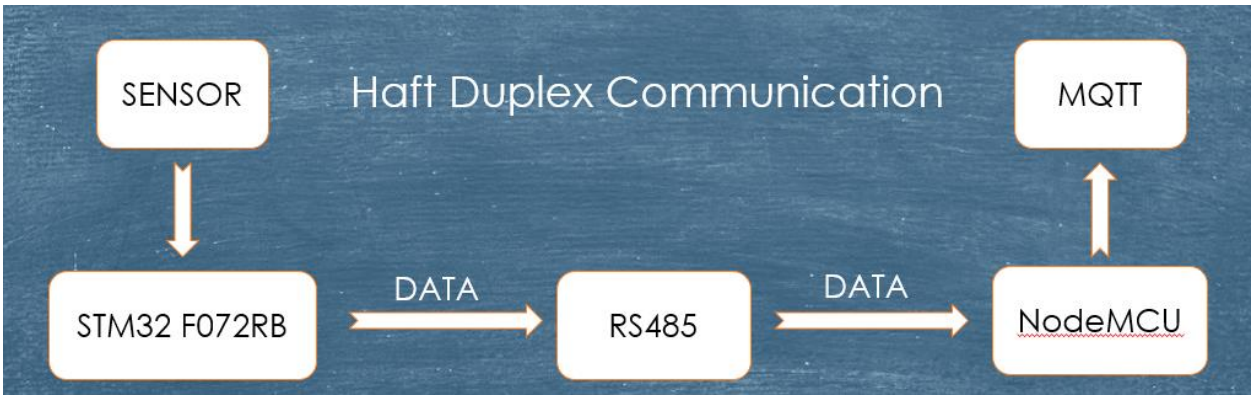
กลุ่มที่ 3

การส่งข้อมูลจาก STM32F072RB ผ่าน Modbus RS485 ไปยัง NodeMCU

การทดลองนี้เป็นการทดลองที่ส่งข้อมูลที่อ่านได้จาก Sensor Potentiometer ที่เชื่อมต่อกับ STM32F072RB แล้วส่งข้อมูลที่ได้เป็นชุดข้อมูล String ส่งข้อมูลผ่าน RS485 จากนั้นก็ส่งไปยัง NodeMCU จากนั้น NodeMCU ก็จะส่งข้อมูลไปยัง Cloud MQTT ซึ่งแสดงตาม Block Diagram ข้างล่าง



Block Diagram แสดงการเชื่อมต่ออุปกรณ์

หลักการทำงานของ RS485

RS232, **RS422**, **RS423** และ **RS485** เป็นการสื่อสารแบบ Serial สำหรับคอมพิวเตอร์และอุปกรณ์ต่างๆ ถ้าพูดถึง RS232 คงจะรู้จักกัน เพราะมันเป็นฮาร์ดแวร์มาตราฐานที่ติดมากับเครื่อง Desktop (ความจริงในสมัยก่อนเครื่อง Notebook ก็มี RS232)

RS232 จะมีข้อจำกัดอยู่หลายอย่าง เช่น ความยาวของสายต้องไม่เกิน 50 ฟุต และความเร็วสูงสุดอยู่ที่ 20 kbs ซึ่งไม่เพียงพอสำหรับการสื่อสารที่ต้องเดินสายไกล/ความเร็วสูง ต่อมาได้มี RS485 มาแทนที่ RS232

ปัญหาหลักของ RS232 คือไม่ทนต่อ Noise เนื่องจากข้อมูลในสาย TX และ RX ต้องเปรียบเทียบระดับสัญญาณกับ GND เมื่อ GND ถูกรบกวนทำให้ GND เปลี่ยนไปจากเดิม แต่ RS485 ไม่ได้ใช้การอ้างอิงสัญญาณกับ GND RS485 ใช้ความแตกต่างระหว่างสาย 2 สาย (A และ B) เป็นตัวบอกว่า Logic “1” หรือ Logic “0” วิธีนี้จะป้องกัน GND loop ที่เกิดขึ้น จากประสบการณ์ที่ใช้งานพบว่าสายแบบ Twist จะป้องกัน Noise ได้ดีกว่าสายตรงที่เดินขนานกันไป และจะให้ดียิ่งขึ้นต้องเป็นสายที่ Shield จะสามารถป้องกันสนามไฟฟ้า สนามแม่เหล็กเข้ามากวนได้

การใช้ไมโครคอนโทรลเลอร์สื่อสารแบบ RS485

RS485 เป็นการรับส่งแบบ Half-Duplex การเขียนโปรแกรมจะกำหนดให้มี Master 1 ตัวเพื่อคอยจัดคิวการสื่อสารใน Network และให้อุปกรณ์ที่เหลือเป็น Slave โดย Slave แต่ละตัวจะมี Address ของตัวเอง เวลาที่ Master ต้องการจะสื่อสารกับ Slave ทำได้โดย ส่ง Address ที่ต้องการจะสื่อสารออกไป แล้วตามด้วยฟังก์ชัน Slave ทุกตัวจะรับข้อมูลได้เหมือนกัน Slave จะเช็คดูว่า Address นั้นใน Address ของตัวเองหรือไม่ ถ้าเป็น Address ของตัวเองก็จะทำการตอบข้อมูลกลับตามที่ Master ต้องการ

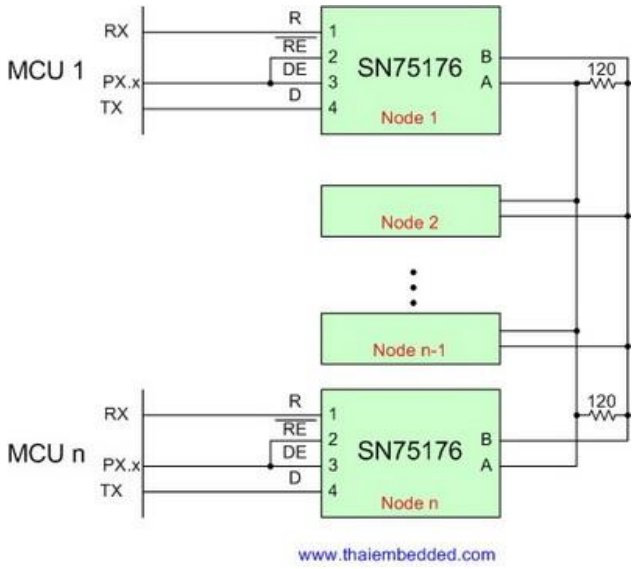
โปรโตคอลที่ใช้ในการสื่อสารใน RS485

เราสามารถกำหนดโปรโตคอลเองได้ว่าจะให้มีลักษณะยังไง หรือจะใช้ Open โปรโตคอลก็ได้เช่นโปรโตคอล MODBUS ที่นิยมใช้ใน PLC งานอุตสาหกรรม

IC ที่นิยมใช้แปลงสัญญาณ UART <—> RS485 จะเป็นเบอร์ SN75176 มีราคาถูก สามารถต่อได้มากสุด 32 Node

