

ESP8266 ESP8285 กับการใช้งาน Firebase ระบบฐานข้อมูลเรียลไทม์จาก Google

HTML Content



* Firebase อัปเดตใหม่เรื่อง Key ทำให้โลบารี่สำหรับ ESP8266 อาจจะใช้งานไม่ได้ในอนาคด

Firestore มีบริการหลักเป็น Realtime Database เกิดขึ้นด้วยแนวคิดที่คนทำแอปพลิเคชันไม่จำเป็นต้องตั้งเซิร์ฟเวอร์เอง และไม่ต้องต้องเขียนโปรแกรมหลังบ้านซ้ำ ๆ แบบเดิม ซึ่งหากคนที่ทำเว็บไซต์ ทำแอปพลิเคชัน จะทราบดีว่างาน 1 โปรเจค จะต้องมีความรู้ และจะต้องมีการเก็บตารางของผู้ใช้งาน ระบบ Log ต่าง ๆ มีการติดต่อกับผู้ใช้ ซึ่งเป็นงานที่มีการทำซ้ำ ๆ ตลอดมา ดังนั้น Firestore จึงมาช่วยแก้ปัญหาดังนี้ได้ ทำให้ไม่ต้องมีการจัดการฐานข้อมูลเอง ไม่ต้องเขียนโปรแกรมหลังบ้านเอง (ด้วยภาษา PHP Python และอื่นๆ)

ตัว Firestore ทำไว้ให้หมดแล้ว

ในงานด้านแอปพลิเคชัน ตัว Firestore ถือเป็นบริการฐานข้อมูลออนไลน์ตัวหนึ่ง ซึ่งแอปพลิเคชันส่วนใหญ่ต้องใช้งานฐานข้อมูลตรงส่วนนี้ แต่หากมองในมุมของ IoT ตัว Firestore ถือว่าเป็นตัวกลางการเชื่อมต่อทุกอุปกรณ์เข้าด้วยกันได้ โดยมีจุดเด่นคือ เรียลไทม์ และสามารถบันทึกข้อมูลไว้ได้

ในด้านของ API ตัว Firestore ไม่ได้อิงการใช้งานไปกับภาษาใดภาษาหนึ่ง กรณีที่ภาษาใด ๆ ไม่มีไลบรารีให้ใช้งาน สามารถใช้ REST API (โปรโตคอล HTTP, HTTPS) ในการร้องขอข้อมูล (GET) หรือส่งข้อมูล (PUT) เข้าไปได้เลย

Firestore คือฐานข้อมูลประเภท NoSQL

ฐานข้อมูล MySQL MSSQL และฐานข้อมูลชนิด RDBMS ต่าง ๆ จะมีลักษณะเป็นตารางข้อมูล มีคอลัมน์ มีการกำหนดชนิดของข้อมูลไว้อย่างชัดเจน และใช้ภาษา SQL ในการติดต่อ

เฉพาะข้อมูลที่ต้องการได้ด้วยการใช้ WHERE และบางครั้งมีปัญหาเรื่องช่องโหว่ (SQL Injection ถือเป็นวิธีพื้นฐานที่นิยมใช้และได้ผลมากที่สุดขณะนี้)

ฐานข้อมูลชนิด NoSQL จะไม่ใช่ภาษา SQL ในการจัดการข้อมูล และออกแบบให้มีความยืดหยุ่น และเน้นความเร็วในการใช้งานมากที่สุด ฐานข้อมูล NoSQL ที่นิยมใช้งานในปัจจุบันคือ MongoDB ซึ่งมีการเก็บข้อมูลเป็นชนิด JSON (เจสัน) มีตารางเหมือนเดิม แต่ไม่มีคอลัมน์ข้อมูลที่ตายตัว ใน 1 แถว สามารถเก็บข้อมูลได้ทั้งข้อความ (String) ตัวเลข (Number) และอื่น ๆ รวมไปถึงอาร์เรย์ และออบเจกต์

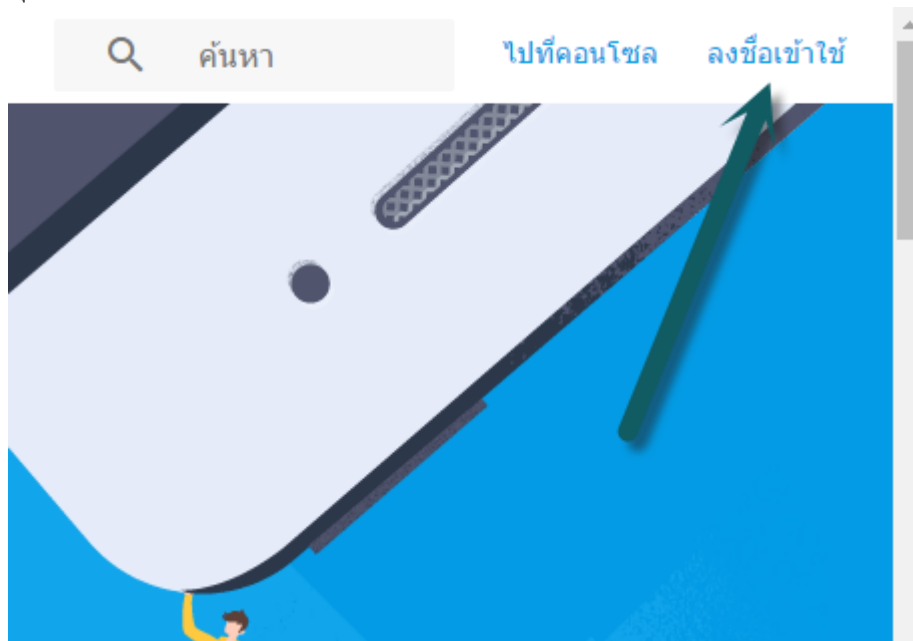
Firebase มีการทำงานคล้าย ๆ กับ MongoDB คือมีฐานข้อมูล แต่ไม่มีตาราง มีการเก็บข้อมูลในรูป JSON สามารถเพิ่มข้อมูลไปในออบเจกต์ใด ๆ ก็ได้ แต่เก็บเป็นอาร์เรย์ไม่ได้ ถ้าต้องการเพิ่มข้อมูลแบบอาร์เรย์ จะต้องใช้การ PUT ข้อมูลเข้าไปต่อท้ายเรื่อย ๆ ซึ่งจะมี Key ที่ Firebase สร้างให้เป็นตัวอ้างอิง

Firebase กับ API เพื่ออุปกรณ์ IoT

Firebase มี API ของหลายภาษาให้เลือกใช้งาน ทั้งภาษา Python (นิยมใช้ใน Embedded OS) JavaScript (ในบนหน้าเว็บไซต์) และรวมไปถึงใน ESP8266 ที่ใช้ Arduino IDE ด้วย Google ได้จัดทำไลบรารี Firebase สำหรับ Arduino (ซึ่งใช้กับ ESP8266 ได้) ไว้บน GitHub ส่วน API Reference สามารถอ่านได้ที่ [ลิงก์นี้](#)

เตรียม Firebase ให้พร้อมใช้งาน

เข้าไปที่หน้าเว็บไซต์หลักของ Firebase <https://firebase.google.com/> กด [ลงชื่อเข้าใช้](#) ที่มุมขวาบนของหน้าเว็บไซต์



เนื่องจาก Firebase เป็นของ Google จึงต้องใช้บัญชี Google ในการเข้าสู่ระบบ กรอกรหัสและลงชื่อเข้าใช้ให้เรียบร้อย



บัญชีเดียว กับทุกบริการของ Google

ลงชื่อเข้าใช้เพื่อไปยัง Google Developers



Sonthaya Nongnuch

รหัสผ่าน

ลงชื่อเข้าใช้

[ความช่วยเหลือ](#)

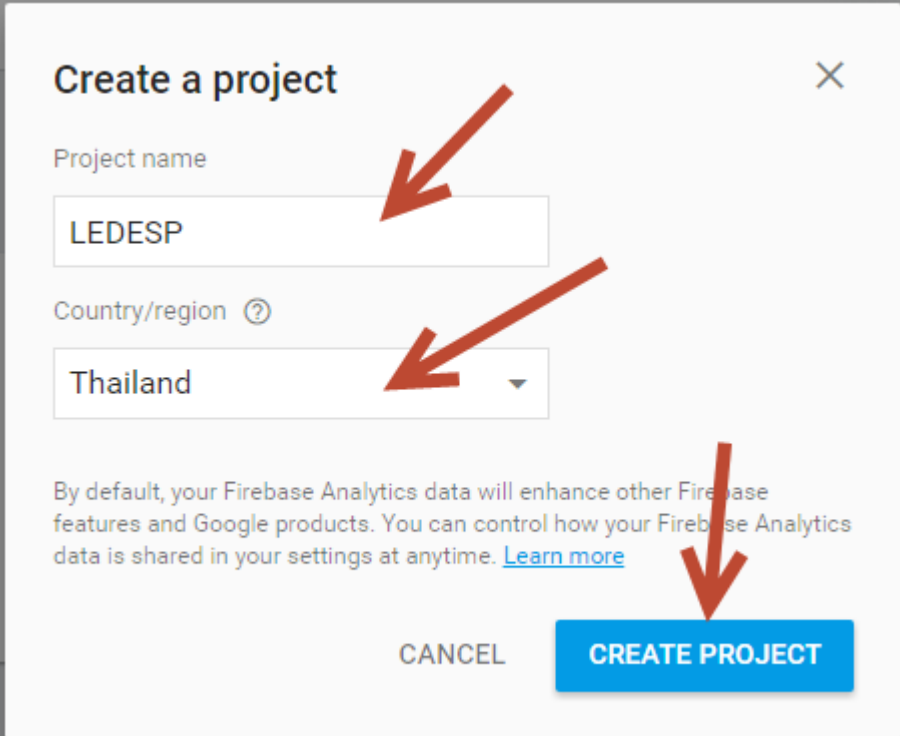
จากนั้นจะด้งกลับมาที่หน้าหลัก ให้กด ไปที่คอนโซล ในเมนูมุมขวบน



กดปุ่ม CREATE NEW PROJECT



ในช่องแรกให้ตั้งชื่อโปรเจกต์ใหม่ และในช่องที่ 2 เลือกเป็น Thailand แล้วกด CREATE PROJECT



Create a project

Project name

LEDESP

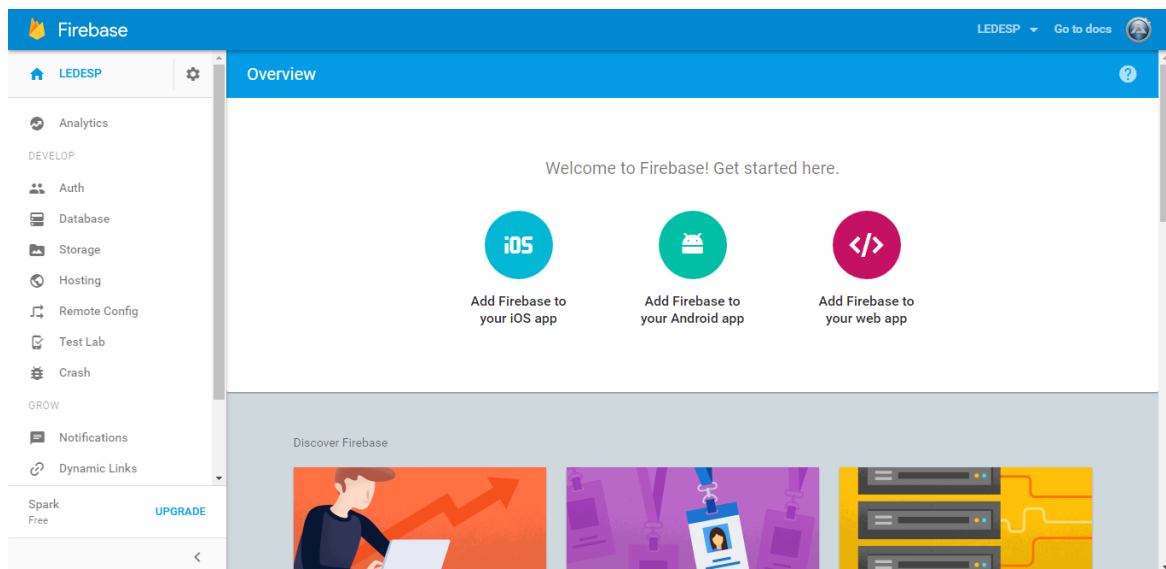
Country/region ?

