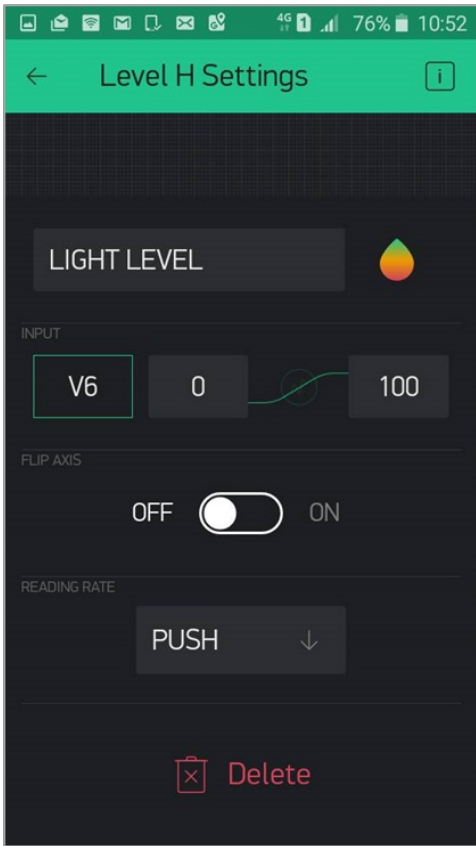


V4 Soil Level

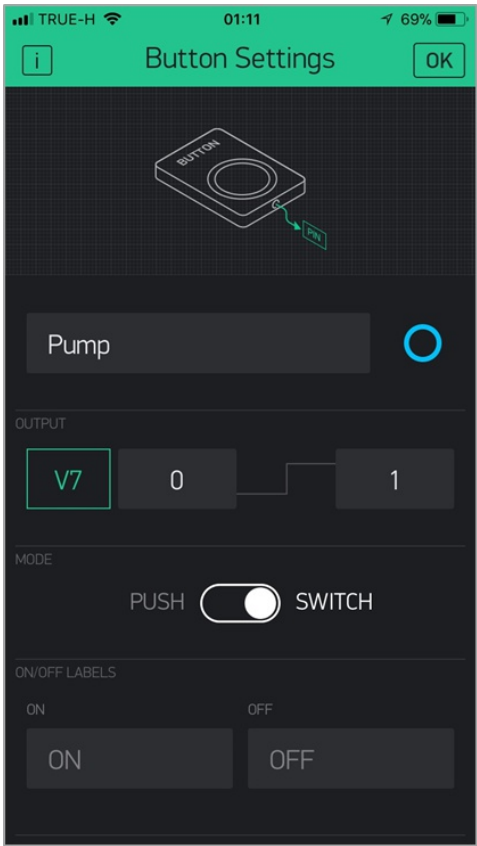
EX21 SMART FARM PROJECT



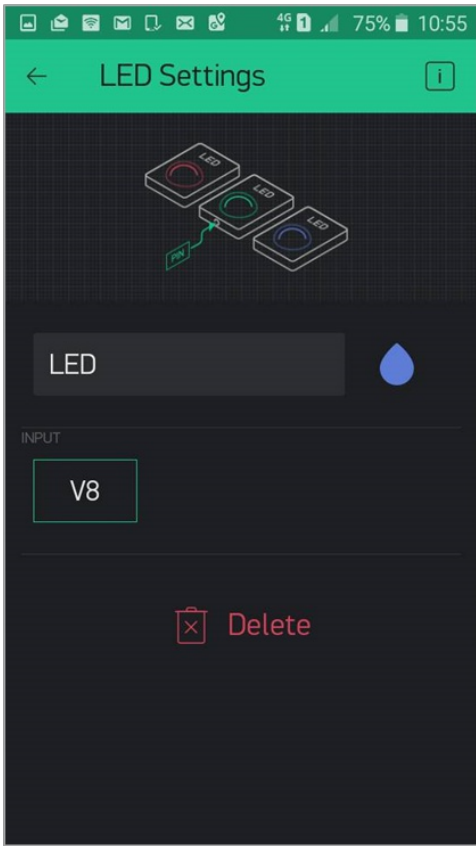
V5 Light Value



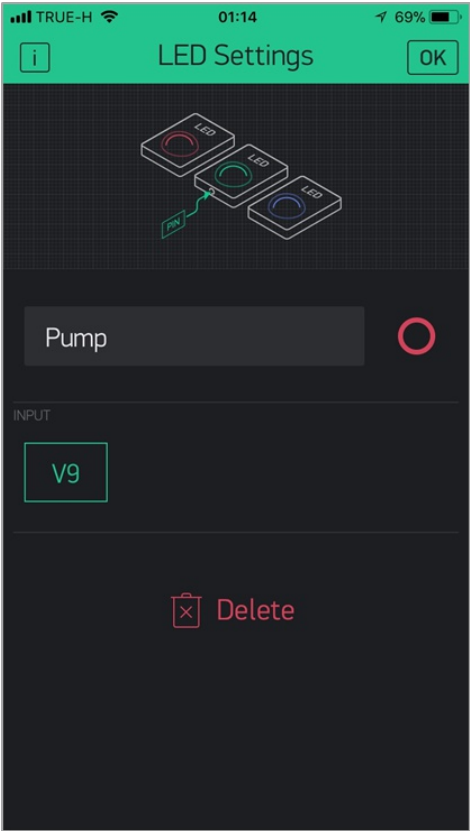
V6 Light Level



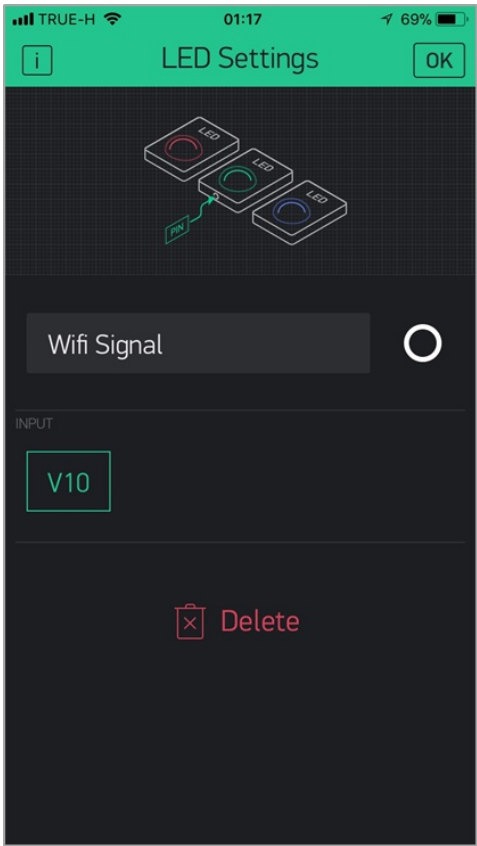
V7 RelayControl