คำแนะนำในการเขียนโปรแกรม

โดยปกติแล้วเราจะ Jump RE และ DE ของ SN75176 เข้าด้วยกัน เวลาจะส่งข้อมูลออกไปต้องให้ MCU ส่ง “1” มาที่ขา RE และ DE เพื่อ Enable การส่งและเมื่อส่งข้อมูลเสร็จแล้วต้องส่ง “0” มาที่ขา RE และ DE เพื่อรอรับข้อมูล ใน Bus RS485 ถ้ามีตัวใดตัวหนึ่ง Enable DE ไว้ตัวที่เหลือจะไม่สามารถส่งข้อมูลได้เลยและเมื่อส่งข้อมูลเสร็จแล้วต้องส่ง “0” มาที่ขา RE และ DE เพื่อรอรับข้อมูล ใน Bus RS485 ถ้ามีตัวใดตัวหนึ่ง Enable DE ไว้ตัวที่เหลือจะไม่สามารถส่งข้อมูลได้เลย



รูปแบบของ Network (Network Topology)

ในการเดินสาย RS485 ที่ถูกต้องจะต้องเดินเป็นเส้นยาว เราจะเรียกอุปกรณ์ในแต่ละตัวว่า Node เราจะวาง Node แรกไว้ที่ต้นสาย และ Nodeสุดท้ายไว้ที่ปลายสาย และNode อื่นๆ ก็จะ Jump เข้าที่กลางสาย ใน Datasheet จะแนะนำว่าให้ต้อง R ค่า 120 Ohm ไว้ที่หัวและท้ายของสายด้วยตำแหน่งละตัว ถ้าเราเอา VOM มาวัดความต้านทานในสาย A และ B จะได้ความต้านทานเท่ากับ R 120 Ohm ขนานกัน หรือเท่ากับ 60 Ohm

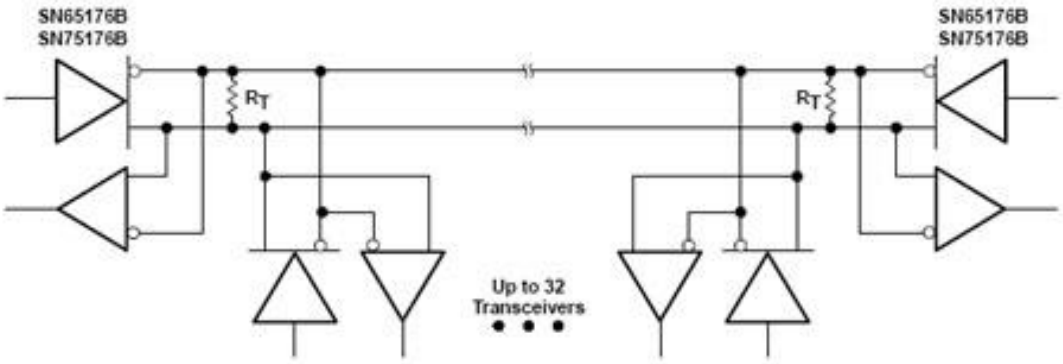
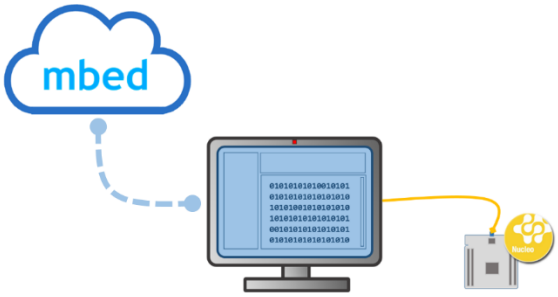


Figure 17. Typical Application Circuit

NOTE: The line should be terminated at both ends in its characteristic impedance ($R_T = Z_O$). Stub lengths off the main line should be kept as short as possible.