Thailand

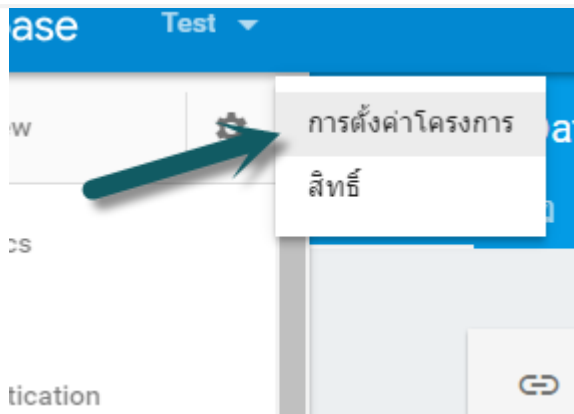
By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime. [Learn more](#)

CANCEL CREATE PROJECT

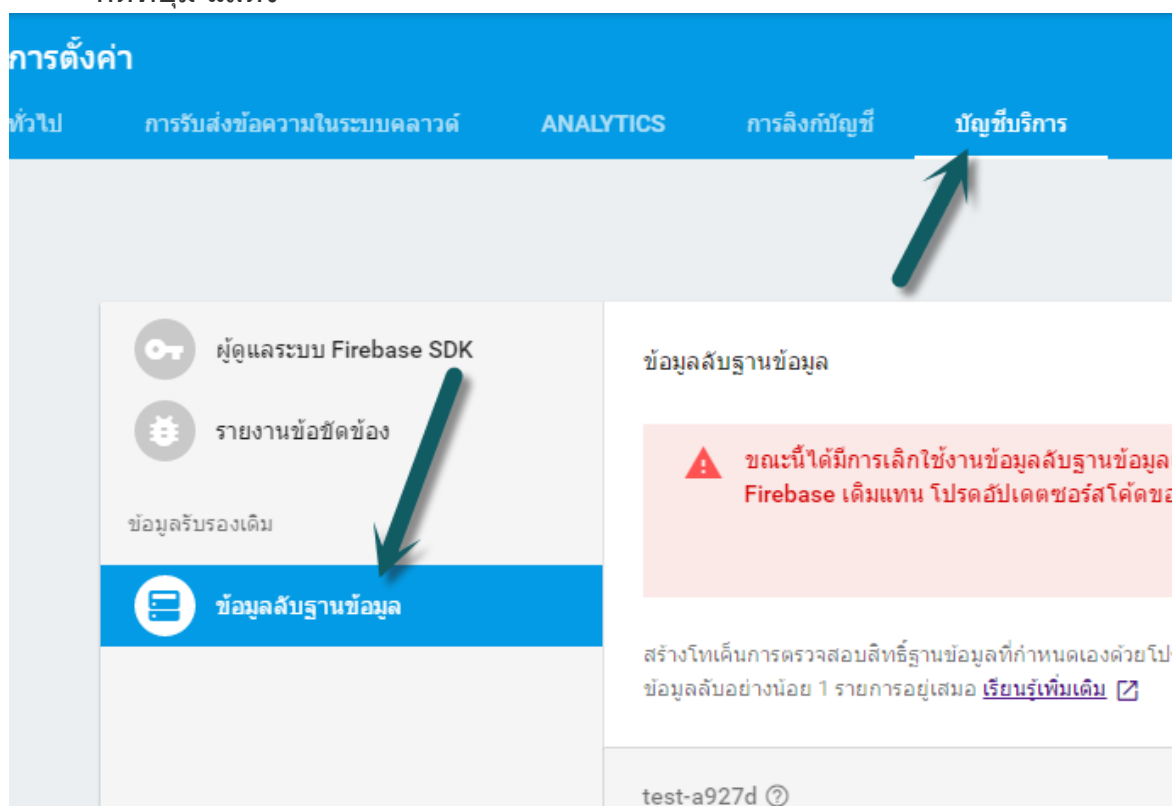
หน้าเว็บจะได้เรีคมาที่หน้าหลักของโปรเจกต์



กดไปที่รูปเฟืองมุมซ้ายบน แล้วคลิกที่เมนู การตั้งค่าโครงการ



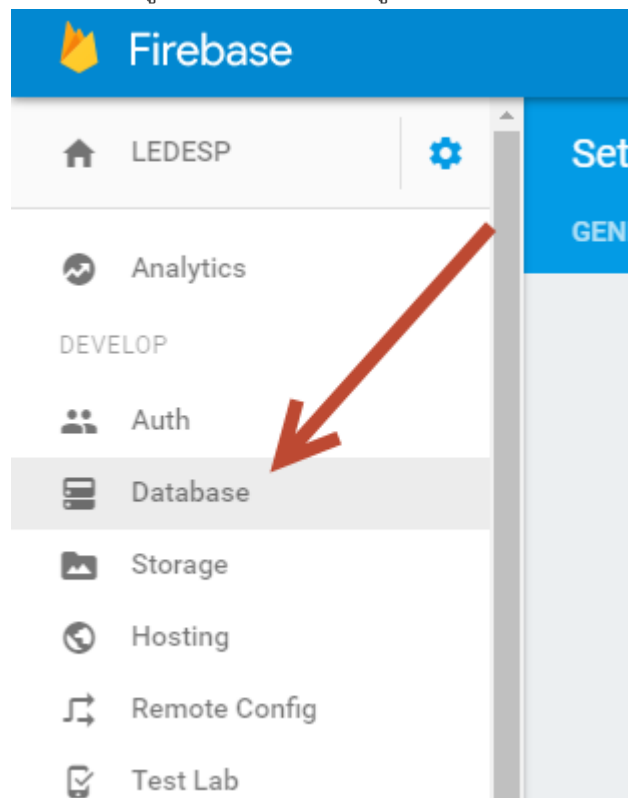
คลิกที่แท็บ ปัญหาบริการ ในเมนูด้านซ้าย เลือก ข้อมูลพื้นฐานข้อมูล นำเมาส์เลื่อนไปตรงช่อง กดที่ปุ่ม แสดง



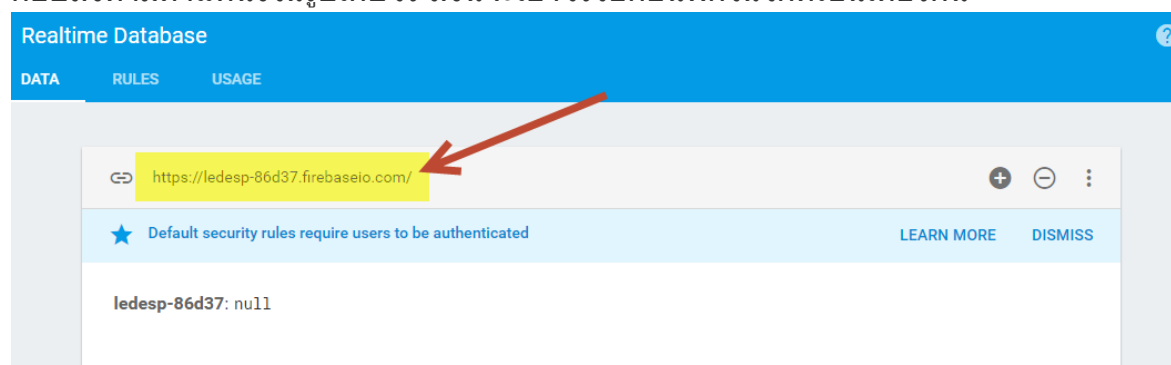
คัดลอก Key เก็บไว้ ตัว Key นี้จะใช้ในการคอนฟิกในโค้ด เพื่อให้สามารถเข้ามาแก้ไขข้อมูลในฐานข้อมูล Firebase ได้



กดไปที่เมนู Database ในเมนูด้านซ้าย



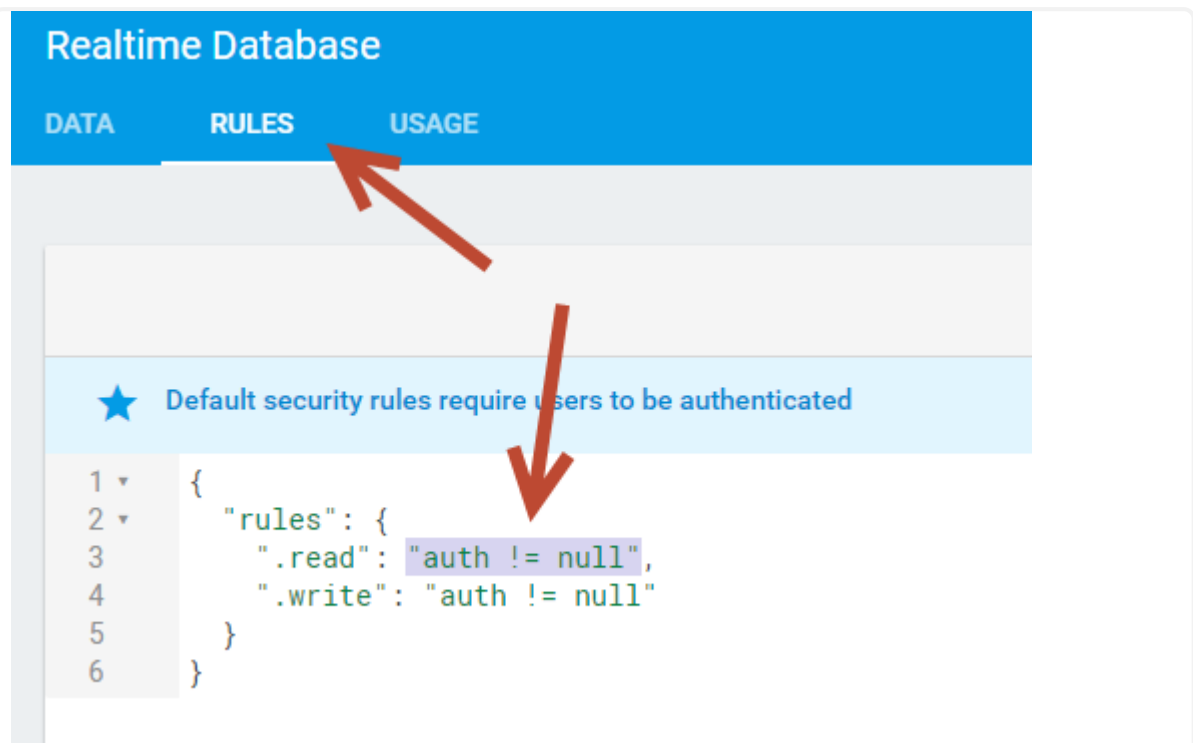
ก็อปลิงก์ตามตำแหน่งในรูปเก็บไว้ ลิ้งนี้จะเอาไว้ใช้คอนฟิกในโค้ดเช่นเดียวกัน



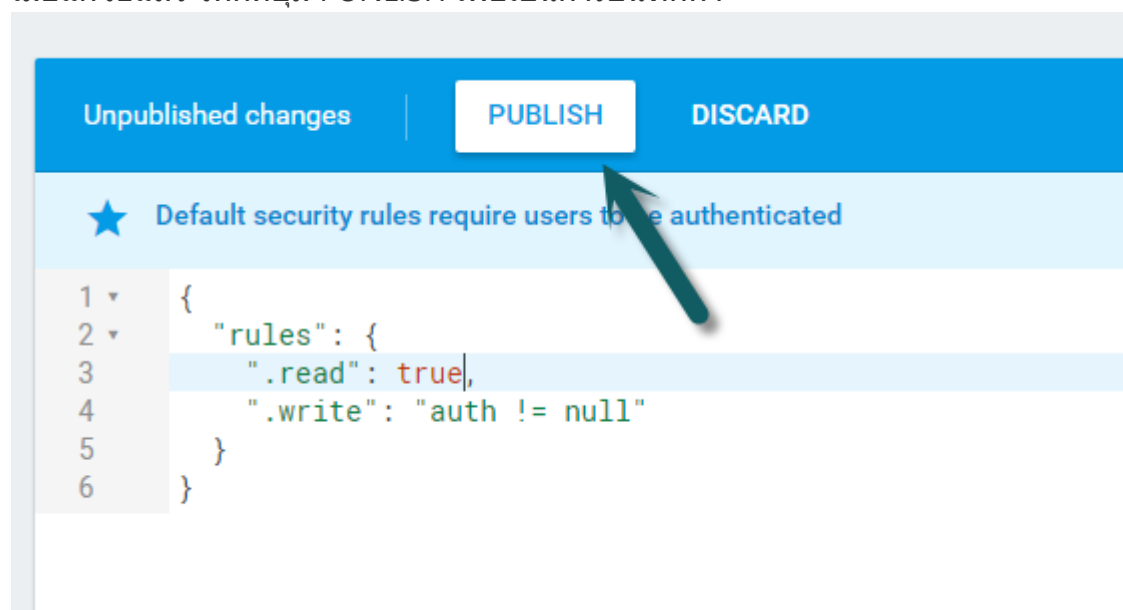
การกำหนดสิทธิ์การเข้าถึงข้อมูล

ส่วนนี้เป็นส่วนเสริม และมีความสำคัญหากนำไปใช้งานจริง เนื่องจาก Firebase ได้กำหนดค่าเริ่มต้นให้ต้องมี Key เท่านั้น จึงจะสามารถอ่าน - เขียนฐานข้อมูลได้ ซึ่งจะมีปัญหาหากนำไปใช้งานบนหน้าเว็บไซต์ เพราะในหน้าเว็บจะไม่สามารถเปิดเผย Key ได้ การเปิดเผย Key จะทำให้ Users สามารถเข้ามาแก้ไขข้อมูลในฐานข้อมูลได้ ดังนั้นจึงควรกำหนดให้ในการเขียนข้อมูลลงไปฐานข้อมูล สามารถทำได้เฉพาะมี Key และหากมี หรือไม่มี Key จะอ่านได้ ซึ่งสามารถกำหนดได้ดังนี้

กดไปที่แท็บ RULES ในส่วนที่คลุมแถบด้านล่างในรูป ให้ใส่ true ลงไปแทน



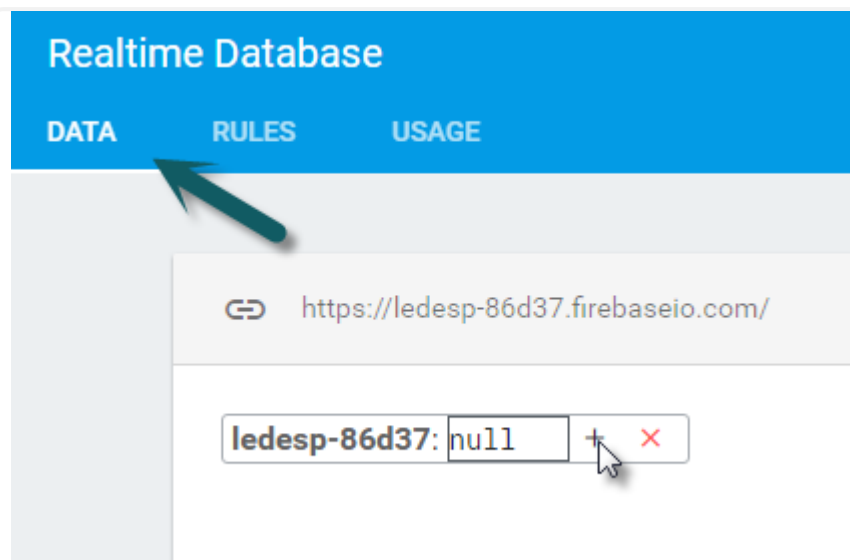
เมื่อแก้ไขแล้ว ให้กดปุ่ม PUBLISH เพื่อเป็นการบันทึกค่า



