REALTEK

REALTEK

User guide of Google Nest's cloud database

The example guides how to import and get data from Google Nest's cloud database through Ameba and some simple Javascript and Java examples.

Table of Contents

1 Introduce

This document illustrates how to get or insert data to Google Nest's cloud database - Firebase.

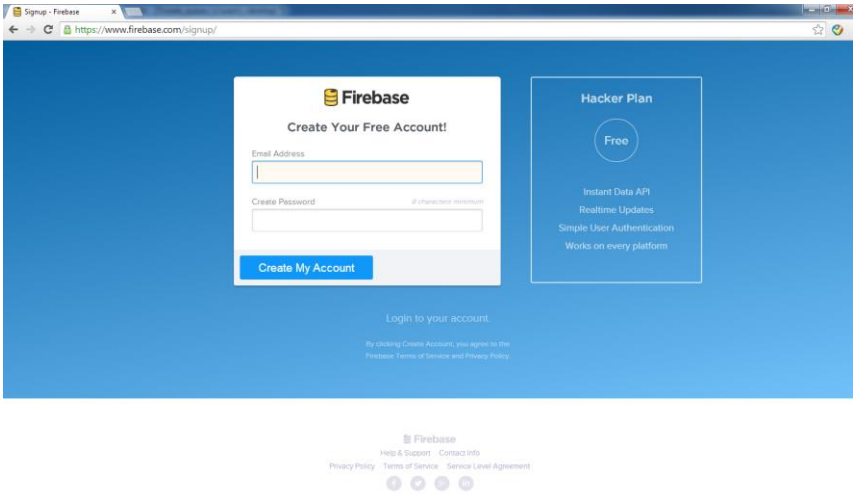
Firebase provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. It is very simple and convenient to register an account on Firebase. Then all the data can be stored in your own Firebase account and you can also manage the database easily.

Here, the Google Nest APIs are provided to access to the Google Nest's cloud database – Firebase. In these Google Nest APIs, PolarSSL is used to support SSL connection. Two examples are provided to show how to send and retrieve data from database. Besides of that, a sample code named "atcmd_google.c" shows how to do some simple actions with Firebase using AT Command.