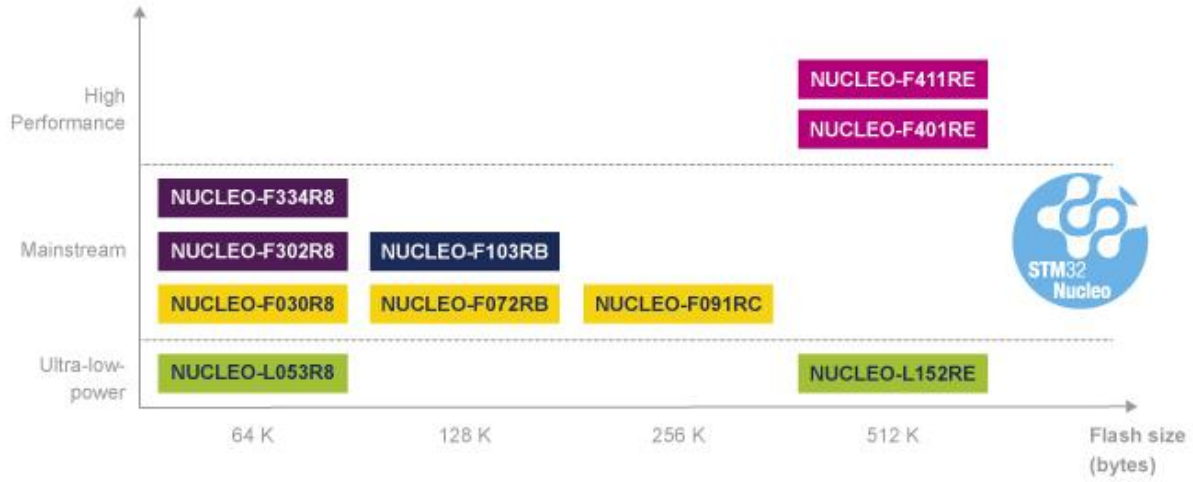
ไมโครคอนโทรเลอร์ STM32 Nucleo



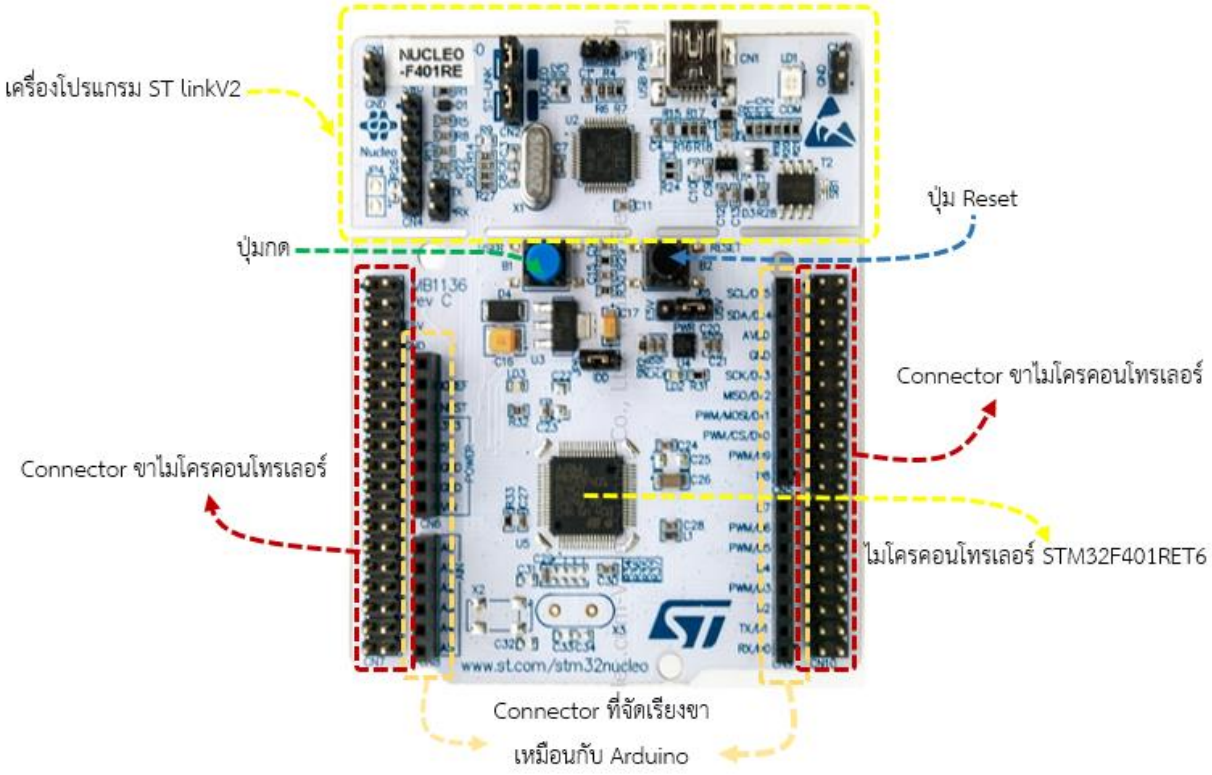
บอร์ด STM32 Nucleo เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล ARM ขนาด 32 bit จากบริษัท STMicroelectronics จุดเด่นของบอร์ด Nucleo คือใช้การพัฒนาอยู่บนแพลตฟอร์ม mbed สำหรับบอร์ดตระกูล ARM Cortex-M จากบริษัท ARM ผ่านโปรแกรมบนหน้าเว็บไซต์ mbed ผู้ใช้สามารถสามารถสร้างโปรเจ็ค เขียน แก้ไข หรือบันทึก โค้ดโปรแกรมผ่านเว็บไซต์ แล้วดาวน์โหลดโปรแกรมไปที่บอร์ดได้โดยตรงโดยไม่ต้องติดตั้งโปรแกรมสำหรับการพัฒนาบนคอมพิวเตอร์เลย ทำให้ไม่ต้องเสียเวลาดาวน์โหลดหรืออัพเดตโปรแกรมในการพัฒนา และไม่ต้องจ่ายเงินซื้อลิขสิทธิ์ของโปรแกรม รวมทั้งรูปแบบการเขียนโค้ดเป็นภาษา C/C++ ที่เข้าใจง่าย เหมาะสำหรับผู้ใช้งานเริ่มต้นที่ต้องการใช้งานไมโครคอนโทรเลอร์ 32 บิตหรือผู้ใช้งานทั่วไปที่ต้องการพัฒนาบนแพลตฟอร์มที่ใช้งานง่ายและสะดวกรวดเร็ว

เริ่มใช้งาน STM32 Nucleo บน mbed

บอร์ด Nucleo มีรุ่นย่อย ให้ผู้ใช้เลือกตามงานที่เหมาะสม โดยในตัวอย่างนี้เลือกใช้ บอร์ด Nucleo รุ่น F401RE ซึ่งจัดอยู่ในกลุ่มบอร์ดประสิทธิภาพสูง ใช้ MCU ตระกูล ARM Cortex-M4 เบอร์ STM32F401RE ความเร็ว 84 MHz หน่วยความจำแฟลชขนาด 512 Kbytes sหน่วยความจำแรมขนาด 96 Kbytes ขา I/O จำนวน 81 ขา ทุกขา I/O ทนแรงดันที่ 5V ได้ (5 V-tolerant)



แนะนำบอร์ด STM32 Nucleo รุ่น F072RB

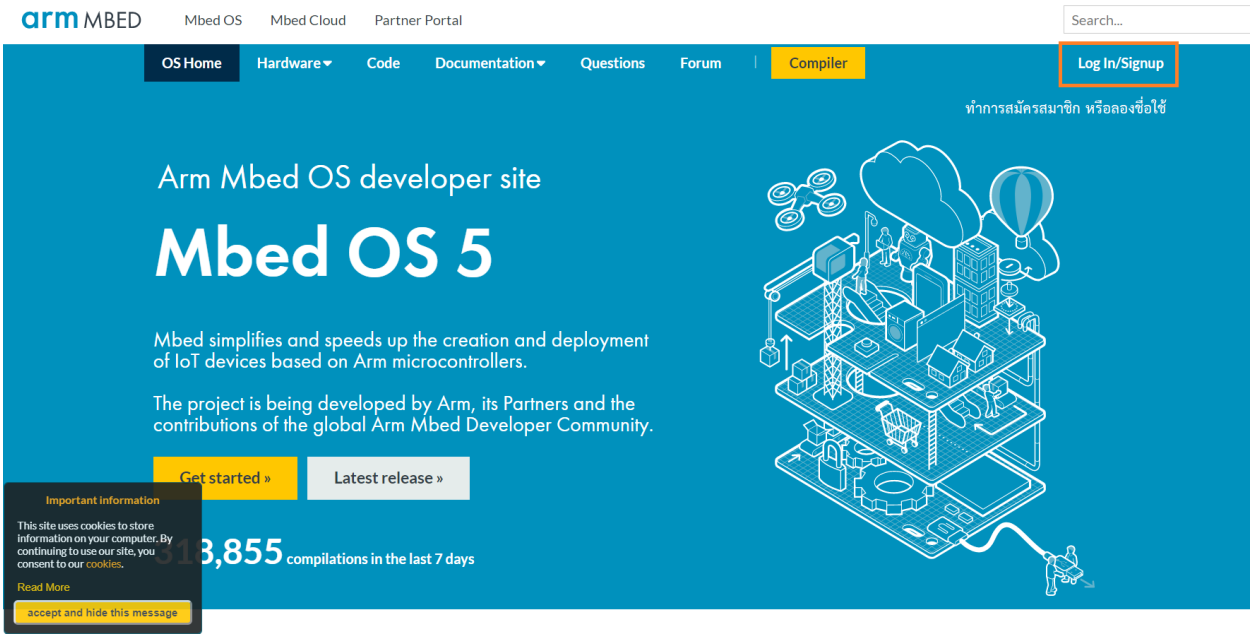


ตัวบอร์ดแยกเป็น 2 ส่วนคือ

1. เครื่องโปรแกรม ST link V2 ใช้สำหรับโปรแกรม MCU บนบอร์ดเชื่อมต่อเข้ากับคอมพิวเตอร์ผ่านพอร์ต USB หรือใช้เป็นเครื่องโปรแกรมให้กับบอร์ดอื่น โดยใช้ Pin Header ที่อยู่ด้านข้าง

2. บอร์ดไมโครคอนโทรลเลอร์ STM32F072RB ขาต่างๆ ออกแบบมาให้ สามารถใช้ร่วมกับ Shield ของ Arduino ได้ ขาที่เหลือจัดเป็น Pin Header ด้านข้างของบอร์ด ปุ่มกดสำหรับรับค่าจากผู้ใช้ และปุ่ม reset ดังภาพ