แค่นี้ฐานข้อมูล Firebase ก็พร้อมใช้งานแล้วครับ

การใช้งานคอนโซลของ Firebase

กดกลับมาที่แท็บ DATA นำเมาส์เลื่อนไปที่ชื่อแอปเจ็คแรก เครื่องหมาย + จะปรากฏขึ้นมา



กดที่เครื่องหมาย + จะมีชื่อขึ้นมา 2 ช่อง และเยื้องทางซ้ายเล็กน้อย การเยื้องทางซ้ายเพื่อให้ทราบได้ว่าเรากำลังจะเพิ่มข้อมูลลงในออปเจ็คใด ออปเจ็คที่อยู่ในระดับเดียวกันจะมีการเยื้องซ้ายเท่ากัน

ในช่อง Name จะเป็นชื่อของออปเจ็ค (Index, Key name) ส่วน Value ทดลองใส่ Hello !!! แล้วกดปุ่ม ADD



จากนั้นในระดับที่สูงกว่าทั้งหมดจะเปลี่ยนสีเป็นสีส้ม และออปเจ็คที่เพิ่มเข้ามาใหม่ (ในที่นี้คือ text) จะเป็นสีเขียว เมื่อผ่านไปช่วงเวลาหนึ่งสีจะกลับมาเป็นสีดำนั่งเดิม



ที่แอปเจ็คแรกสุด (อยู่สูงที่สุด) กด + อีกครั้ง พิมพ์ Name เป็น subObj แล้วกดเครื่องหมาย + ตามลูกศรชี้ในรูป



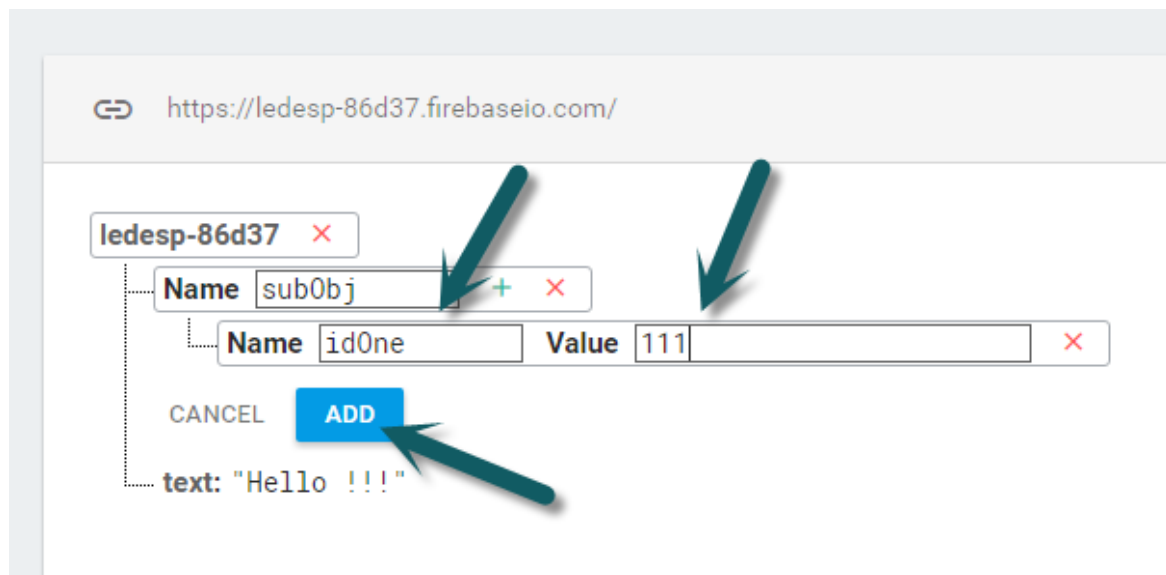
ledesp-86d37 ✕

Name subObj Value value + ✕

CANCEL ADD

text: "Hello !!!"

กรอก Name เป็น idOne แล้วกด Value เป็น 111 จากนั้นกด ADD



ledesp-86d37 ✕

Name subObj + ✕

Name idOne Value 111 ✕

CANCEL ADD

text: "Hello !!!"

จะเห็นได้ว่าผลคือ แอปเจ็ค ledesp-86d37 จะอยู่ในระดับสูงที่สุด และมีแอปเจ็ค subObj เป็นแอปเจ็คย่อย แล้วมี idOne ซึ่งมีค่าเป็น 111 เป็นแอปเจ็คย่อยของ subObj อีกที่ หากนำไปเทียบกับเนื้อหาในวิชา โครงสร้างข้อมูล ซึ่งเป็นวิชาพื้นฐานที่โปรแกรมเมอร์จะต้องเรียน จะพบว่าการทำงานแบบนี้จะคล้าย ๆ กับ [โครงสร้างข้อมูลแบบต้นไม้](#) ให้มอง ledesp-86d37 เป็นโน้ตแรก มอง subObj เป็นโน้ตลูกของโน้ต edesp-86d37 และโน้ต idOne เป็นโน้ตลูกของโน้ต subObj อีกที่ ซึ่งในโน้ตใบแต่ละโน้ต (โน้ตที่ไม่มีโน้ตลูก) สามารถมีค่าเป็นข้อความ ตัวอักษร หรือตัวเลขก็ได้

<https://ledesp-86d37.firebaseio.com/>

ledesp-86d37

```
- subObj
  - idOne: 111
  - text: "Hello !!!"
```

นำเมาส์เลื่อนไปตรงออปเจ็คที่ชื่อ text แล้วคลุมแถบดำ ทดลองแก้เป็น 1212312121 แล้วกดปุ่ม Enter บนคีย์บอร์ด

<https://ledesp-86d37.firebaseio.com/>

ledesp-86d37

```
- subObj
  - idOne: 111
  - text: "Hello !!!" ✕
```

<https://ledesp-86d37.firebaseio.com/>

ledesp-86d37

```
- subObj
  - idOne: 111
  - text: 1212312121 ✕
```

ผลที่ได้คือออปเจ็คที่ถูกแก้ไขจะเปลี่ยนเป็นสีส้ม แล้วออปเจ็คที่อยู่สูงกว่าทั้งหมด ก็เปลี่ยนเป็นสีส้มเช่นกัน

<https://ledesp-86d37.firebaseio.com/>

ledesp-86d37

subObj

idOne: 111

text: 1212312121

นำเมาส์ไปวางที่ออปเจ็ค subObj จากนั้นกด x

<https://ledesp-86d37.firebaseio.com/>

ledesp-86d37

subObj + x

idOne: 111

text: 1212312121

จะแจ้งว่าหากลบออปเจ็คนี้ไปแล้ว ออปเจ็คที่อยู่ล่าง ๆ ก็จะถูกลบไปด้วย ให้กดปุ่ม DELETE

Delete data



All data at this location, including nested data, will be permanently deleted

Data location

/subObj

To delete data without seeing this message,
hold shift + click when deleting a node.

CANCEL

DELETE

จากนั้นออปเจ็คตั้งแต่ subObj ลงไปจะเปลี่ยนเป็นสีแดง หมายถึงออปเจ็คนั้นไม่มีอยู่แล้วนั่นเอง และออปเจ็คที่อยู่สูงกว่าจะเปลี่ยนเป็นสีส้ม หมายถึงมีการอัปเดตข้อมูลนั่นเอง

<https://ledesp-86d37.firebaseio.com/>

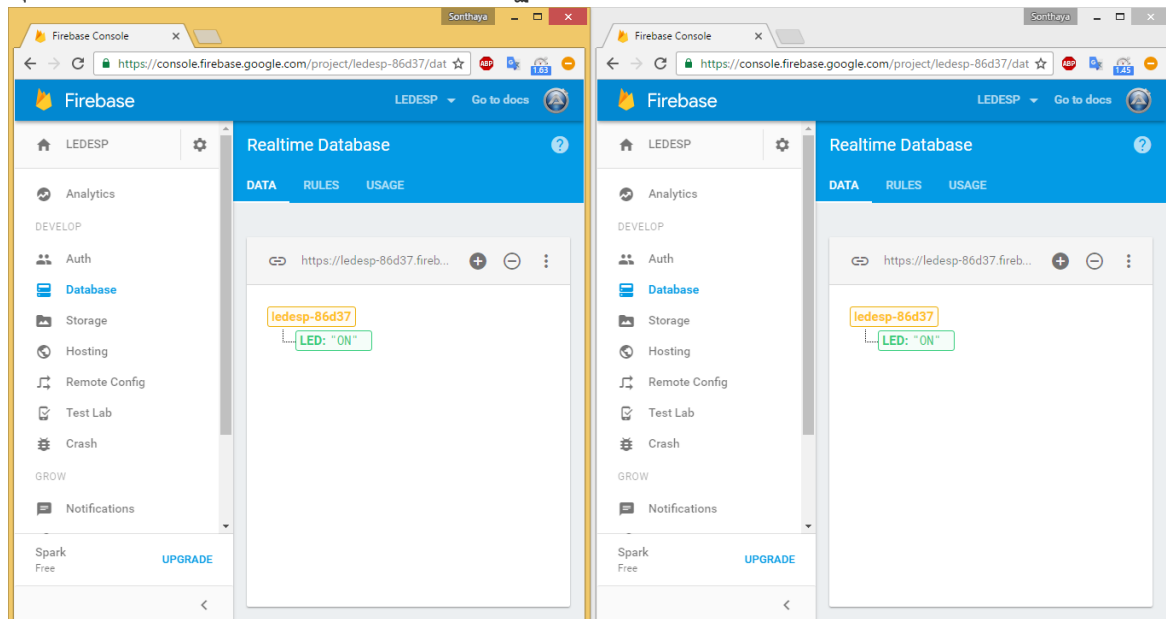


การใช้งานคอมโซลก็มีเพียงเท่านี้ ในตอนแก้ไขขอให้แก้ไขแล้วกด Enter ทันที เพราะหากไม่ทำการกด Enter ข้อมูลจะไม่ถูกแก้ไข และต้องเข้าใจไว้เสมอว่าหากลบออปปเจ็คที่อยู่ในระบบดับสูง ๆ ก็จะทำให้ออปปเจ็คระดับต่ำลงมากถูกลบไปด้วย และไม่สามารถกู้คืนกลับมาได้

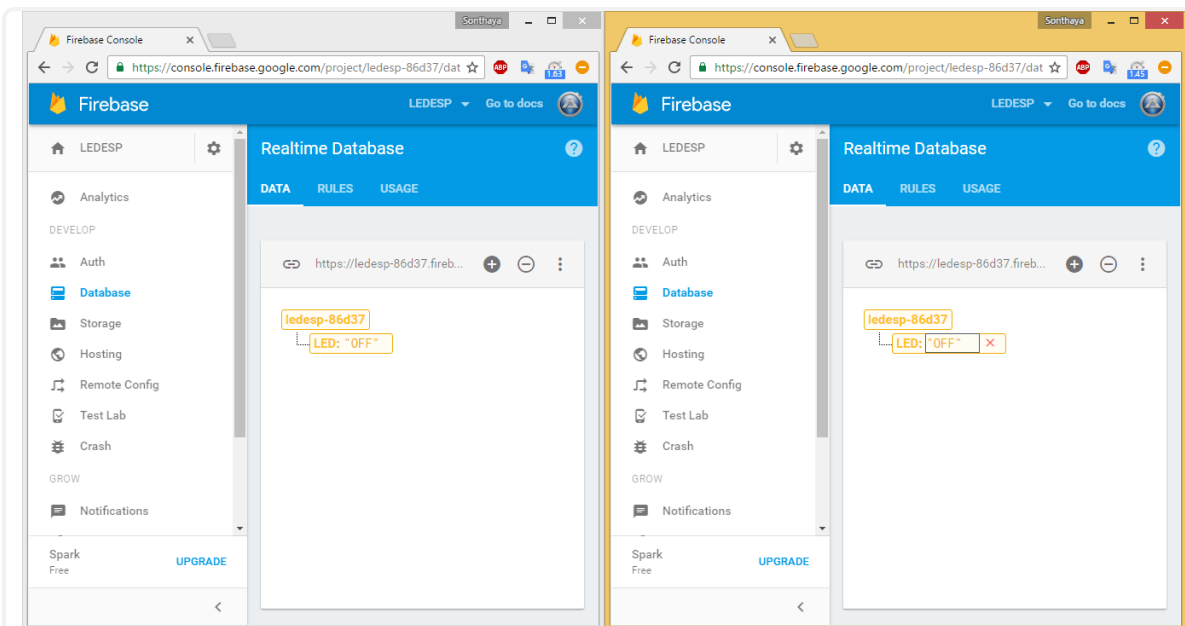
ทำความเข้าใจกับ Realtime Database

Realtime Database หมายถึงฐานข้อมูลที่มีเมื่อมีการอัปเดตข้อมูล จะมีการส่งข้อมูลที่อัปเดตนี้ ไปที่อุปกรณ์อื่น ๆ ที่เชื่อมต่ออยู่ด้วย ดังตัวอย่างด้านล่างนี้