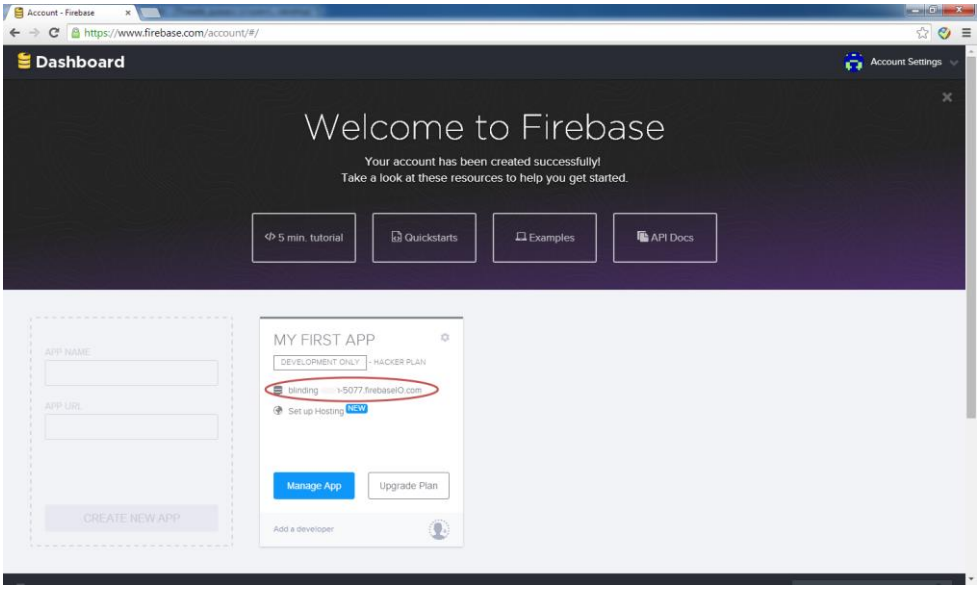
2 Start the REST API

2.1 Sign up a firebase Account

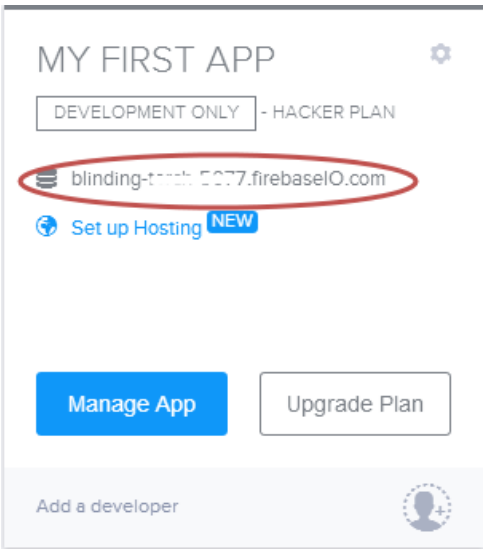
It is very simple to register an account for Firebase. It needs an Email address and then sign up on their website – www.firebase.com/signup/.



Inputting Email Address and creating a password, and a Firebase account will be generated. The homepage of Firebase account is as following, and the Firebase address is the one circled by red color.



The information required for database access is the Firebase address, so keeping this Firebase address for later usage. Then you can read and write data with your Google Nest's database. One user account can have single one Firebase address.



After this, you can also click the "Manage App" to see what information inside.

There are plenty of resources to teach how to use Firebase. Please check https://www.firebase.com/, and start from "Start Hacking", and read documentation in https://www.firebase.com/docs/.

2.2 Data Types

This sub-section lists the data types used by provided Google Nest APIs.

2.2.1 googlenest_context

```
typedef struct {
        int socket;
        char *host;
        ssl_context ssl;
} googlenest_context;
```

This structure is used to store the context of Google Nest SSL connection information.

2.3 Google Nest APIs

This sub-section lists the provided APIs for Google Nest operations.

2.3.1 gn_connect

This function triggers to connect to the Google Nest's database.

2.3.1.1 Syntax

```
int gn_connect(
googlenest_context *googlenest,
char *host,
int port
);
```

2.3.1.2 Parameters

googlenest

SSL connect information to connect the database using SSL

host

The Firebase address to access your Firebase

port

Service port information for the service to connect, normally use 443

2.3.1.3 Return Value

If the function succeeds, the return value is 0.

2.3.1.4 Remarks

This API must be used before using the request API.

2.3.2 gn_close

This function triggers to close the connection to the Google Nest's database.

2.3.2.1 Syntax

```
void gn_close(
```

```
googlenest_context *googlenest
);
```

2.3.2.2 Parameters

googlenest

SSL connect information to connect the database using SSL

2.3.2.3 Return Value

None

2.3.2.4 Remarks

This API must be used after using the request API.

2.3.3 gn_get

This function triggers to send a GET request to the Google Nest's database.

2.3.3.1 Syntax

```
int gn_get(
googlenest_context *googlenest,
char *uri,
unsigned char *out_buffer,
size_t out_len
);
```

2.3.3.2 Parameters

googlenest

SSL connect information to connect the database using SSL

uri

The path you want to get under the database and must follow ".json" behind

out_buffer

Buffer to store the data you get from the database

out_len

Length of the buffer which store the data get from database

2.3.3.3 Return Value

If the function succeeds, the return value is 0.

2.3.3.4 Remarks

It can read data from Firebase's defined path. The data retrieved from Firebase is in format of JSON.

2.3.4 gn_put

This function triggers to send a PUT request to write or replace data in the Google Nest's database.

2.3.4.1 Syntax

```
int gn_put(
googlenest_context *googlenest,
char *uri,
char *content
);
```

2.3.4.2 Parameters

googlenest

SSL connect information to connect the database using SSL

uri

The path defined to write or replace under the database and must follow ".json" behind

content

Data defined to write or replace

2.3.4.3 Return Value

If the function succeeds, the return value is 0.

2.3.4.4 Remarks

The data PUT to Firebase must in format of JSON.

2.3.5 gn_patch

This function triggers to send a PATCH request to update some of the keys for a defined path without replacing all of the data in the Google Nest's database.

2.3.5.1 Syntax

```
int gn_patch(
googlenest_context *googlenest,
char *uri,
char *content
);
```

2.3.5.2 Parameters

googlenest

SSL connect information to connect the database using SSL

uri

The path defined to update under the database and must follow ".json" behind

content

Data defined to update

2.3.5.3 Return Value

If the function succeeds, the return value is 0.