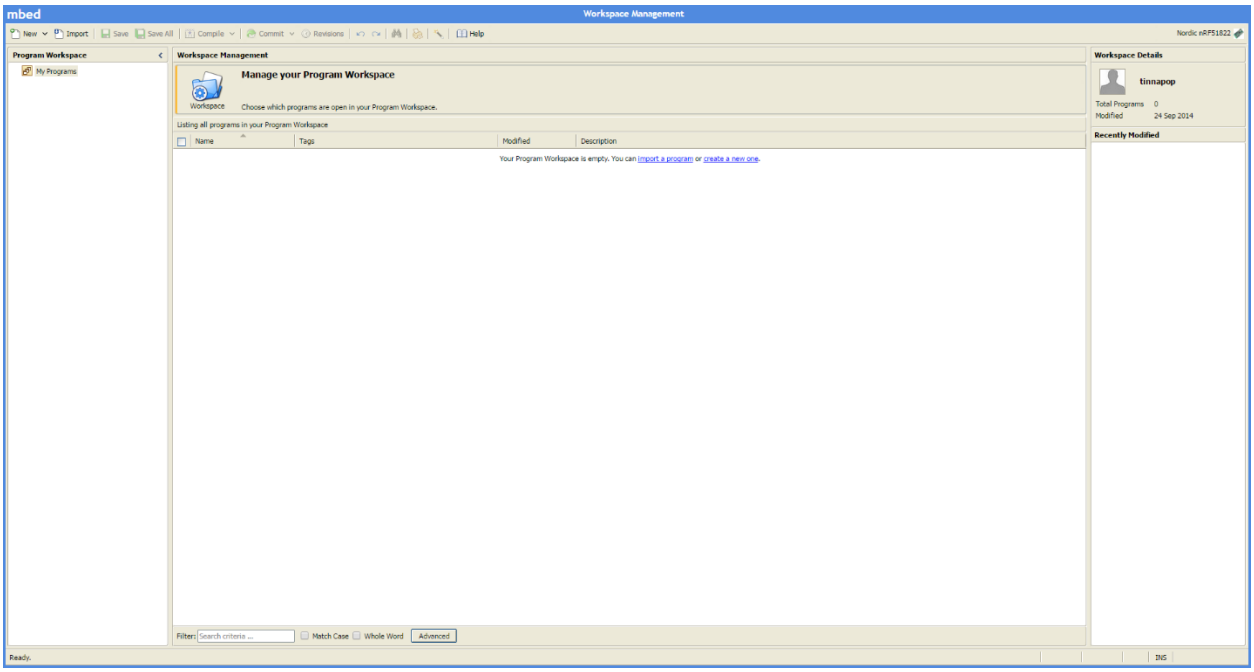
เริ่มใช้งาน STM32 Nucleo



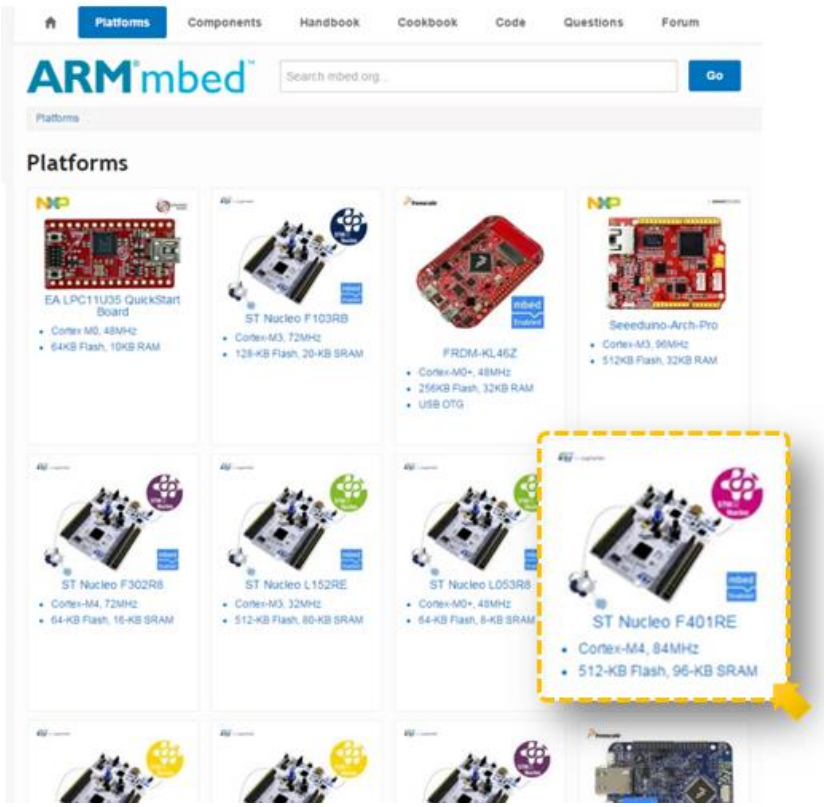
ขั้นตอนที่ 1 Login หรือสมัครสมาชิก

ในขั้นแรก เราต้องเข้าลงชื่อเข้าใช้ หรือถ้าใช้งานครั้งแรกให้สมัครสมาชิกก่อน โดยเข้าเวปไปที่ http://developer.mbed.org/ คลิกที่ Login or signup

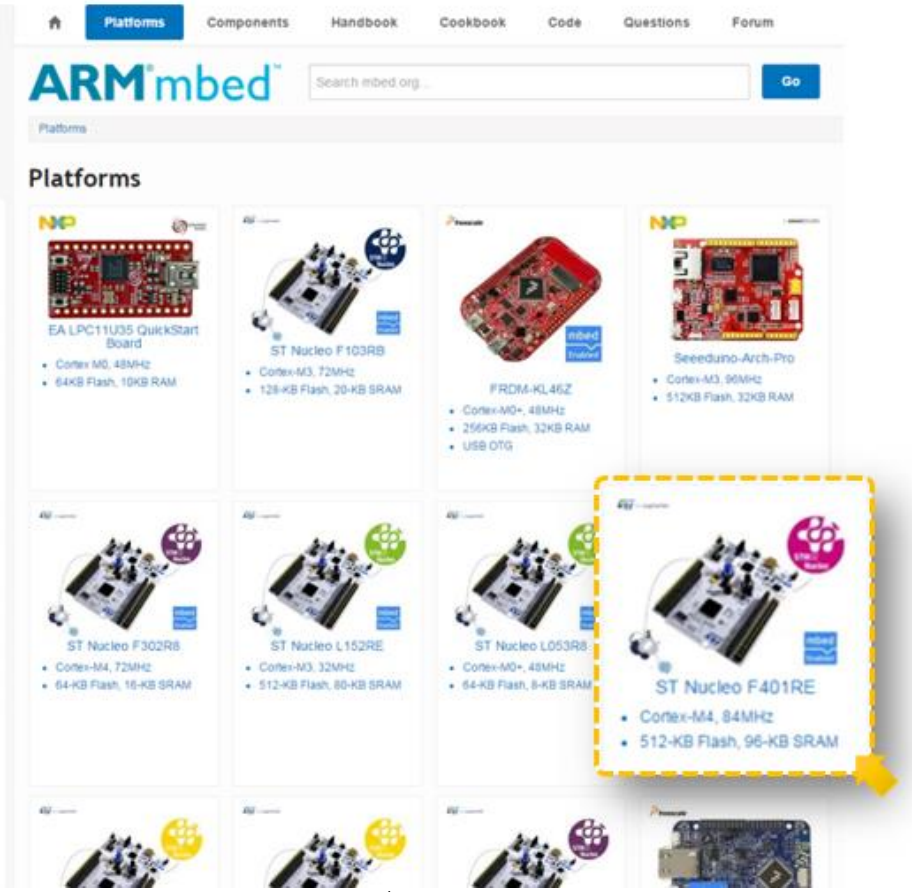
เมื่อล็อกอินแล้วจะปรากฏชื่อล็อกอินที่มุมบนขวาหน้าเว็บกดปุ่ม Compiler เพื่อเข้าสู่หน้าโปรแกรม Compiler ของ mbed