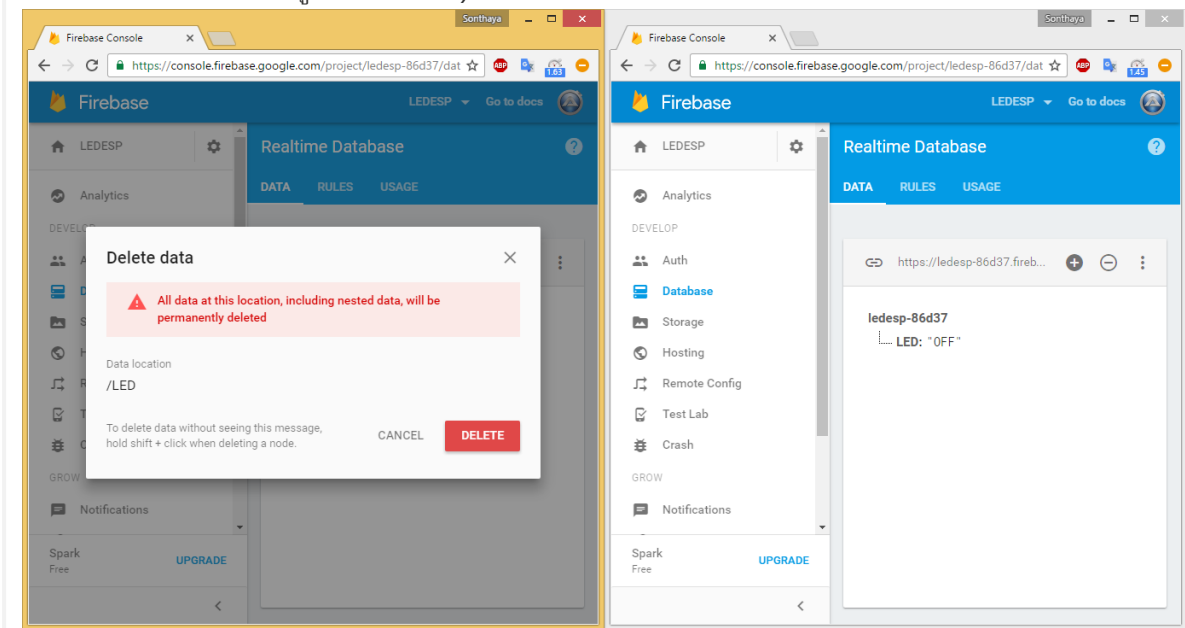
ในรูปด้านล่างนี้ ทางซ้ายผมได้เพิ่มออปปเจ็คที่มี Key เป็น LED และมีค่าเป็น ON เมื่อผมกดที่ปุ่ม ADD จะมีค่าของ LED: ON ไปปรากฏในทั้ง 2 หน้าทันที

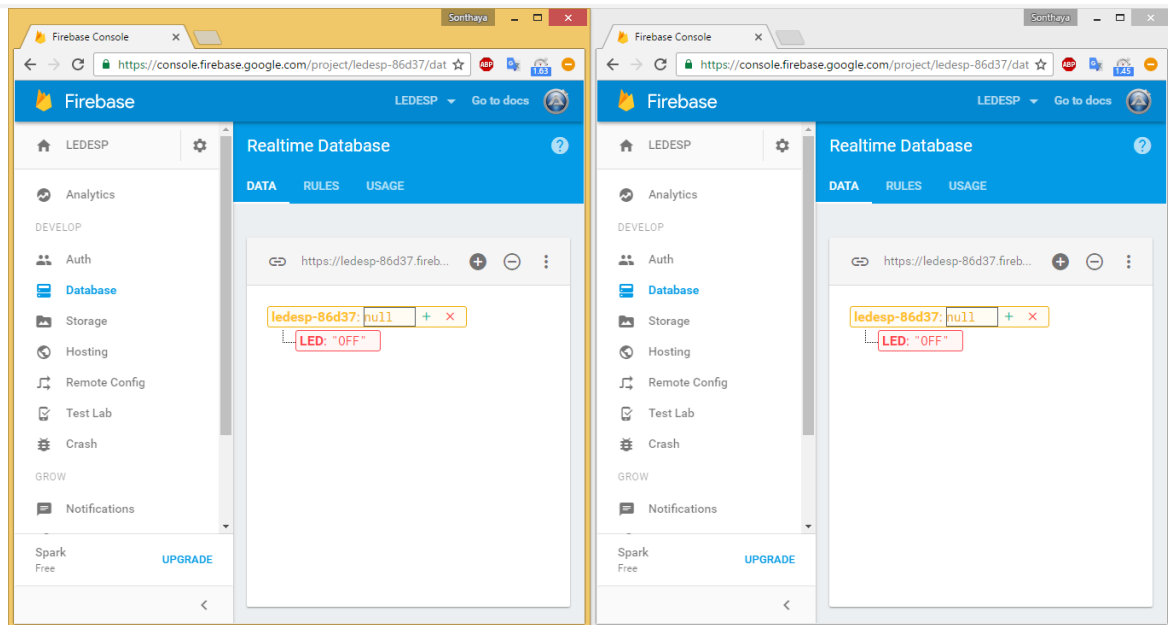


ต่อมาผมได้แก้ไขข้อมูลของออปปเจ็ค LED ในหน้าเว็บทางขวาเป็น OFF ผลที่ได้คือในหน้าต่างทั้ง 2 ด้านมีการเปลี่ยนแปลงทันที



สุดท้ายผมกดลบออกเจ็ด LED ออกไป ผลคือลบเจ็ด LED หายไปทันทีจากทั้ง 2 หน้า (สี
แดงหมายถึงลบออกแล้ว)



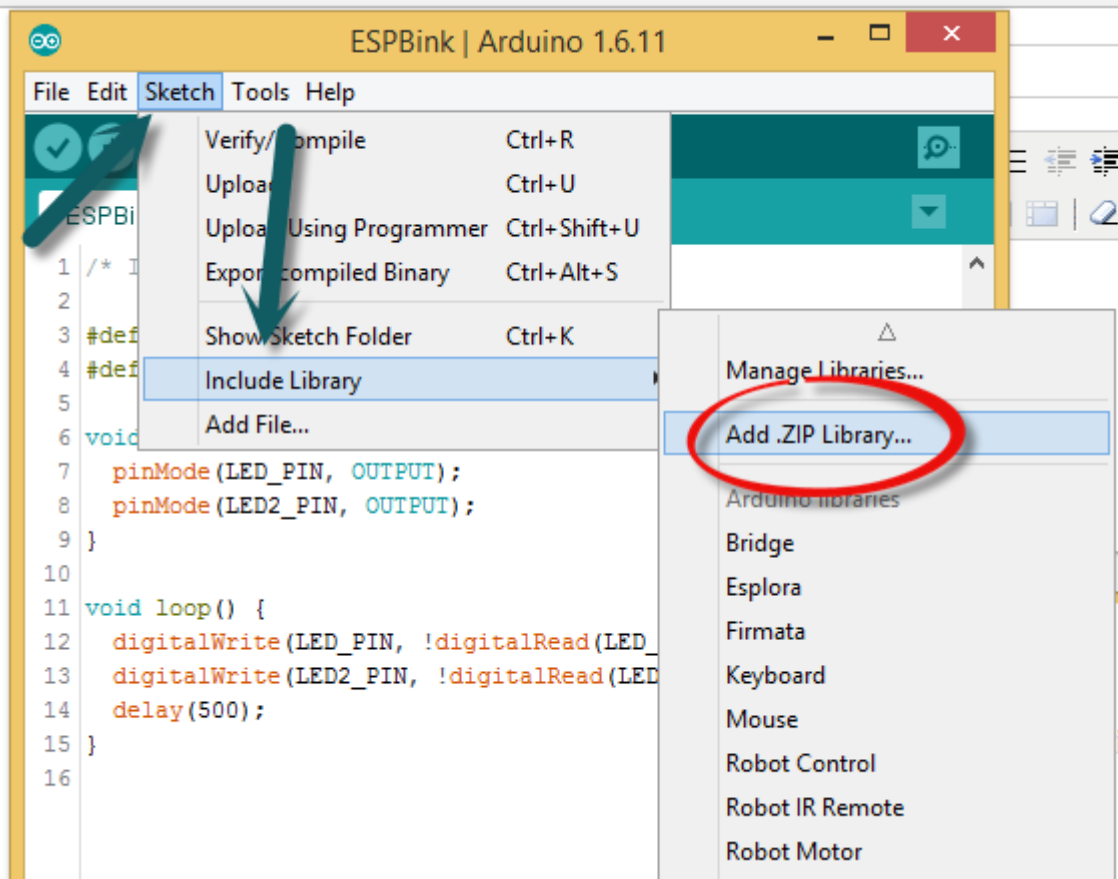


Firebase กับ ESP8266 ESP8285

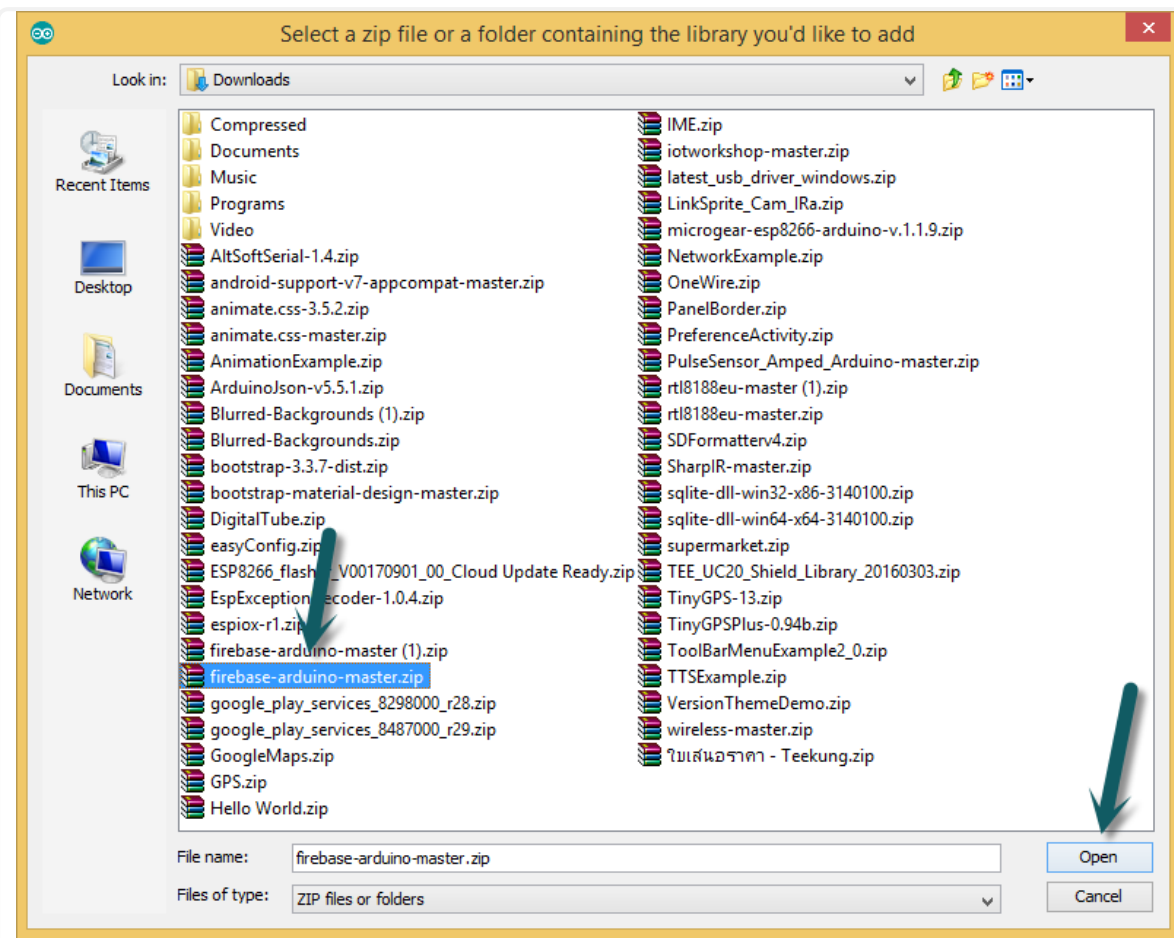
ทดลองให้ ESP8266 อัปเดตข้อมูลในฐานข้อมูลของ Firebase
ก่อนอื่นให้ลงไลบรารีก่อน โดยดาวน์โหลด

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

เปิดโปรแกรม Arduino ขึ้นมา กดไปที่ Sketch > Include Library > Add .ZIP Library...