2.3.5.4 Remarks

The data PATCH to Firebase must in format of JSON.

2.3.6 gn_post

This function triggers to send a POST request to add to a list of data in the Google Nest's database. Every time you send a POST request, a unique ID will be generated.

2.3.6.1 Syntax

```
Int gn_post(
googlenest_context *googlenest,
char *uri,
char *content,
unsigned char *out_buffer,
size_t out_len
```

```
);
```

2.3.6.2 Parameters

googlenest

SSL connect information to connect the database using SSL

uri

The path defined to get under the database and must follow ".json" behind

content

Data defined to insert

out_buffer

Buffer to store the unique ID returned from the database

out_len

Length of the buffer which store the unique ID returned from database

2.3.6.3 Return Value

If the function succeeds, the return value is 0.

2.3.6.4 Remarks

The data POST to Firebase must in format of JSON.

2.3.7 gn_delete

This function triggers to send a DELETE request to remove data the Google Nest's database.

2.3.7.1 Syntax

```
Int  gn_delete(
googlenest_context *googlenest,
char *uri
);
```

2.3.7.2 Parameters

googlenest

SSL connect information to connect the database using SSL

uri

The path you want to delete under the database and must follow ".json" behind

2.3.7.3 Return Value

If the function succeeds, the return value is 0.

2.3.7.4 Remarks

None.

NOTICE: The usage of cJSON to store the data in format of JSON is provided.

2.3.8 gn_stream

This function triggers to send a Streaming request to get data which changed in the Google Nest's database through Server-Send Event.

2.3.8.1 Syntax

```
Int  gn_stream(
googlenest_context *googlenest,
char *uri
);
```

2.3.8.2 Parameters

googlenest

SSL connect information to connect the database using SSL

uri

The path you want to delete under the database and must follow ".json" behind

2.3.8.3 Return Value

If the function succeeds, the return value is 0.

2.3.8.4 Remarks

Using the callback function to get the real-time data:

void google_data_retrieve_cb(char *response_buf, int buf_size)

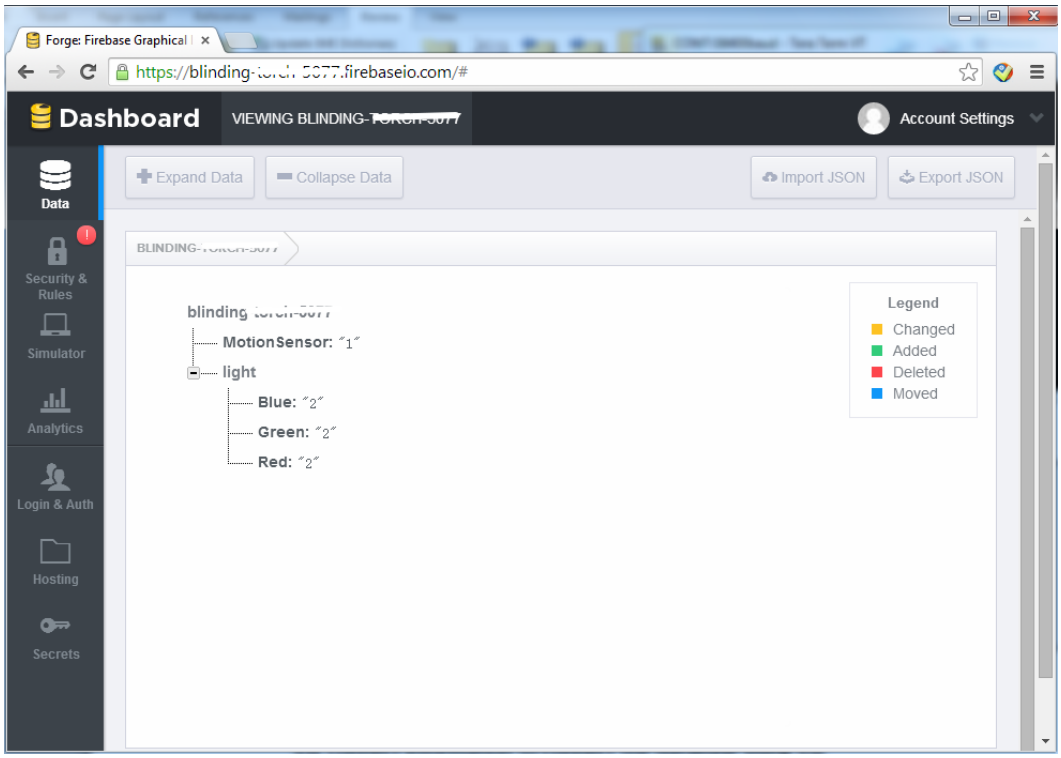
3 Example

Two examples are provided. The first example shows how a device put state to database every five seconds. The second example shows how to get data from database using streaming.