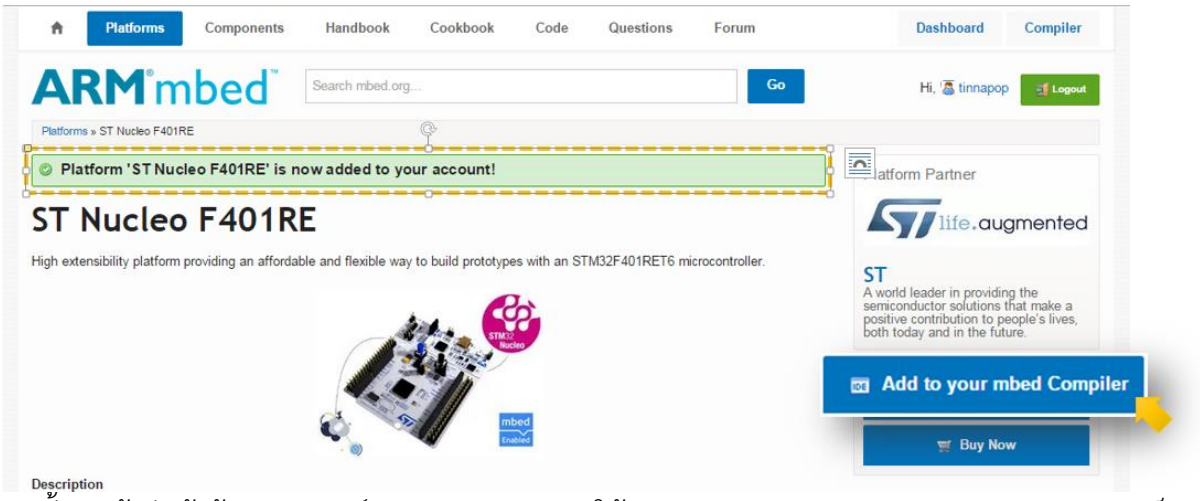
ขั้นตอนที่ 2 เลือกบอร์ดที่ต้องการใช้



ให้ผู้ใช้สังเกตที่มุมบนขวาของหน้าเว็บแสดงข้อความ "No device select" คือยังไม่ได้เลือกบอร์ดใช้งาน



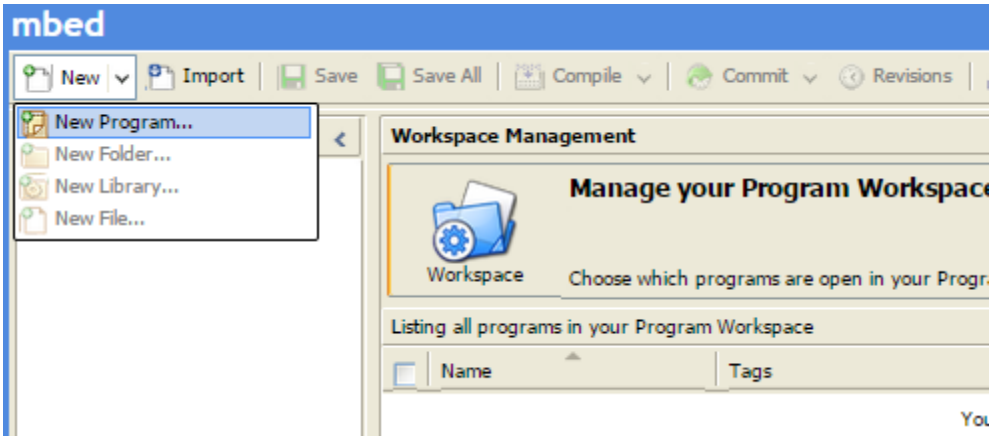
ให้ผู้ใช้เลือกบอร์ดใช้งานโดยเข้าไปที่ http://developer.mbed.org/platforms/ ให้เลือกบอร์ด ST Nucleo F401RE ดังภาพ



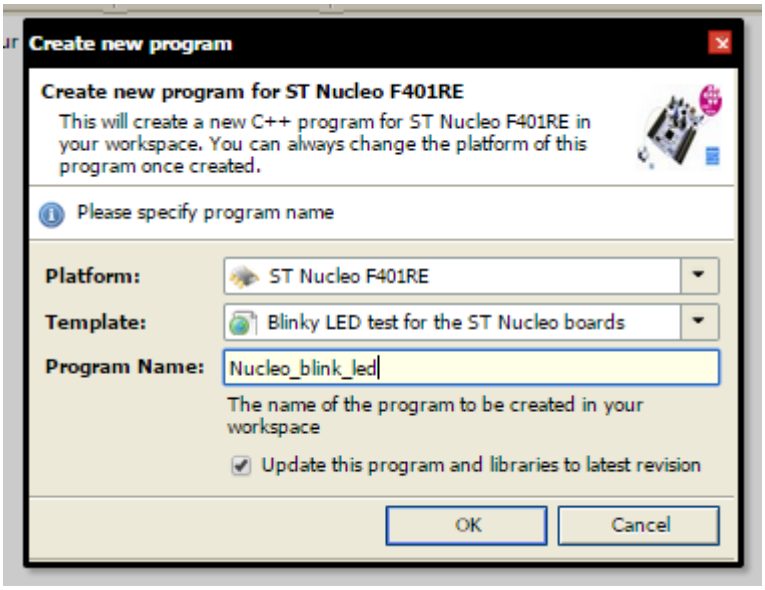
จากนั้นจะเข้าสู่หน้าข้อมูลของบอร์ด Nucleo F401RE ให้กดปุ่ม "Add to your mbed Compiler" จะเห็นข้อความ "Platform 'ST Nucleo F401RE' is now added to your account!" ด้านบน แสดงว่าผู้ใช้ได้ทำ