เลือกไฟล์ที่ได้ดาวน์โหลดไว้ กดปุ่ม Open



ก๊อบโค้ดด้านล่างนี้ลงโปรแกรม Arduino

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

```
#include <FirebaseArduino.h>
```

```
// Config Firebase
```

```
#define FIREBASE_HOST "ledesp-86d37.firebaseio.com"
```

```
#define FIREBASE_AUTH "<YOUR KEY>"
```

```
// Config connect WiFi

#define WIFI_SSID "<YOUR WIFI NAME>"

#define WIFI_PASSWORD "<YOUR WIFI PASSWORD>"


int i = 0;


void setup() {

  Serial.begin(9600);


  WiFi.mode(WIFI_STA);


  // connect to wifi.

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("connecting");


  while (WiFi.status() != WL_CONNECTED) {

    Serial.print(".");
```



```
delay(500);

}

Serial.println();

Serial.print("connected: ");

Serial.println(WiFi.localIP());


Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

}


void loop() {

  Firebase.setInt("number", i);

  if (Firebase.failed()) {

    Serial.print("set /number failed:");

    Serial.println(Firebase.error());

    return;

  }
```

```
Serial.print("set /number to ");
```

```
Serial.println(Firebase.getInt("number"));
```

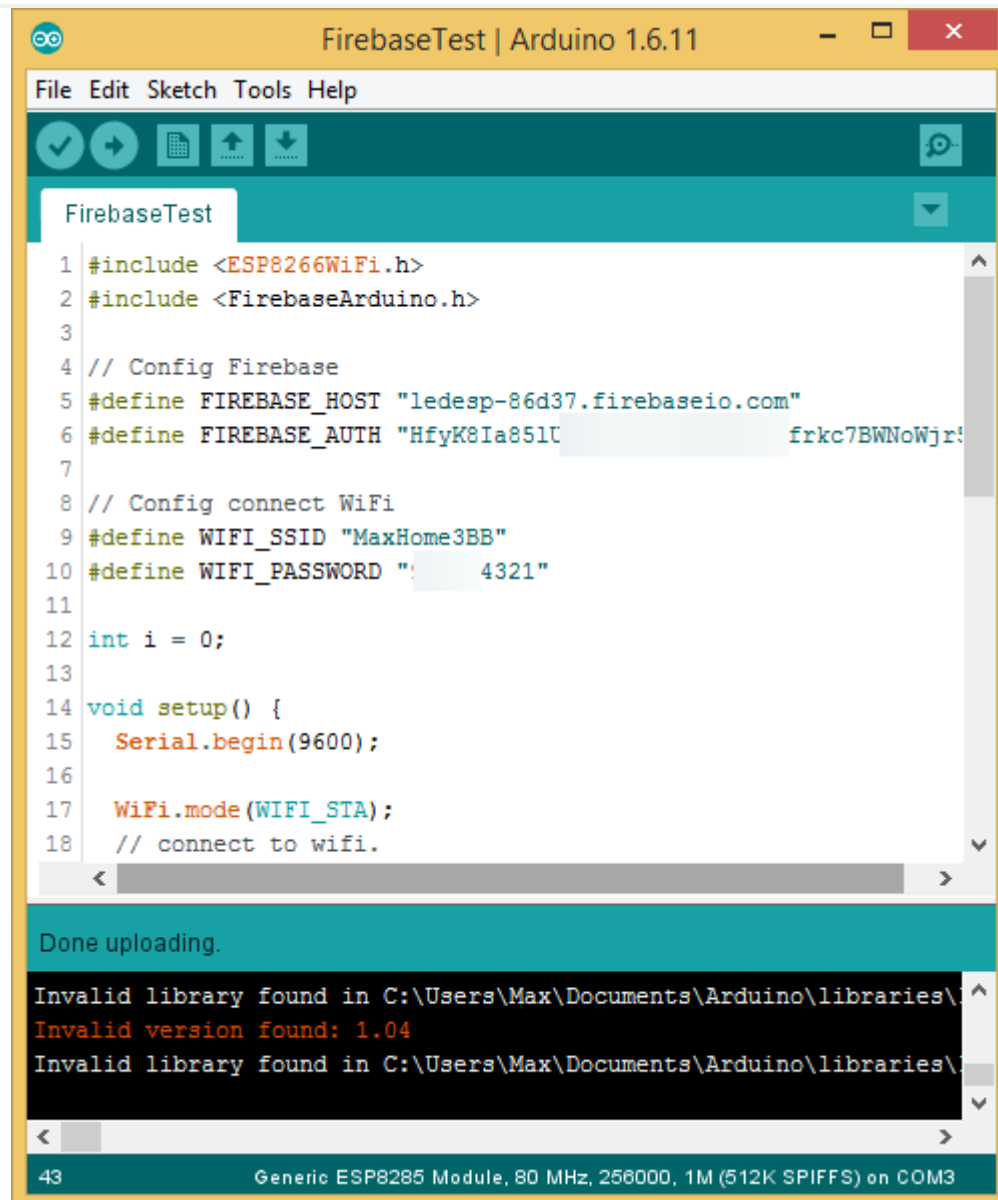
```
i++;
```

```
delay(500);
```

```
}
```

[FirebaseTest1.ino](#) hosted with ❤ by [GitHub](#) [view raw](#)

แก้ไขข้อมูลของ FIREBASE_HOST, FIREBASE_AUTH, WIFI_SSID และ WIFI_PASSWORD
ให้ถูกต้อง ดังตัวอย่างด้านล่างนี้



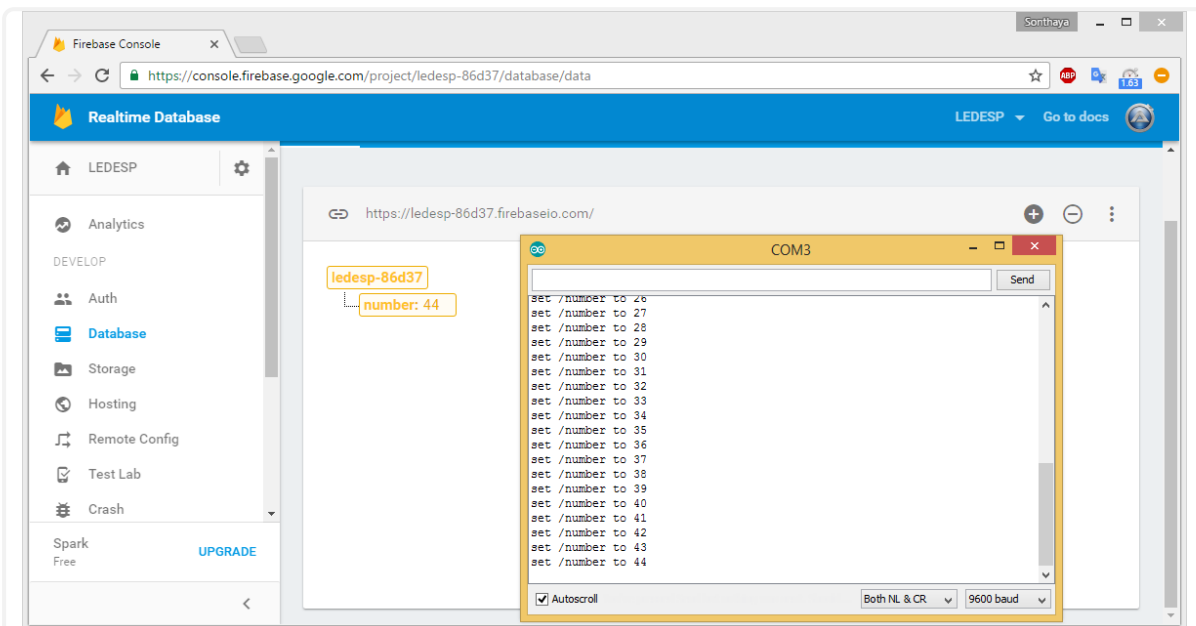
```
1 #include <ESP8266WiFi.h>
2 #include <FirebaseArduino.h>
3
4 // Config Firebase
5 #define FIREBASE_HOST "ledesp-86d37.firebaseio.com"
6 #define FIREBASE_AUTH "HfyK8Ia85lUfrkc7BWN0Wjr!"
7
8 // Config connect WiFi
9 #define WIFI_SSID "MaxHome3BB"
10 #define WIFI_PASSWORD "1234567890"
11
12 int i = 0;
13
14 void setup() {
15   Serial.begin(9600);
16
17   WiFi.mode(WIFI_STA);
18   // connect to wifi.
```

Done uploading.

Invalid library found in C:\Users\Max\Documents\Arduino\libraries\
Invalid version found: 1.04
Invalid library found in C:\Users\Max\Documents\Arduino\libraries\
Invalid version found: 1.04

43 Generic ESP8285 Module, 80 MHz, 256000, 1M (512K SPIFFS) on COM3

เมื่ออัปโหลดโค้ดเข้าไปแล้ว และรอจนกว่าจะเชื่อมต่อ WiFi เสร็จ ตัว ESP8266 / ESP8285 จะมีการเชื่อมต่อปลั๊ก number ให้มีค่าเพิ่มขึ้นเรื่อย ๆ ทุก ๆ ครึ่งวินาที หากเปิดหน้าคอนโซล ร่วมกับหน้ามอนิเตอร์จะเห็นผลที่ชัดเจน



ทำความเข้าใจกับฟังก์ชันต่าง ๆ

คำสั่งใช้ set - เป็นคำสั่งที่ใช้เซตค่าในออปเจ็กต์ต่าง ๆ เช่น เซตให้ออปเจ็กต์ A มีค่าเป็น 10 (Int)

การใช้งานอยู่ในรูปแบบ `Firebase.set???(const String &path, ??? value)` ซึ่ง ??? สามารถเป็นไปได้นี้

???=	รายละเอียด	ตัวอย่างการใช้
Int	ใช้นำตัวเลขจำนวนเต็มเข้าออปเจ็กต์	<code>Firebase.setInt("number", 10);</code>
Float	ใช้นำตัวเลขที่มีจุดทศนิยมเข้าออปเจ็กต์	<code>Firebase.setFloat("number", 2.56);</code>
Bool	ใช้นำค่า True หรือ False เข้าออปเจ็กต์	<code>Firebase.setBool("trueORfalse", true);</code>
String	ใช้นำข้อความเข้าออปเจ็กต์	<code>Firebase.setString("title", "Hello !");</code>

คำสั่งใช้ get - เป็นคำสั่งใช้อ่านค่าจากออปเจ็กต์ต่าง ๆ ออกมาในรูปชนิดข้อมูลต่าง ๆ

การใช้งานอยู่ในรูปแบบ `Firebase.get???(const String &path)` ซึ่ง ??? สามารถเป็นไปได้นี้

???=	รายละเอียด	ตัวอย่างการใช้
Int	อ่านค่าตัวเลขจำนวนเต็มจากออปเจ็กต์	<code>int number = Firebase.getInt("number");</code>
Float	อ่านค่าตัวเลขที่มีจุดทศนิยมจากออปเจ็กต์	<code>flost number = Firebase.getFloat("number");</code>
Bool	อ่านค่าค่า True หรือ False จากออปเจ็กต์	<code>bool trueORfalse = Firebase.getBool("trueORfalse");</code>
String	อ่านค่าข้อความจากออปเจ็กต์	<code>String title = Firebase.getString("title");</code>

คำสั่งใช้ push - เป็นคำสั่งที่ใช้เพิ่มข้อมูลเข้าไปต่อท้ายในออปเจ็ค โดยจะมี Key เป็นแบบ PRIMARY KEY คือชื่อ Key จะเป็นข้อความยาว ๆ และจะไม่มีซ้ำกัน

การใช้งานอยู่ในรูปแบบ `Firebase.push???(const String &path, ??? value)` ซึ่ง ??? สามารถเป็นไปได้ดังนี้

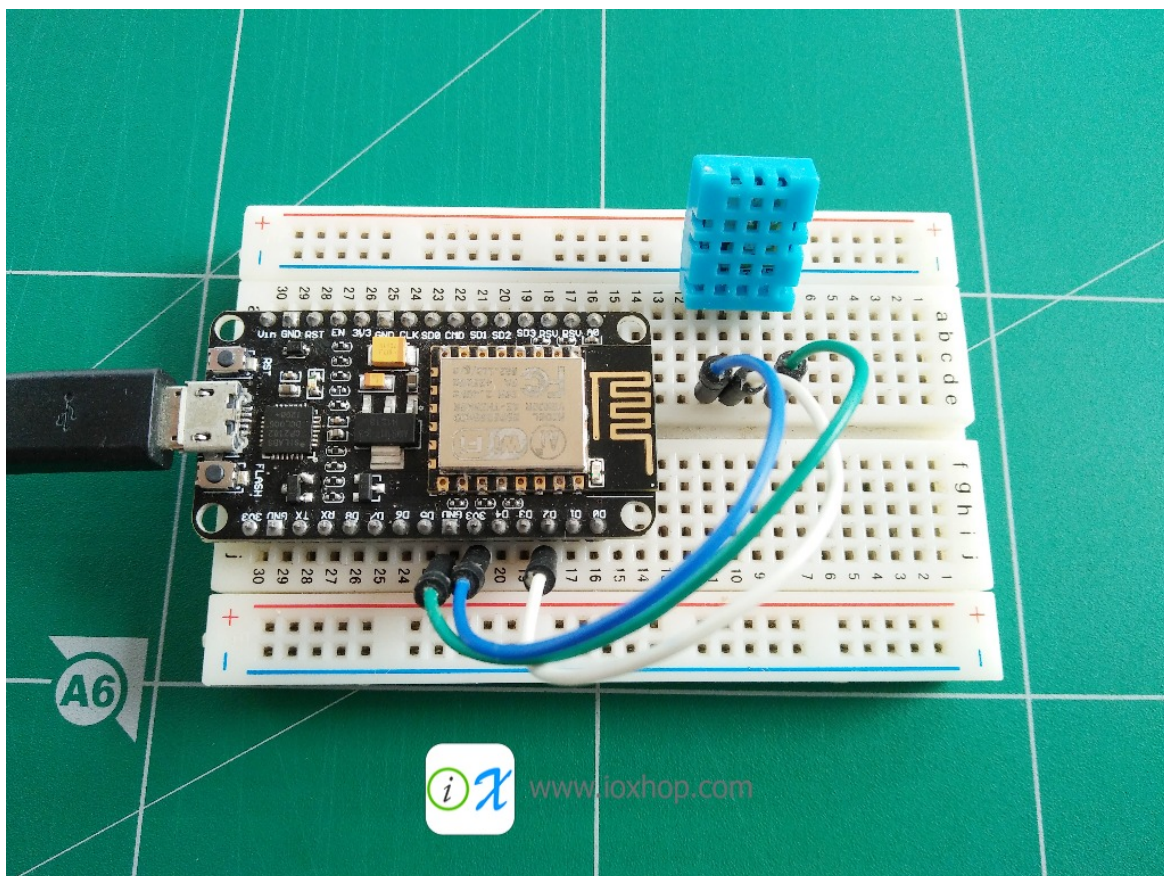
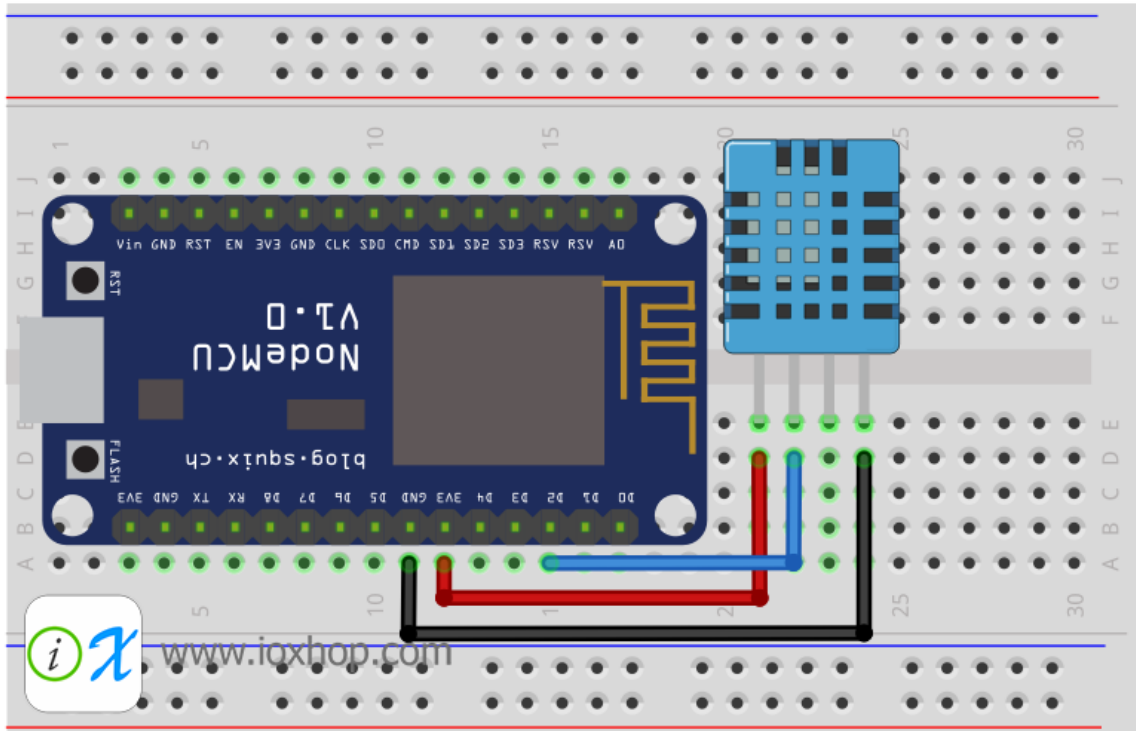
???=	รายละเอียด	ตัวอย่างการใช้
Int	ให้นำตัวเลขจำนวนเต็มเข้าไปต่อท้ายในออปเจ็ค	<code>Firebase.pushInt("arrNumber", 10);</code>
Float	ให้นำตัวเลขที่มีจุดทศนิยมเข้าไปต่อท้ายในออปเจ็ค	<code>Firebase.pushFloat("arrNumber", 2.56);</code>
Bool	ให้นำค่า True หรือ False เข้าไปต่อท้ายในออปเจ็ค	<code>Firebase.pushBool("arrTrueORfalse", true);</code>
String	ให้นำข้อความเข้าไปต่อท้ายในออปเจ็ค	<code>Firebase.pushString("arrTitle", "Hello !");</code>