Turning on the flag CONFIG_EXAMPLE_GOOGLE_NEST
in project\realtek_ameba1_va0_example\inc\platform_opts.h, you can enable the Google Nest's example (make sure the TCP also enabled).

The data will used in database shows as following:

```
{
        "Motion_Sensor" : "i",
        "Light" : {
                        "Red" : "0",
                        "Green" : "0",
                        "Blue" : "0",
        }

}
```



3.1 Storing data from device

In this example, the Motion Sensor will send the count of person to the Firebase.

The data is as following:

{ “MotionSensor” : “i”

}

Where the count “i” will be added once person pass by. This example simulates there is a person pass by every 5 seconds.

3.1.1 How to test

Activate browser and paste the url of the Firebase address in the browser, then it is able to check the status changed overtime.

3.1.2 Access to the Firebase

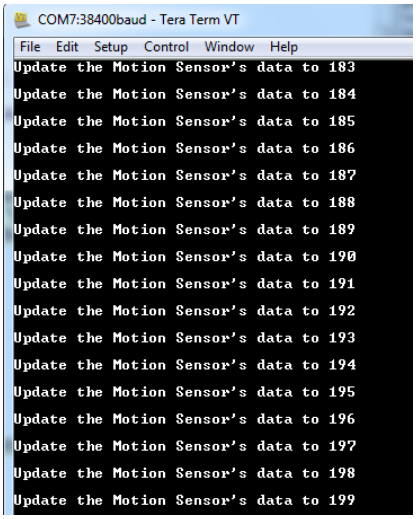
Making some modifies:

- Making sure the flag CONFIG_EXAMPLE_GOOGLE_NEST in project\realtek_ameba1_va0_example\inc\platform_opts.h turn on (make sure the TCP service also enabled).
- In example_entry.c, choose the type “#define TYPE FromDevice”
- Define your Firebase Account address in “example_google.h”

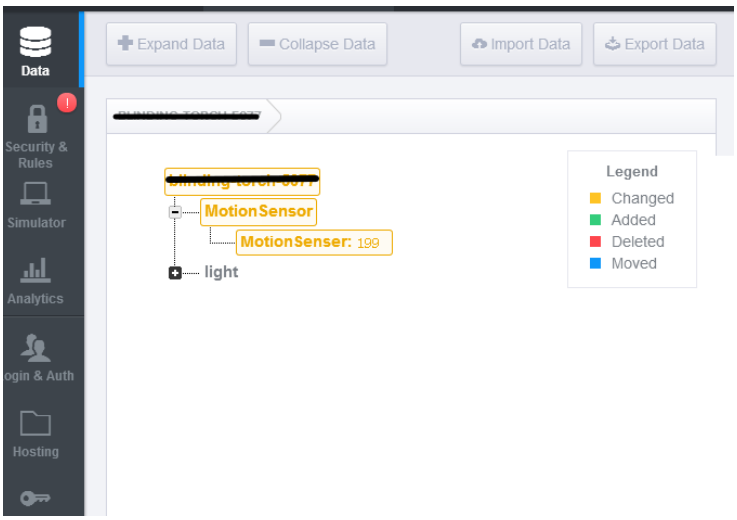
3.1.3 Start to store data

Then build this code and the data will be updated in Firebase.

Updating shows on Ameba:



Result shows on browser



3.2 Reading data

In this example, the RGB information of a light will be controlled by a webpage through Firebase.

3.2.1 How to test

Edit Firebase address in the xxx.html.

```
<script>

var myDataRef = new Firebase("https://your_firebase_address.firebaseio.com");

function submit()

{
```

Note: It is required to fill Firebase address in the javascript.

Double click xxx.html.