การเลือกบอร์ดแล้ว

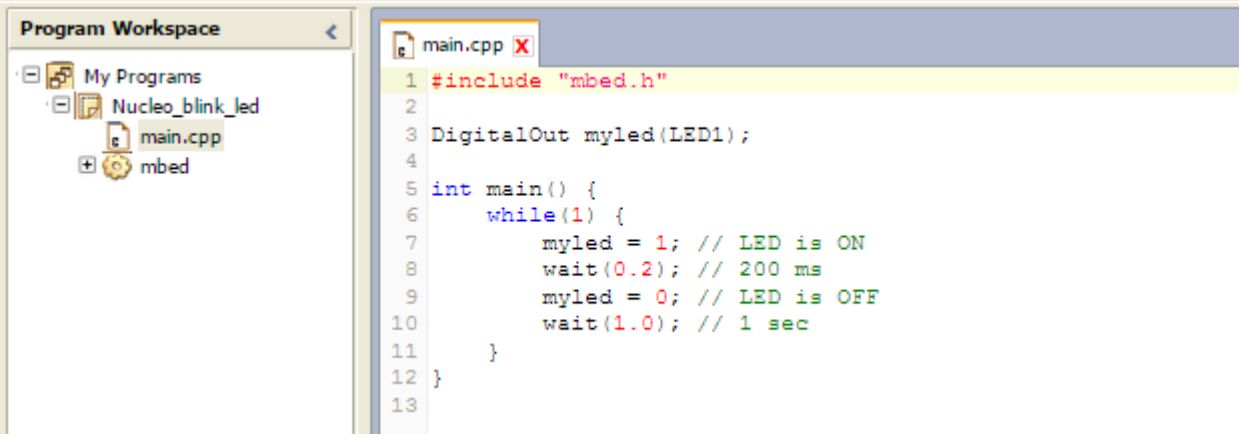
ขั้นตอนที่ 3 เขียนโปรแกรม



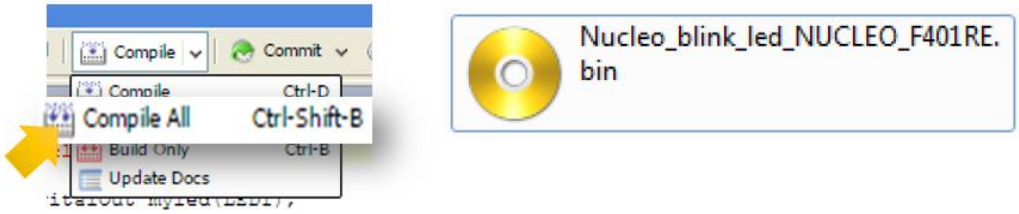
กลับมาที่หน้า Compiler เลือกที่หัวข้อ New > New Program... เพื่อสร้างโปรเจคใหม่



เว็บจะแสดงหน้าต่าง Create new program ที่ช่อง Platform เลือกเป็น ST Nucleo F401E ในช่อง Template เลือกเป็น “Blinky LED test for ST Nucleo board” โดยโปรแกรมนี้ทำไฟกระพริบ LED บนบอร์ด ตั้งชื่อ Program Name จากนั้นกดปุ่ม OK

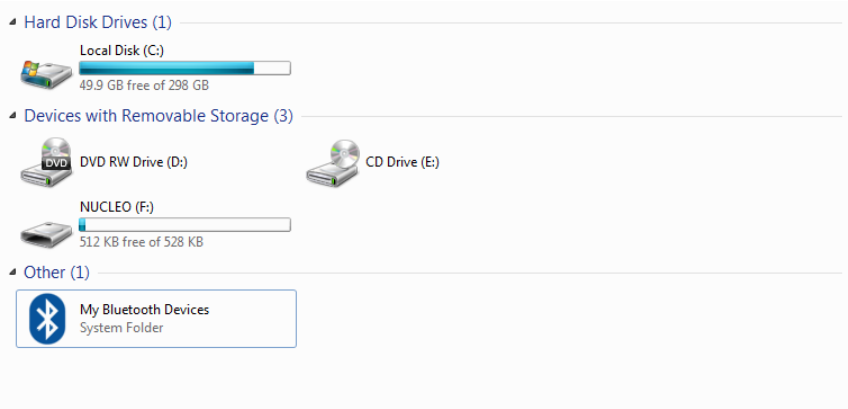


จากนั้นจะแสดงหน้าโค้ดโปรแกรมจะเห็นได้ว่าในไฟล์ main.cpp กำหนดให้ขา LED1 คือขาที่ PA5 เป็น Output จากนั้นวนกำหนดให้ขา LED1 เป็น High จากนั้น Delay 0.2 วินาที กำหนดขา LED1 เป็น Low และ Delay 1 วินาที

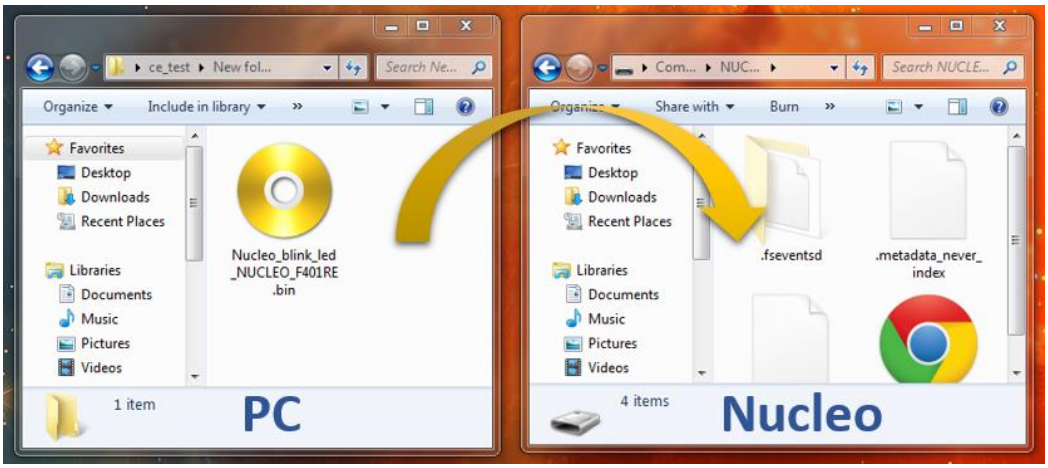


จากนั้นกดปุ่ม Compile All ถ้าไม่ข้อผิดพลาดอะไร จากหน้าเวปจะให้ดาวโหลดไฟล์ Nucleo_blink_led_NUCLEO_F401RE.bin

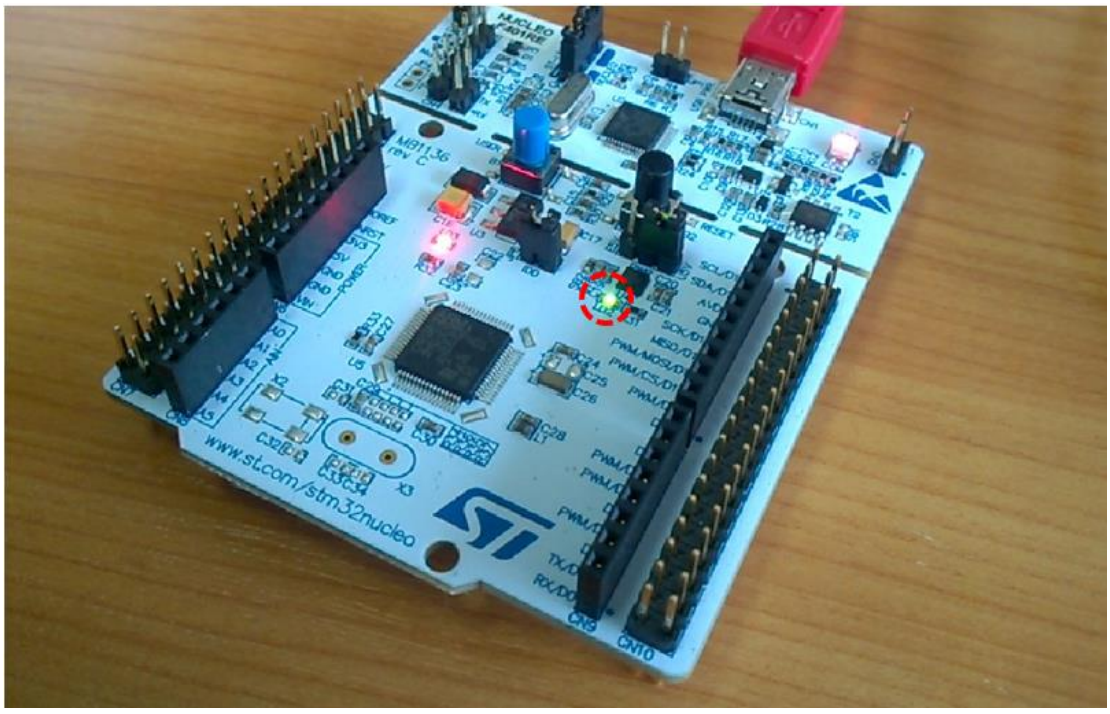
ขั้นตอนที่ 4 โปรแกรมบอร์ด Nucleo



เชื่อมต่อบอร์ด Nucleo ผ่านพอร์ต USB เข้าคอมพิวเตอร์ เมื่อดูที่ My Computer จะเห็นบอร์ดเป็น Drive ตัวหนึ่ง