ทดลองใช้คำสั่ง Push

ในตัวอย่างที่แล้วมีการใช้เพียงคำสั่ง set และ get ในตัวอย่างนี้จะเป็นการทดลองใช้คำสั่ง push กันครับ

คำสั่ง push เหมาะสำหรับการเก็บข้อมูลในลักษณะคล้ายตาราง โดยทำการเพิ่มแถวของตารางไปเรื่อย ๆ ภายในแถวของตารางสามารถมีคอลัมได้ไม่จำกัด แต่ต้องมีชื่อของ Key ตามข้อกำหนดการใช้ Firebase

ในตัวอย่างนี้ผมได้ต่อ DHT11 เข้ากับ ESP8285 ที่ขา GPIO4 เพื่ออ่านค่าอุณหภูมิ และได้เขียนโค้ดให้ทำการ push ค่าอุณหภูมิลงในออปเจ็ค temperature และมีการ push ค่าของความชื้นลงออปเจ็ค humidity



อัปเดตโค้ดด้านล่างนี้ลงบอร์ด อย่าลืมเปลี่ยน FIREBASE_HOST, FIREBASE_AUTH, WIFI_SSID และ WIFI_PASSWORD เหมือนเดิม กรณีต่อ DHT ไรท์ที่ขาอื่น สามารถเปลี่ยนได้ที่ DHTPIN และหากไม่ได้ใช้ DHT11 สามารถเปลี่ยนได้ที่ DHTTYPE

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

```
#include <FirebaseArduino.h>
```

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

```
// Config Firebase
```

```
#define FIREBASE_HOST "ledesp-86d37.firebaseio.com"
```

```
#define FIREBASE_AUTH "<YOUR KEY>"
```

```
// Config connect WiFi
```

```
#define WIFI_SSID "<YOUR WIFI NAME>"
```

```
#define WIFI_PASSWORD "<YOUR WIFI PASSWORD>"
```

```
// Config DHT
```

```
#define DHTPIN 4
```

```
#define DHTTYPE DHT11
```

```
String name;
```

```
DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
```

```
  Serial.begin(9600);
```

```
  WiFi.mode(WIFI_STA);
```

```
  // connect to wifi.
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
  Serial.print("connecting");
```

```
  while (WiFi.status() != WL_CONNECTED) {
```

```
    Serial.print(".");
```

```
    delay(500);
```

```
  }
```

```
  Serial.println();
```



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

Serial.println(WiFi.localIP());


Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

dht.begin();

}


void loop() {

// Read temp & Humidity for DHT22

float h = dht.readHumidity();

float t = dht.readTemperature();


if (isnan(h) || isnan(t)) {

Serial.println("Failed to read from DHT sensor!");

delay(500);

return;
```

```
}
```

```
// append a new value to /temperature
```

```
name = Firebase.pushFloat("temperature", t);
```

```
if (Firebase.failed()) {
```

```
Serial.print("pushing /temperature failed:");
```

```
Serial.println(Firebase.error());
```

```
return;
```

```
}
```

```
Serial.print("pushed: /temperature/");
```

```
Serial.println(name);
```

```
// append a new value to /temperature
```

```
name = Firebase.pushFloat("humidity", h);
```

```
if (Firebase.failed()) {
```

```
Serial.print("pushing /humidity failed:");
```

```
Serial.println(Firebase.error());

return;

}

Serial.print("pushed: /humidity/");

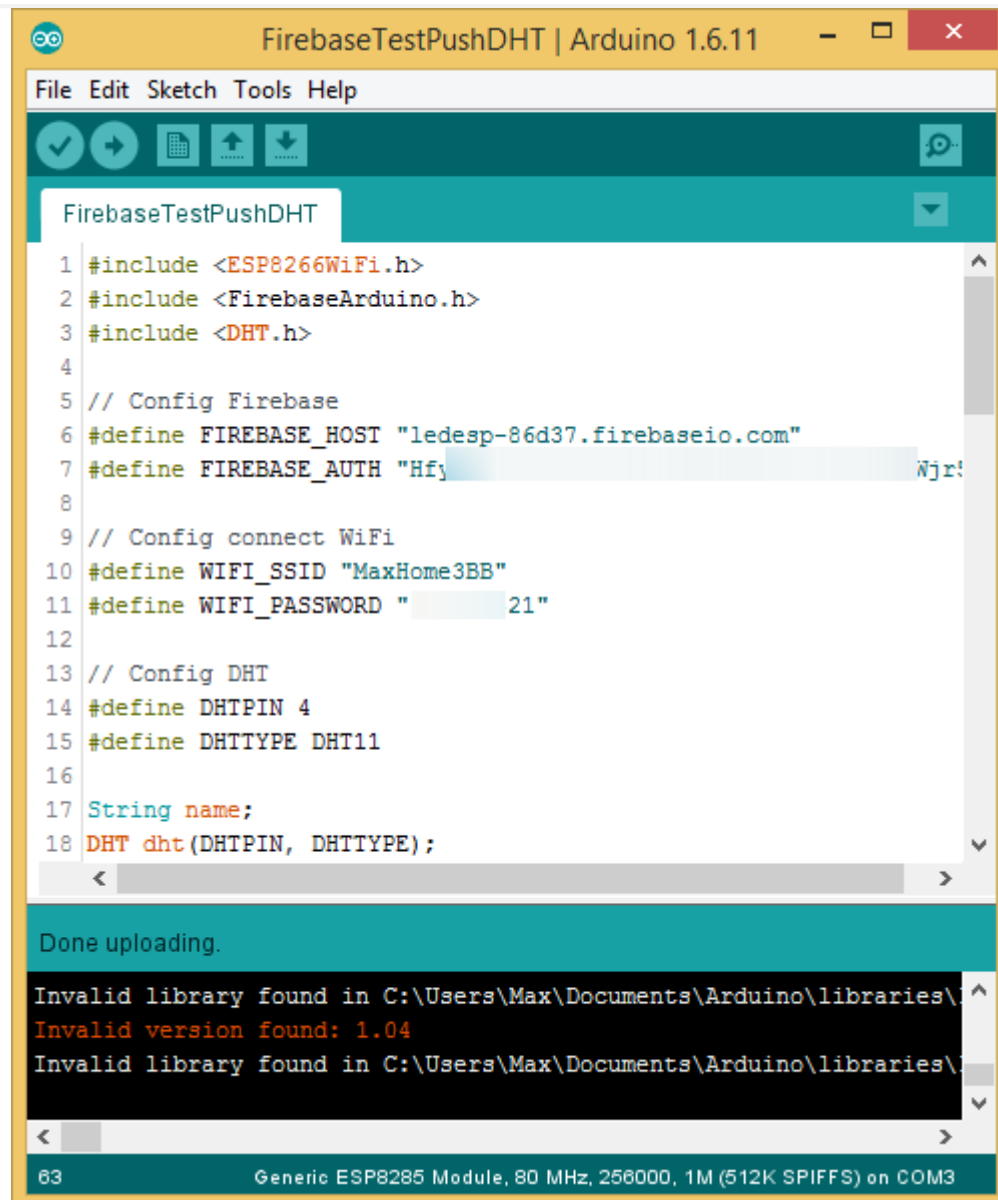
Serial.println(name);


delay(5000);