3.2.2 Access to the Firebase

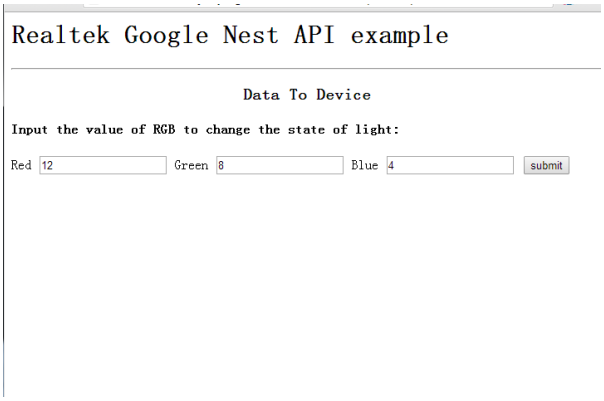
Making some modifies:

- Making sure the flag CONFIG_EXAMPLE_GOOGLE_NEST in project\realtek_ameba1_va0_example\inc\platform_opts.h turn on (make sure the TCP service also enabled).
- In example_entry.c, choose the type “#define TYPE ToDevice”
- Define your Firebase Account address in “example_google.h” and the html code

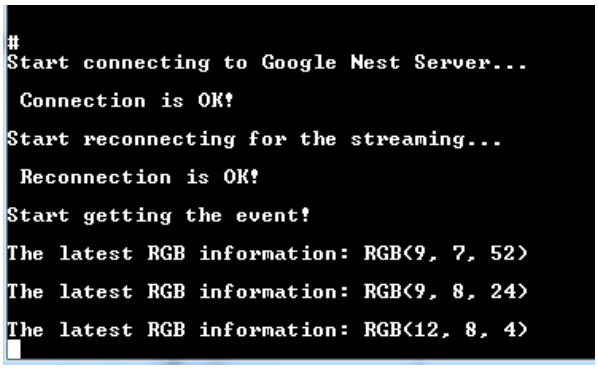
3.2.3 Start to store data

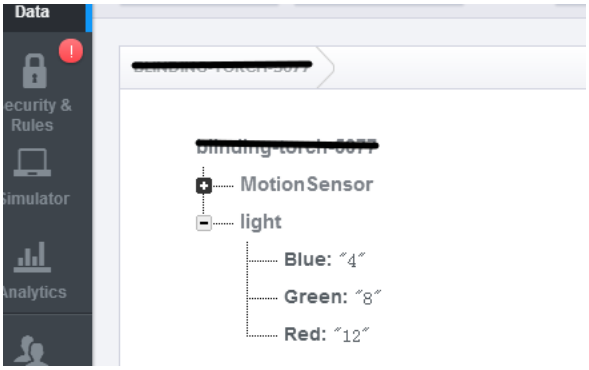
Insert data to your Firebase using the webpage provided. Then build this code and the data will be retrieved from Firebase. Control the value of RGB through webpage any time to test retrieving the latest data.

Changing RGB through webpage:



Results show on Ameba and the Firebase Account:





4 AT Command Usage

The atcmd_google.c is provided to support using the AT Command to do some simple requests. And before sending the Google Nest request, make sure the Wi-Fi is connected.

4.1 AT Command used

4.1.1 Connecting Wi-Fi

4.1.1.1 'ATW0' Wlan Set Network SSID

Description:
Command Format: ATW0=SSID<CR>
Default Value: None
Response: None

4.1.1.2 'ATW1' Wlan set Network Passphrase

Description:
Command Format: ATW1=password<CR>
Default Value: None
Response: None

4.1.1.3 'ATW2' Wlan Set Key ID

Description:
Command Format: ATW2=Key_ID<CR>
Default Value: None
Response: None

4.1.1.4 'ATWC' Wlan Join a Network

Description:
Command Format: ATWC<CR>
Default Value: None
Response: None

4.1.1.5 'ATWD' Wlan Disconnect from Network

Description:
Command Format: ATWD<CR>
Default Value: None

Response: None

4.1.2 Google Nest API

The “ATG0” command can be used to send request to Google Nest’s database.