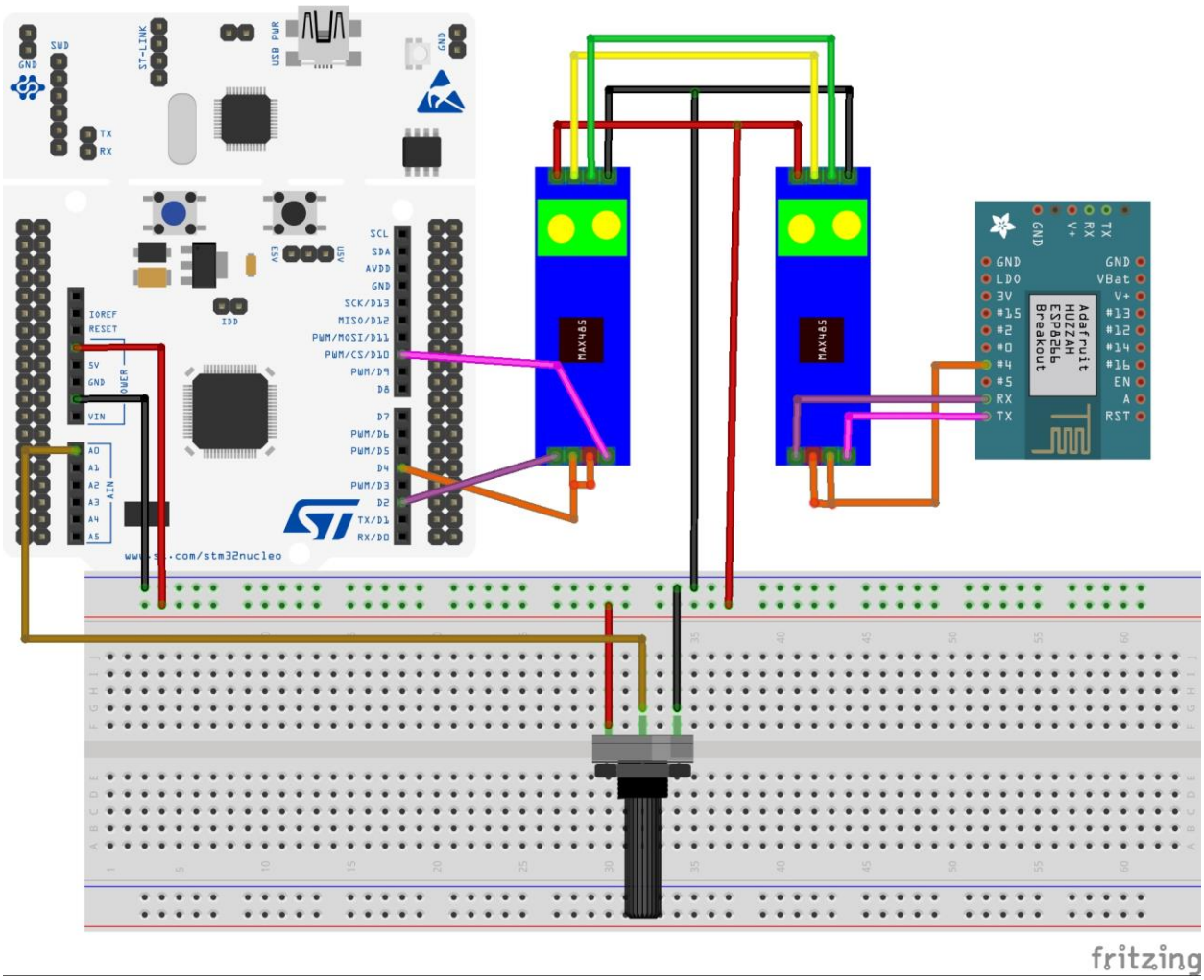


คัดลอกไฟล์บนเครื่องไปยังบอร์ด



จะเห็นว่าโปรแกรมไฟกระพริบที่ผู้ใช้โปรแกรมลงไปที่บอร์ดได้ทำงานแล้ว

การต่ออุปกรณ์


SWD
GND
ST-LINK
USB PWR
GND
TX
RX
SCL
SDA
AVDD
GND
SCK/D13
MISO/D12
PWM/MOSI/D11
PWM/CS/D10
PWM/D9
D8
D7
PWM/D6
PWM/D5
D4
PWM/D3
D2
TX/D1
RX/D0
IOREF
RESET
5V
GND
VIN
A0
A1
A2
A3
A4
A5
MAX4485
MAX4485
GND
LDO
3V
#15
#2
#0
#4
#5
RX
TX
GND
VBat
V+
#13
#12
#14
#16
EN
A
RST
Adafruit
HUZZAH
ESP8266
Breakout
fritzing

โค้ดสำหรับการส่งข้อมูลที่อ่านจาก Potential meter STM32

```
#include "mbed.h"

Serial RS485(D10,D2); // TX , RX
Serial pc(USBTX,USBRX);
DigitalOut enablePin(D4); //เวลาจะส่งข้อมูลออกไปต้องให้ MCU ส่ง “1” มาที่ขา RE และ DE เพื่อ Enable
AnalogIn analog_value(A0);//อ่านค่าจาก potential meter

int main() {

   enablePin = 1; // enable to tranmitter
   float meas;
   RS485.baud(9600);

   while(1) {
      meas = analog_value.read();//อ่านค่า Analog
      meas = meas * 3300;
      RS485.printf("&.0f\n", mean);
      wait(0.2);
   }
}
```

โค้ดสำหรับ NodMCU การรับข้อมูลที่ได้จาก STM32

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>

SoftwareSerial com_serial(D2,D3); // RX, TX
int enablePin = D4;
int num;


// Update these with values suitable for your network.
const char* ssid = "Phone";  // WIFI NAME
const char* password = "Kengkkun1234"; // WIFI PASSWORD


// Config MQTT Server
#define mqtt_server "m14.cloudmqtt.com"
#define mqtt_port 10670
```

```
#define mqtt_user "TEST"
#define mqtt_password "1234"

//#define LED_PIN D5

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
//  pinMode(LED_PIN, OUTPUT);
  pinMode(enablePin, OUTPUT);
  Serial.begin(9600);
  com_serial.begin(9600);
  digitalWrite(enablePin, LOW); // เมื่อส่งข้อมูลเสร็จแล้วต้องส่ง “0” มาที่ขา RE และ DE เพื่อรอรับข้อมูล
  delay(10);

  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

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

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  client.setServer(mqtt_server, mqtt_port); // (server, port)
  client.setCallback(callback);
}

void loop() {
  int n;
  char message[64];
  while(com_serial.available()) {
    message[n] = com_serial.read(); // read data
    n++;
  }
   num = atoi(message); // convert string to int
   //client.publish("/ESP/LED", "Hello");
   client.publish("/ESP/LED", message); //ส่งค่าข้อมูลไปยัง MQTT Cloud
```