}
```

[FirebaseTestDHT.ino](#) hosted with ❤ by [GitHub](#)

[view raw](#)



The screenshot shows the Arduino IDE interface. The title bar reads "FirebaseTestPushDHT | Arduino 1.6.11". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for checking, running, uploading, and downloading. The sketch editor displays the following code:

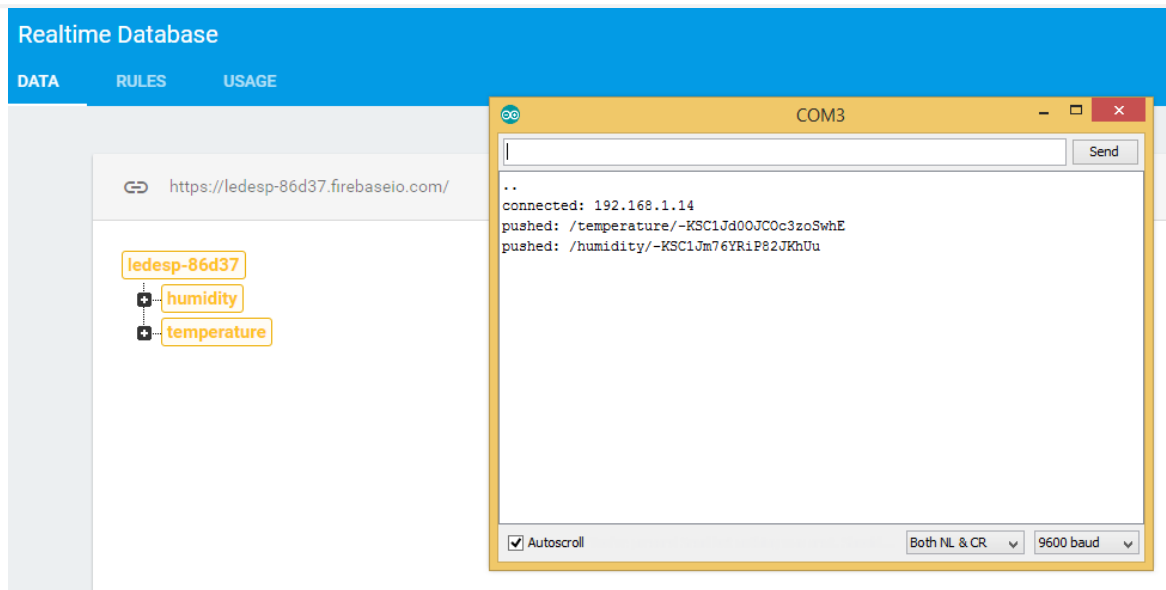
```
1 #include <ESP8266WiFi.h>
2 #include <FirebaseArduino.h>
3 #include <DHT.h>
4
5 // Config Firebase
6 #define FIREBASE_HOST "ledesp-86d37.firebaseio.com"
7 #define FIREBASE_AUTH "Hfy[redacted]Wjr!"
8
9 // Config connect WiFi
10 #define WIFI_SSID "MaxHome3BB"
11 #define WIFI_PASSWORD "21"
12
13 // Config DHT
14 #define DHTPIN 4
15 #define DHTTYPE DHT11
16
17 String name;
18 DHT dht(DHTPIN, DHTTYPE);
```

Below the code editor, a status bar indicates "Done uploading." and a message box displays the following error:

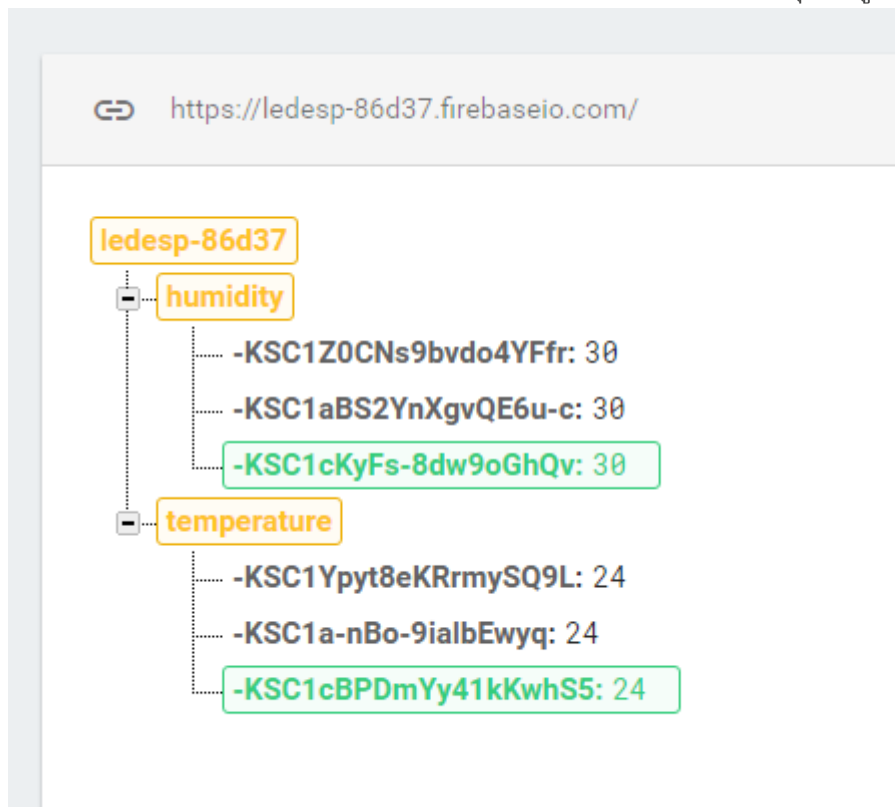
```
Invalid library found in C:\Users\Max\Documents\Arduino\libraries\
Invalid version found: 1.04
Invalid library found in C:\Users\Max\Documents\Arduino\libraries\
```

The bottom status bar shows "63 Generic ESP8285 Module, 80 MHz, 256000, 1M (512K SPIFFS) on COM3".

เมื่อเปิดหน้ามอเนเตอร์ขึ้นมา พร้อม ๆ กับหน้าคอนโซล จะเห็นว่าค่าอุณหภูมิ และค่าความชื้น ได้ถูกเพิ่มเข้ามาในทุก ๆ 5 วินาที สังเกตได้จากสีของชื่อออปเจ็คจะเปลี่ยนเป็นสีส้ม หมายถึงมีการเพิ่มข้อมูลเข้ามาใหม่นั้นเอง



หากกดที่รูปเครื่องหมาย + (หน้าชื่อออปเจ็ค) จะพบกับ Key ของออปเจ็คที่มีลักษณะเป็นข้อความไม่มีความหมาย และจะพบค่าของออปเจ็คเป็นตัวเลขอุณหภูมิ หรือความชื้น



ต่อมาเราจะมาลองเก็บอุณหภูมิ และความชื้นไว้ในออปเจ็คตัวเดียวกันครับ เนื่องจากข้อมูลทั้ง 2 ตัวนี้มาจากแหล่งเดียวกัน ดังนั้นมันควรจะอยู่ด้วยกันเสมอเพื่อความง่ายในการเรียกใช้ การที่จะทำให้ทั้งอุณหภูมิ และความชื้นมาอยู่ในออปเจ็คเดียวกันได้ จะต้องทำให้ข้อมูลทั้ง 2 นี้มารวมกันเสียก่อน โดยการสร้างออปเจ็คทั้ง 2 รอไว้โดยใช้ JSON แล้วค่อย Push ทั้ง 2 ออปเจ็คนี้ขึ้นไปพร้อมกัน จะทำให้ทั้ง 2 ออปเจ็คมีออปเจ็คที่สูงกว่าเป็น Key เดียวกัน อธิบายอย่างเดียวอาจจะไม่เห็นภาพ มาลองทำกันเลยดีกว่าครับ ก๊อบโค้ดด้านล่างนี้ไปลองอัปโหลดเข้า ESP8266 ESP8285 ได้เลย (อย่าลืมแก้ไขข้อมูลที่เดิม)

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

```
#include <FirebaseArduino.h>
```

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

```
// Config Firebase
```

```
#define FIREBASE_HOST "ledesp-86d37.firebaseio.com"
```

```
#define FIREBASE_AUTH "<YOUR KEY>"
```

```
// Config connect WiFi
```

```
#define WIFI_SSID "<YOUR WIFI NAME>"
```

```
#define WIFI_PASSWORD "<YOUR WIFI PASSWORD>"
```

```
// Config DHT
```

```
#define DHTPIN 4
```

```
#define DHTTYPE DHT11
```

```
DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
```

```
  Serial.begin(9600);
```

```
  WiFi.mode(WIFI_STA);
```

```
  // connect to wifi.
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
  Serial.print("connecting");
```

```
  while (WiFi.status() != WL_CONNECTED) {
```

```
    Serial.print(".");
```

```
    delay(500);
```

```
  }
```

```
  Serial.println();
```

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

```
Serial.println(WiFi.localIP());
```

```
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
```

```
dht.begin();
```

```
}
```

```
void loop() {
```

```
// Read temp & Humidity for DHT22
```

```
float h = dht.readHumidity();
```

```
float t = dht.readTemperature();
```

```
if (isnan(h) || isnan(t)) {
```

```
Serial.println("Failed to read from DHT sensor!");
```

```
delay(500);
```

```
return;
```

```
}
```



```
StaticJsonBuffer<200> jsonBuffer;

JsonObject& root = jsonBuffer.createObject();

root["temperature"] = t;

root["humidity"] = h;


// append a new value to /logDHT

String name = Firebase.push("logDHT", root);

// handle error

if (Firebase.failed()) {

  Serial.print("pushing /logDHT failed:");

  Serial.println(Firebase.error());

  return;

}

Serial.print("pushed: /logDHT/");

Serial.println(name);
```

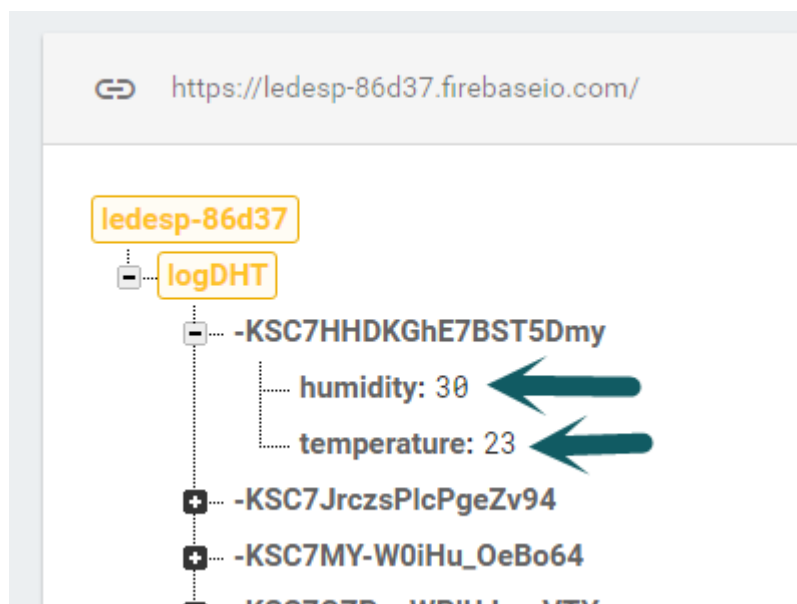
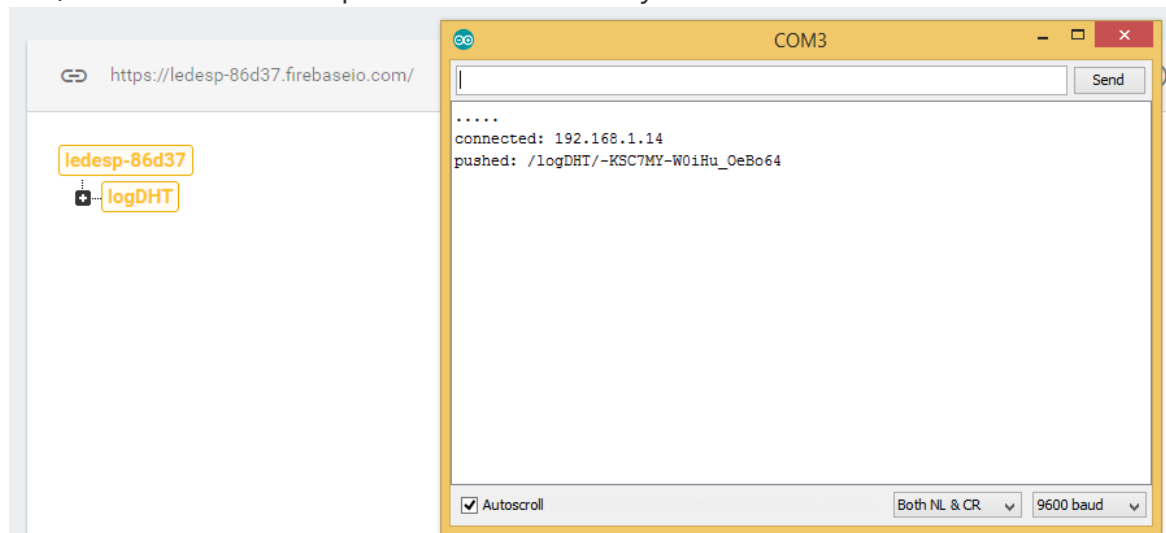
```
delay(5000);
```