4.1.2.1 Syntax

ATG0=[method,address,path,data]or[method,address,path]

4.1.2.2 Parameters

method

provided methods: get, put, patch, post, delete

address

The Firebase address which show how to get in section 2.1

path

The path defined to get under the database and must follow “.json” behind

Data

Only needed when using method of put, patch or post

4.2 Google Nest API AT Command usage

4.2.1 Network Connection

The “ATWC” command can be used to connect to an access point. To process the connection, an SSID should be set first. Meanwhile a password must be set except in open mode, and a key id is also required for WEP mode.

To disconnect AP, type “ATWD”.

WPA2 mode

Command sequence: (refer to 3.2.1)

#ATW0=SSID
#ATW1=passphrase
#ATWC

```
# ATW0=rtk
ATW0 match ATW0, search cnt 2
[ATW0]: _AT_WLAN_SET_SSID_ [rtk]

[MEM] After do cmd, available heap 47264


# ATW1=12345678
ATW1 match ATW1, search cnt 1
[ATW1]: _AT_WLAN_SET_PASSPHRASE_ [12345678]

[MEM] After do cmd, available heap 47264


# ATWC
ATWC match ATWC, search cnt 2
[ATWC]: _AT_WLAN_JOIN_NET_

Joining BSS ...RTL8195A[Driver]: set ssid [rtk]
RTL8195A[Driver]: start auth
RTL8195A[Driver]: auth success, start assoc
RTL8195A[Driver]: association success(res=2)

wifi_handshake_done_hdl 31
CCConnected after 1261ms.
RTL8195A[Driver]: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:1
RTL8195A[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)

IP address : 192.168.1.100

GGGot IP after 2782ms.


[MEM] After do cmd, available heap 46616
```

#ATWD

```
# ATWD
ATWD match ATWD, search cnt 1
[ATWD]: _AT_WLAN_DISC_NET_

Deassociating AP ...
ioctl[SIOCGIWESSID] ssid = NULL, not connected
WIFI disconnected

[MEM] After do cmd, available heap 47376
```

WEP mode

Command sequence: (refer to 3.2.1)

#ATW0=SSID
#ATW1=Password
#ATW2=Key id
#ATWC

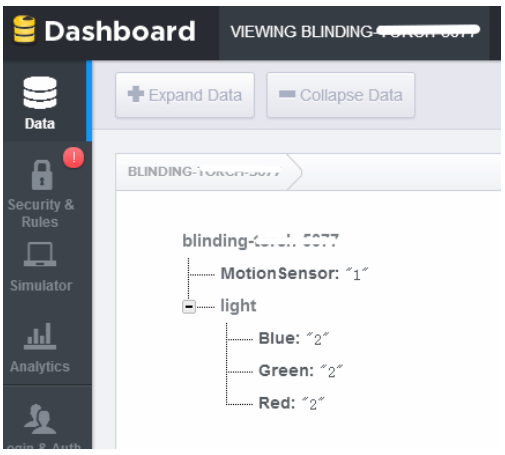
The WEP key can be 5 ASCII characters for WEP 40 or 13 ASCII characters for WEP 104. The key ID should be 0, 1, 2 or 3. The following is an example to connect network by using WEP 40 with key ID 0.

```
# ATW0=rtk
ATW0 match ATW0, search cnt 2
[ATW0]: _AT_WLAN_SET_SSID_ [rtk]

[MEM] After do cmd, available heap 47480


# ATW1=12345
ATW1 match ATW1, search cnt 1
[ATW1]: _AT_WLAN_SET_PASSPHRASE_ [12345]

[MEM] After do cmd, available heap 47480


# ATW2=0
ATW2 match ATW2, search cnt 2
[ATW2]: _AT_WLAN_SET_KEY_ID_ [0]

[MEM] After do cmd, available heap 47480


# ATWC
ATWC match ATWC, search cnt 2
[ATWC]: _AT_WLAN_JOIN_NET_

Joining BSS ...RTL8195A[Driver]: set ssid [rtk]
RTL8195A[Driver]: set group key to hw: alg:1(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:0
RTL8195A[Driver]: start auth
RTL8195A[Driver]: auth success, start assoc
RTL8195A[Driver]: association success(res=1)

wifi_connected_hdl 31
CCConnected after 1286ms.

IP address : 192.168.1.100

GGGot IP after 1801ms.


[MEM] After do cmd, available heap 46616
```

4.2.2 Use Google Nest API

4.2.2.1 Get data from database

- The database:



- Command:

  ATG0=[get,xxxxxxxx.firebaseio.com,path.json]