```
  if (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client", mqtt_user, mqtt_password)) {
      Serial.println("connected");
      client.subscribe("/ESP/LED");

    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
      return;
    }
  }
  client.loop();
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  String msg = "";
  int i=0;

  /*while (i<length) msg += (char)payload[i++];
  if (msg == "GET") {
    client.publish("/ESP/LED", (digitalRead(LED_PIN) ? "LEDON" : "LEDOFF"));
    client.publish("/ESP/LED", "TEST SENT DATA");
    Serial.println("Send !");
    return;
  }
  digitalWrite(LED_PIN, (msg == "LEDON" ? HIGH : LOW));*/
  Serial.println(msg);
}
```

โค้ด HTML

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>MQTT WebSocket</title>
<script src="jquery-1.11.3.min.js"></script>
<script src="mqttws31.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs
s">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<style>
body {
font-family: Arial, Helvetica, sans-serif;
}

#status {
background: #333;
color: #FFF;
border-radius: 3px;
font-weight: bold;
padding: 3px 6px;
}

#status.connect {
background: #E18C1A;
color: #FFF;
}

#status.connected {
background: #00AE04;
color: #FFF;
}

#status.error {
background: #F00;
color: #FFF;
}

button {
font-size: 32px;
}
```

```
</style>
<script>
var config = {
mqtt_server: "m14.cloudmqtt.com",
mqtt_websockets_port: 30670,
mqtt_user: "TEST",
mqtt_password: "1234"
};

var check = "";
var dataString = "";

$(document).ready(function(e) {
// Create a client instance
client = new Paho.MQTT.Client(config.mqtt_server, config.mqtt_websockets_port, "web_" +
parseInt(Math.random() * 100, 10));
//Example client = new Paho.MQTT.Client("m11.cloudmqtt.com", 32903, "web_" +
parseInt(Math.random() * 100, 10));

// connect the client
client.connect({
useSSL: true,
userName: config.mqtt_user,
password: config.mqtt_password,
onSuccess: function() {
// Once a connection has been made, make a subscription and send a message.
// console.log("onConnect");
$("#status").text("Connected").removeClass().addClass("connected");
client.subscribe("/ESP/LED");
mqttSend("/ESP/LED", "THis is value from NodeMCU");
},
onFailure: function(e) {
$("#status").text("Error : " + e).removeClass().addClass("error");
// console.log(e);
}
});

client.onConnectionLost = function(responseObject) {
if (responseObject.errorCode !== 0) {
$("#status").text("onConnectionLost:" +
responseObject.errorMessage).removeClass().addClass("connect");
setTimeout(function() { client.connect() }, 1000);
}
}
```

```
client.onMessageArrived = function(message) {
// $("#status").text("onMessageArrived:" + message.payloadString).removeClass().addClass("error");
console.log(message.payloadString);

if (check != message.payloadString){ // ลูปเช็คค่าที่ได้รับมาว่าซ้ำกันหรือไม่ ถ้าเกิดซ้ำกันจะไม่โชว์ค่าที่ซ้ำออกมา
Send(message.payloadString);
check = message.payloadString;
}
}
});

var mqttSend = function(topic, msg) {
var message = new Paho.MQTT.Message(msg);
message.destinationName = topic;
client.send(message);
}
var Send = function( msg) { // ฟังก์ชันในการส่งค่าขึ้นโชว์บนหน้าเว็ป
dataString = dataString +msg+"<br>";
document.getElementById("new").innerHTML = dataString;
}

</script>
</head>

<body>
<h1>MQTT WebSocket</h1>
<h3>NodeMCU Controller : <span id="status" class="connect">Connect...</span></h3>
<!-- <hr /> -->

<div class="container">
<div class="panel panel-default">
  <div class="panel-body">

<p id="new" ></p> <!-- เป็นการเรียกข้อมูลที่ได้รับมาให้แสดงบนหน้าเว็ปโดยอยู่ในรูปแบบกล่องข้อความ -->

</div>
</div>
  </div>
</body>
</html>
```

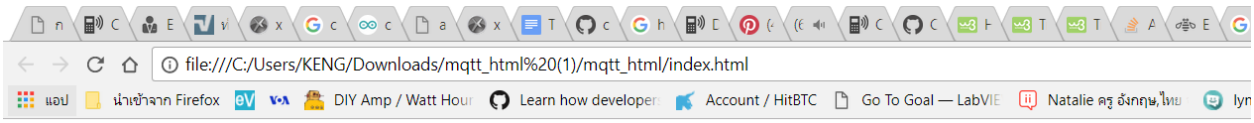
ตัวผลลัพธ์ของโค้ด HTML

MQTT WebSocket

NodeMCU Controller : **Connected**

THis is value from NodeMCU

ผลลัพธ์การส่งข้อมูลผ่านหน้า HTML



MQTT WebSocket

NodeMCU Controller : **Connected**

THis is value from NodeMCU
1353 1430 2028 1509 1348 1445 1414 2008 2682 2218 1489 1351 □□ @
2679 2244 1521 1358 1348 1445 1414 2008 2682 2218 1489 1351 □□ @
1433 2000 2673 1358 1348 1445 1414 2008 2682 2218 1489 1351 □□ @
2267 1534 1372 1431 1348 1445 1414 2008 2682 2218 1489 1351 □□ @
1985 2661 2295 1431 1348 1445 1414 2008 2682 2218 1489 1351 □□ @
1537 1365 1419 1953 1348 1445 1414 2008 2682 2218 1489 1351 □□ @
2434 1600 1378 1407 1760 2585 2511 1646 1389 1393 1667 2525 □ @
2579 1696 1355 1364 1592 2504 2635 1646 1389 1393 1667 2525 □ @
1749 1370 1360 1364 1592 2504 2635 1646 1389 1393 1667 2525 □ @
1550 2454 2656 1783 1592 2504 2635 1646 1389 1393 1667 2525 □ @
1387 1368 1501 1783 1592 2504 2635 1646 1389 1393 1667 2525 □ @
2412 2680 1822 1397 1592 2504 2635 1646 1389 1393 1667 2525 □ @
1355 1476 2377 1397 1592 2504 2635 1646 1389 1393 1667 2525 □ @
2732 1844 1375 1301 1430 2504 2635 1646 1389 1393 1667 2525 □ @
2318 2721 2032 1427 1430 2504 2635 1646 1389 1393 1667 2525 □ @
1339 1449 2265 1427 1430 2504 2635 1646 1389 1393 1667 2525 □ @
2721 2004 1413 1339 1430 2504 2635 1646 1389 1393 1667 2525 □ @
1439 2202 2714 1339 1430 2504 2635 1646 1389 1393 1667 2525 □ @
2098 1455 1339 1422 1430 2504 2635 1646 1389 1393 1667 2525 □ @
2082 2677 2181 1422 1430 2504 2635 1646 1389 1393 1667 2525 □ @
1477 1335 1413 2023 1430 2504 2635 1646 1389 1393 1667 2525 □ @
2674 2258 1512 2023 1430 2504 2635 1646 1389 1393 1667 2525 □ @
1348 1415 1985 2675 1430 2504 2635 1646 1389 1393 1667 2525 □ @

ผลลัพธ์การส่งข้อมูลผ่านหน้า Web Socket