```
}
```

[FirebaseTestDHT2.ino](#) hosted with ❤ by [GitHub](#)

[view raw](#)

จากนั้นเปิดหน้ามอเอนิเตอร์ขึ้นมา เทียบกับหน้าคอนโซลใน Firebase จะพบว่ามีกรเพิ่มข้อมูลเข้าไปที่ออปเจ็ค logDHT เพียงออปเจ็คเดียว เมื่อกดเครื่องหมาย + (หน้าชื่อออปเจ็ค) ก็จะมีพบกับออปเจ็คที่มี Key เป็นรหัสไม่มีความหมาย เมื่อกดเครื่องหมาย + (หน้าชื่อออปเจ็คอีกครั้ง) ก็จะมีพบกับออปเจ็ค temperature และ humidity

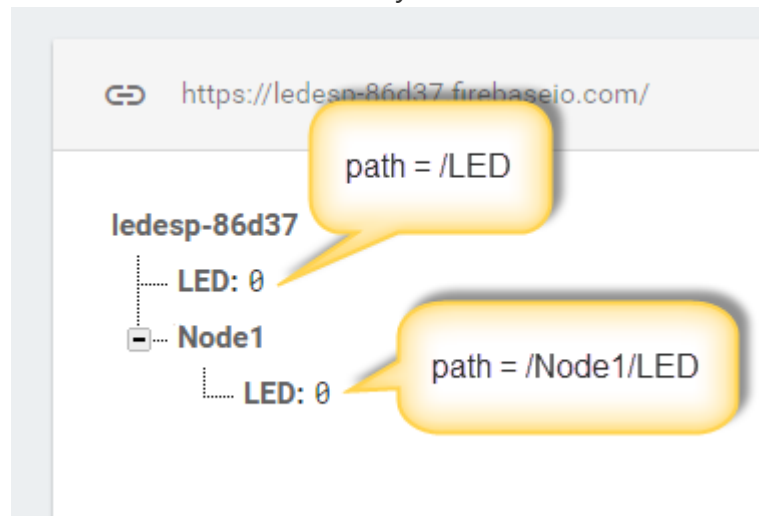


การสตรีมมิ่งข้อมูล

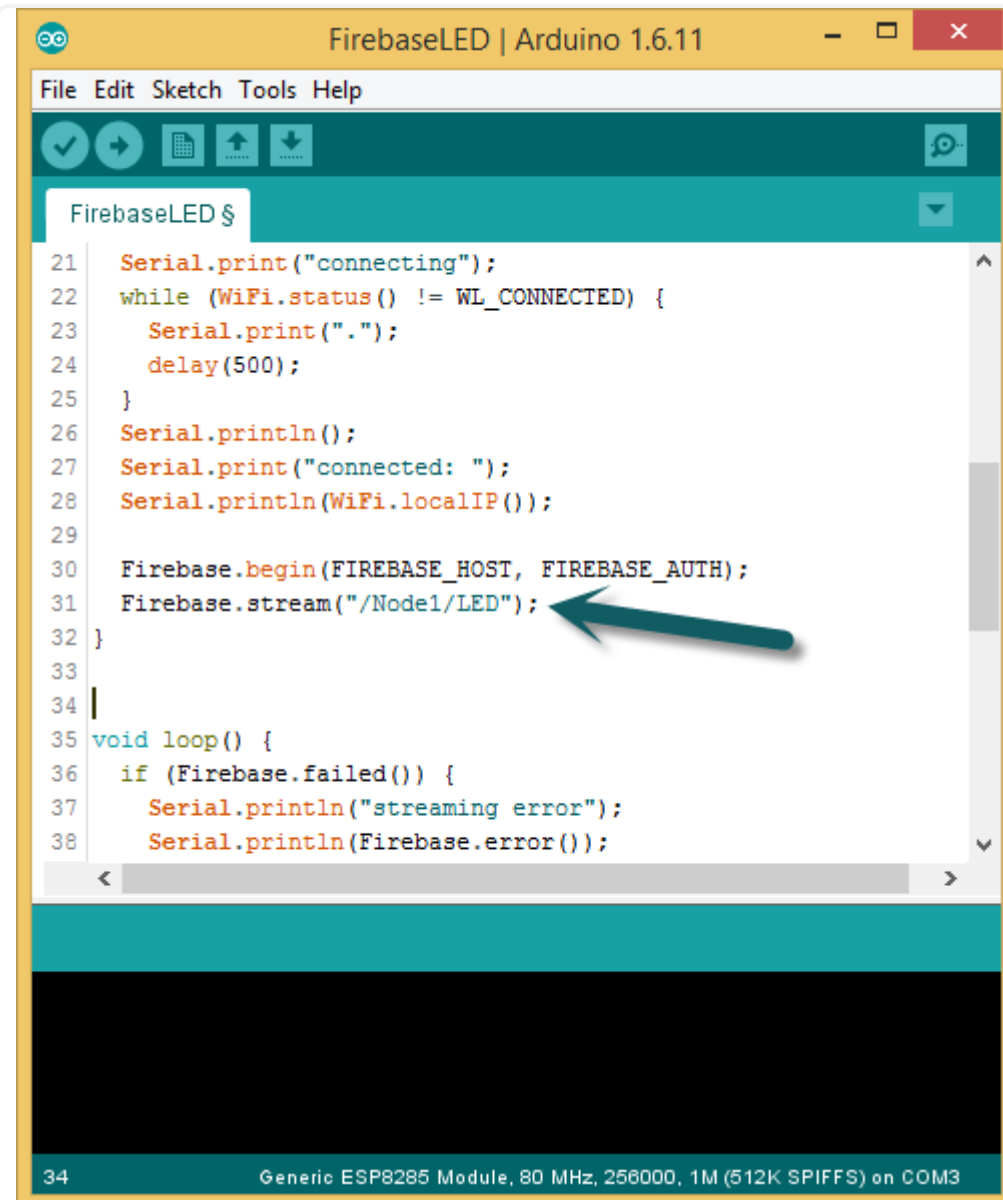
หลังจากที่ได้เรียนรู้การเพิ่มข้อมูลเข้าไปแล้ว ต่อมาเรามาลองเรียนรู้การรออัปเดตข้อมูลกันบ้างครับ สามารถนำไปประยุกต์ใช้การรับ - ส่งข้อมูลแบบเรียลไทม์ได้ครับ
ใน void setup() จะต้องเรียกฟังก์ชัน Firebase.stream() ขึ้นมาก่อน โดยมีรูปแบบการใช้งานดังนี้

Firebase.stream(const String &path)

โดยที่ path หมายถึงออปเจ็คที่เราต้องการตรวจจับความเปลี่ยนแปลง เช่น ถ้าหากออปเจ็ค LED ซึ่งอยู่เป็นออปเจ็คนอกสุด ตัว path ก็จะเป็น /LED , แต่หากออปเจ็ค LED อยู่ในออปเจ็ค Node1 และออปเจ็ค Node1 เป็นออปเจ็คนอกสุด ตัว path ก็จะเป็น /Node1/LED ตัวเครื่องหมาย / จะเป็นตัวคั่น Key หรือชื่อของออปเจ็ค



ผมจะเลือกใช้ path ที่ `/Node1/LED` ดังนั้นจึงต้องนำโค้ดด้านล่างนี้ไปไว้ใน void setup()
Firebase.stream("/Node1/LED");



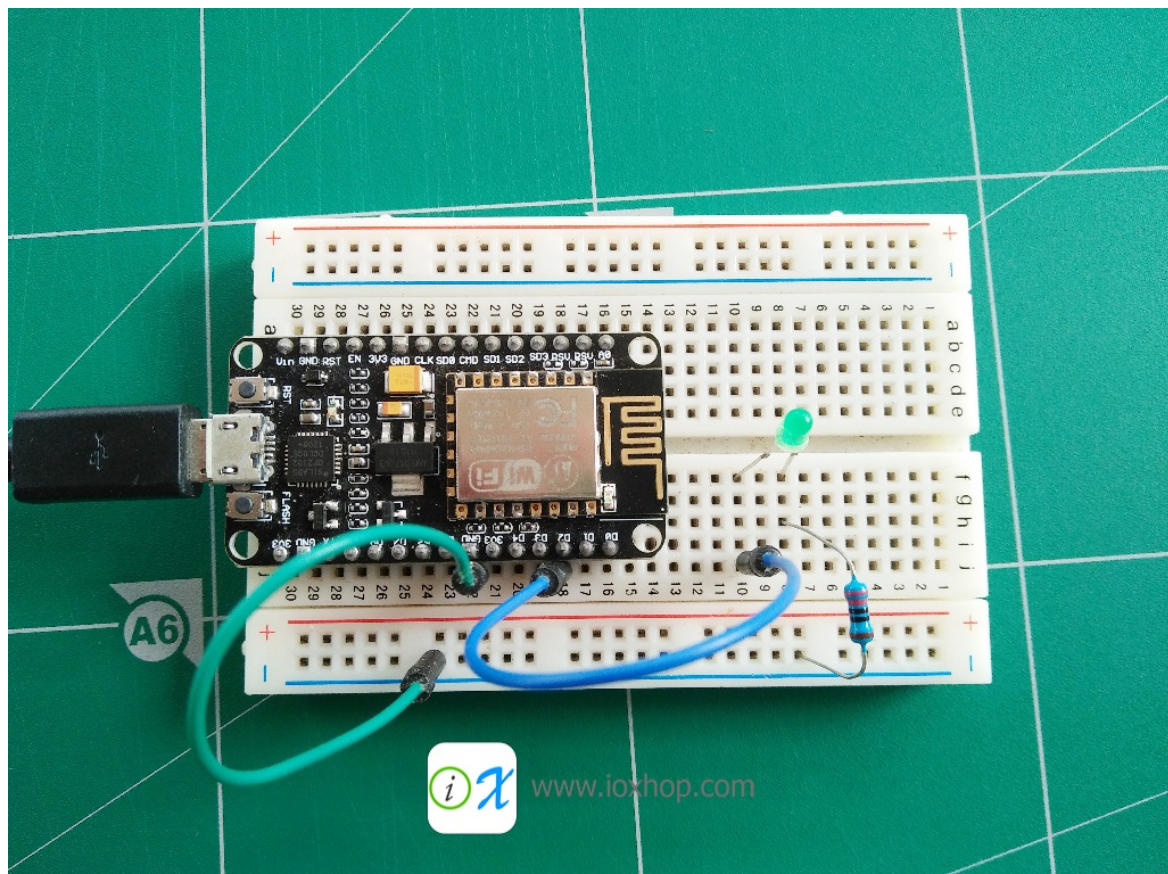
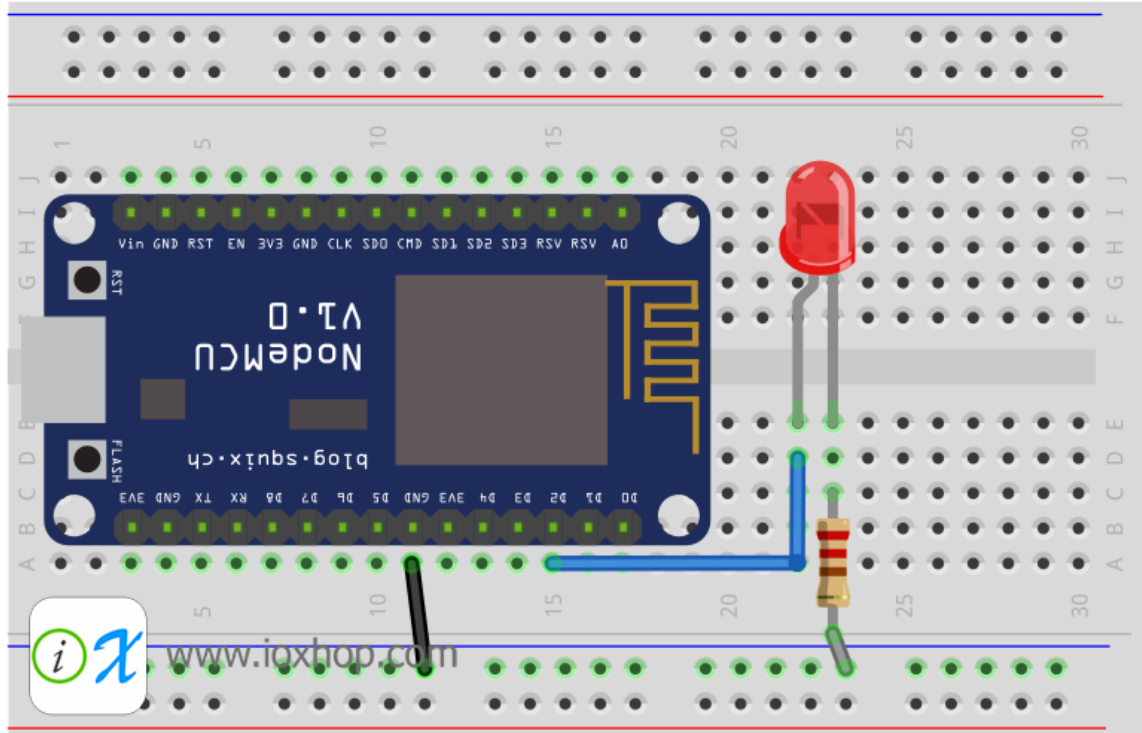
```
21 Serial.print("connecting");
22 while (WiFi.status() != WL_CONNECTED) {
23   Serial.print(".");
24   delay(500);
25 }
26 Serial.println();
27 Serial.print("connected: ");
28 Serial.println(WiFi.localIP());
29
30 Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
31 Firebase.stream("/Node1/LED");
32 }
33
34
35 void loop() {
36   if (Firebase.failed()) {
37     Serial.println("streaming error");
38     Serial.println(Firebase.error());
39   }
40 }
```

ในส่วนของ void loop() สามารถก๊อปปี้ไปได้ทั้งหมด มีเพียงแต่การเรียกฟังก์ชัน event.get???("data"); ที่ต้องระวัง เพราะหากชนิดของข้อมูลผิด ก็จะทำให้ไม่สามารถรับค่าได้เลย

ตัวอย่างเช่น หากในหน้าคอนโซล ได้กรอกไว้เป็น 1 แต่รับค่าแบบ String ผลที่ได้จะเป็นค่าว่าง เพราะ 1 ถือเป็นตัวเลข คำสั่ง event.getString("data"); จะไม่สามารถรับค่าได้ แต่ถ้าหากใช้คำสั่ง event.getInt("data"); จะสามารถรับข้อมูลได้ปกติ

ทดลองควบคุม LED แบบเรียลไทม์

ต่อวงจรด้านล่างนี้ และอัปโหลดโค้ดด้านล่างนี้ลงไป (อย่าลืมแก้ที่เดิม)



```
#include <FirebaseArduino.h>
```

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

```
#define LED_PIN 4
```

```
// Config Firebase
```

```
#define FIREBASE_HOST "ledesp-86d37.firebaseio.com"
```

```
#define FIREBASE_AUTH "<YOUR KEY>"
```

```
// Config connect WiFi
```

```
#define WIFI_SSID "<YOUR WIFI NAME>"
```

```
#define WIFI_PASSWORD "<YOUR WIFI PASSWORD>"
```

```
void setup() {
```

```
  pinMode(LED_PIN, OUTPUT);
```

```
  Serial.begin(115200);
```

```
// connect to wifi.

WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

Serial.print("connecting");

while (WiFi.status() != WL_CONNECTED) {

  Serial.print(".");

  delay(500);

}

Serial.println();

Serial.print("connected: ");

Serial.println(WiFi.localIP());


Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

Firebase.stream("/Node1/LED");

}
```

```
void loop() {

  if (Firebase.failed()) {

    Serial.println("streaming error");

    Serial.println(Firebase.error());

  }


  if (Firebase.available()) {

    FirebaseObject event = Firebase.readEvent();

    String eventType = event.getString("type");

    eventType.toLowerCase();


    if (eventType == "") return ;

    Serial.print("event: ");

    Serial.println(eventType);

    if (eventType == "put") {

      String path = event.getString("path");
```



```

int data = event.getInt("data");

Serial.println "[" + path + "]" + String(data);

if (path == "/") {

digitalWrite(LED_PIN, (data == 0 ? LOW : HIGH));

}

}

}

delay(10);

}

```

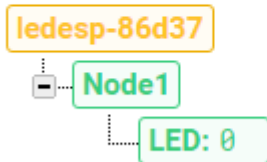
[FirebaseLEDStream.ino](#) hosted with ♥ by [GitHub](#)

[view raw](#)

กลับไปทีคอนโซล ไปเพิ่มออปเจ็ค Node1 --> LED:0 ผลที่ได้คือหลอด LED ยังดับอยู่

The screenshot shows the Firebase Realtime Database interface for the project 'ledesp-86d37'. The URL bar displays 'https://ledesp-86d37.firebaseio.com/'. Below the URL, a tree view shows the current structure: a root node 'ledesp-86d37' with a value of 'null'. A new node is being added with the name 'Node1'. Under 'Node1', a new node is being added with the name 'LED' and a value of '0'. At the bottom of the form, there are 'CANCEL' and 'ADD' buttons.

<https://ledesp-86d37.firebaseio.com/>



เมื่อแก้ออปเจ็ค LED ให้เป็น 1 ผลคือหลอด LED ติดทันที

<https://ledesp-86d37.firebaseio.com/>



สรุปช่วงท้าย

ในบทความนี้จะสอนเฉพาะการใช้ Firebase พื้นฐาน และ การใช้ Firebase บน Arduino ESP8266 / ESP8285 ซึ่งแค่นี้ก็สามารถนำไปประยุกต์ใช้การคุยกันระหว่างอุปกรณ์ กับ อุปกรณ์ได้แล้ว แต่หากต้องการจะให้การควบคุมอุปกรณ์ผ่านหน้าเว็บไซต์ หรือผ่านแอปพลิเคชัน จำเป็นจะต้องเรียนรู้ API ในภาษาอื่น ๆ เพิ่มอีก แต่ยังคงมีฟังก์ชัน set get push อยู่เหมือนเดิม เพียงแต่ในบางภาษาจะใช้งานง่ายขึ้นครับ