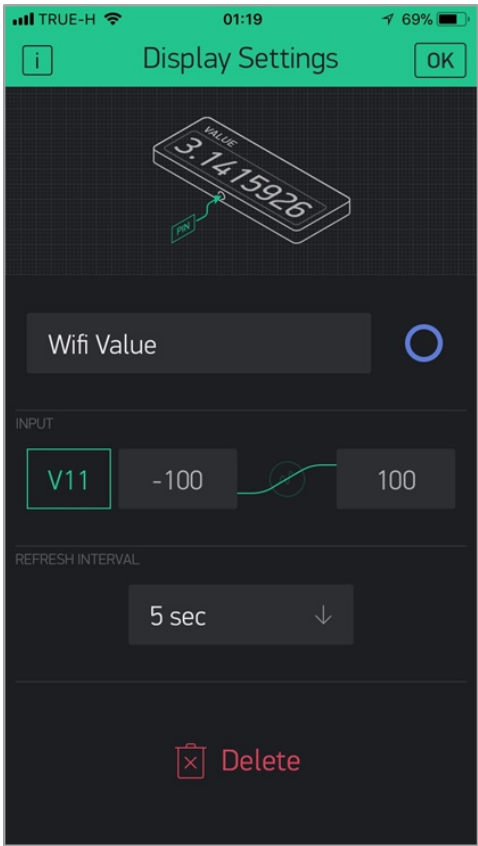
V8 LED status



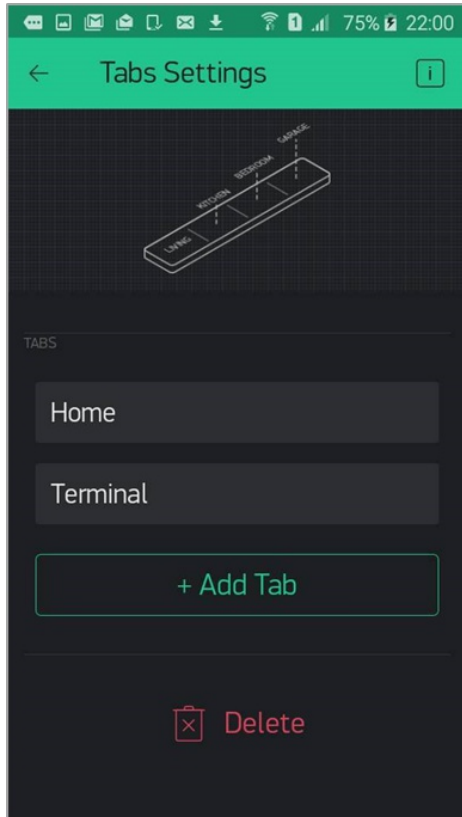
V9 RelayStatus



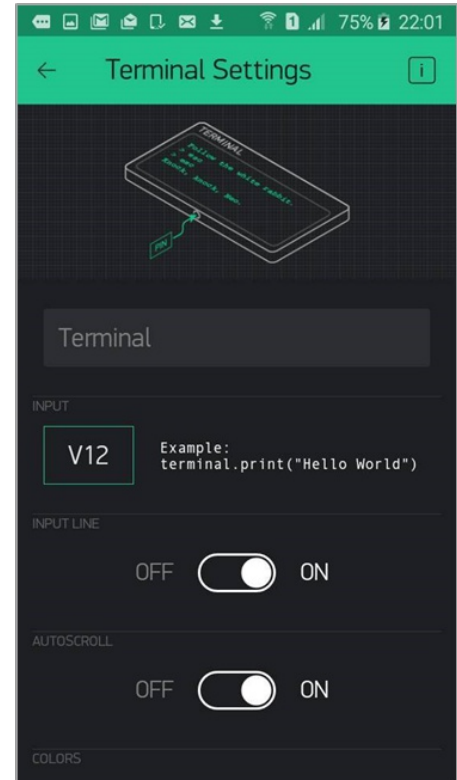
V10 WifiSignal



V11 WifiValue



TAB ให้ไปเพิ่มที่ Wiget ->
interface -> TAB แล้วตั้งค่า
ตามรูป



Terminal ให้ไปเพิ่มที่ Widget
-> Display -> Terminal
V12