- Result:

```
# ATG0=[get,[redacted].firebaseio.com,light.json]
[ATG0]: _AT_WLAN_GOOGLENEST_

Get data from googlenest: {"Blue":"2","Green":"2","Red":"2"}
[MEM] After do cmd, available heap 29448

#
```

4.2.2.2 Put or patch data to database

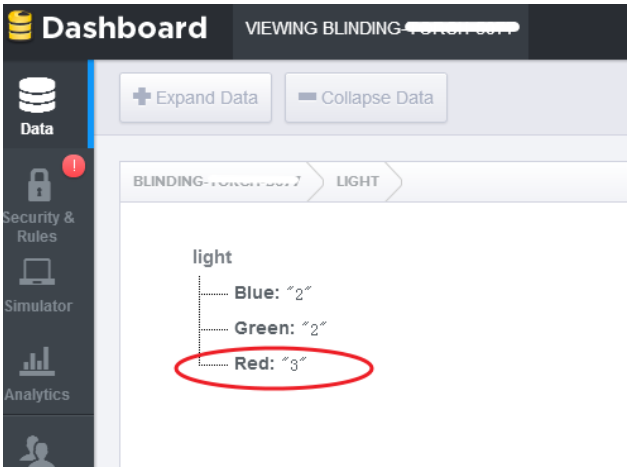
- Command:

  ATG0=[put,xxxxxxxx.firebaseio.com,path.json,data]

- Result:

```
# ATG0=[put,[redacted].firebaseio.com,light/Red.json,"3"]
[ATG0]: _AT_WLAN_GOOGLENEST_

Saving data in firebase is successful!
[MEM] After do cmd, available heap 29448

#
```

- The database:



4.2.2.3 Post a list of data to database

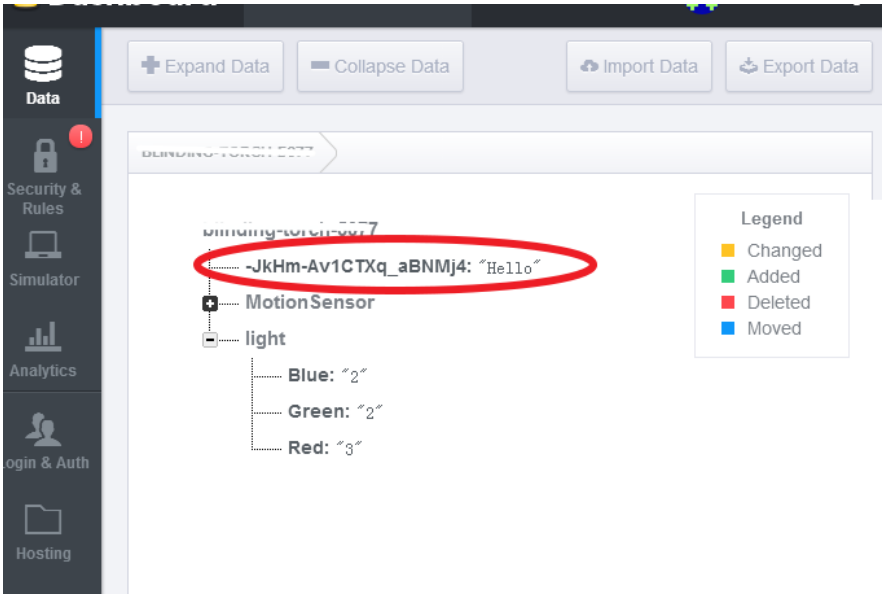
- Command:

  ATG0=[post,xxxxxxxx.firebaseio.com,path.json,data]

- Result:

```
# ATG0=[post,[redacted].firebaseio.com,.json,"Hello"]
[ATG0]: _AT_WLAN_GOOGLENEST_

Inserting data to firebase is successful!

The unique name for this list of data is: {"name":"-JkHm-Av1CTXq_aBNMj4"}
[MEM] After do cmd, available heap 29448

#
```

- The database:



4.2.2.4 *Delete data in database*

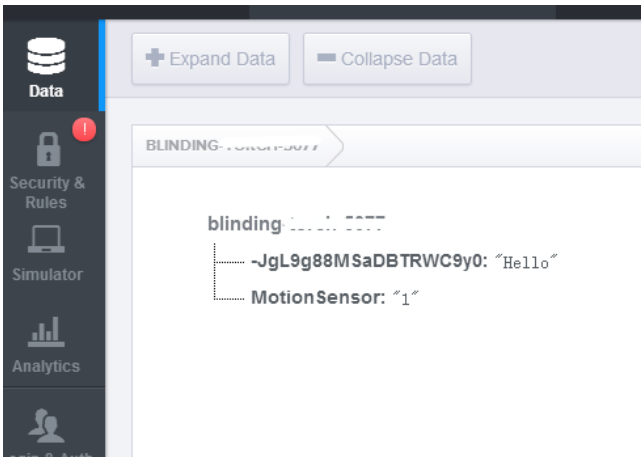
- Command:

  ATG0=[delete,xxxxxxxx.firebaseio.com,path.json]

- Result:

```
# ATG0=[delete,[redacted].firebaseio.com,light.json]
[ATG0]: _AT_WLAN_GOOGLENEST_

Delete the data is successful!
[MEM] After do cmd, available heap 29448

#
```

- The database:

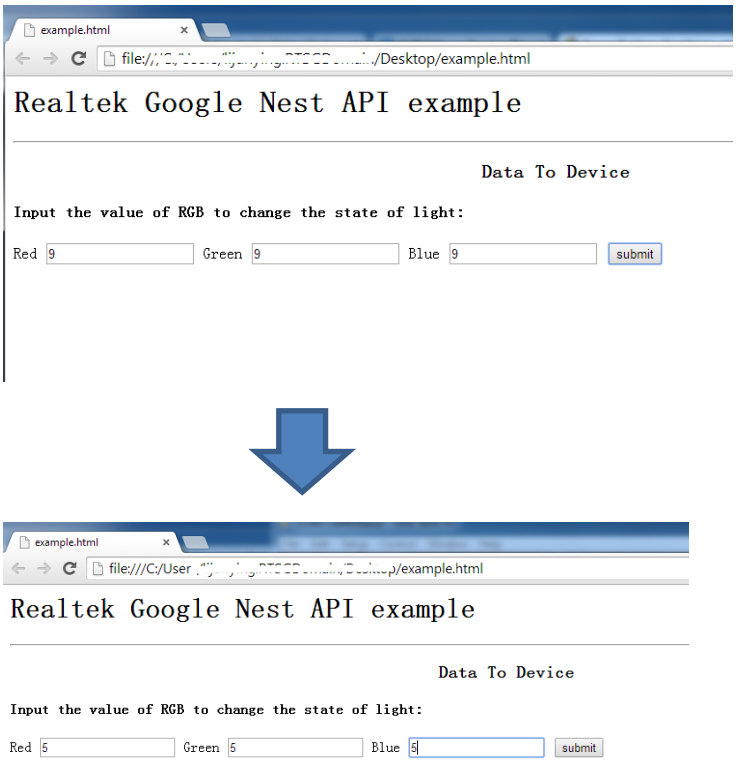


4.2.2.5 Get real-time data in database

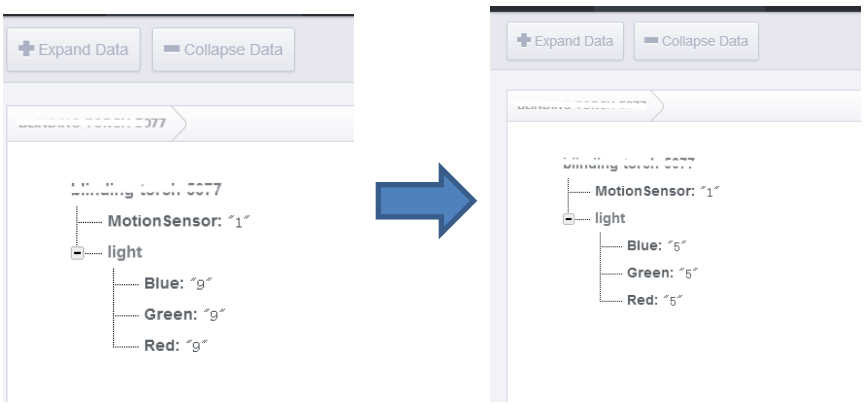
- Command:

  ATG0=[stream,xxxxxxxx.firebaseio.com,path.json]

- Control by webpage:



- The database:



- Result: