

Objective: In this tutorial, you will learn how to install SQLite on a Raspberry Pi, install an SQLite packaged for Node-RED and learn how to write simple SQL queries to interact with your SQLite database using Node-RED.

Required Setup: Connect GrovePi+ board to RPi and have all GrovePi+ libraries installed.

Parts:

- RPi 3 B+
- GrovePi+ board
- Two Grove connection wires

PART A: INSTALL SQLITE

Step 1. First, we need to install the SQLite onto the Raspberry Pi. Open the terminal and run the following command. Type "Y" if anything prompts.

```
sudo apt-get install sqlite3
```

```
pi@raspberrypi:~ $ sudo apt-get install sqlite3
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libc-ares2 libhttp-parser2.8 libuv1 nodejs-doc realpath vlc-plugin-notify vlc-plug
  vlc-plugin-video-splitter vlc-plugin-visualization
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  sqlite3-doc
The following NEW packages will be installed:
  sqlite3
0 upgraded, 1 newly installed, 0 to remove and 75 not upgraded.
Need to get 709 kB of archives.
After this operation, 1,991 kB of additional disk space will be used.
Get:1 http://mirror.web-ster.com/raspbian/raspbian stretch/main armhf sqlite3 armhf
Fetched 709 kB in 0s (747 kB/s)
Selecting previously unselected package sqlite3.
(Reading database ... 155885 files and directories currently installed.)
Preparing to unpack .../sqlite3_3.16.2-5+deb9u1_armhf.deb ...
Unpacking sqlite3 (3.16.2-5+deb9u1) ...
Setting up sqlite3 (3.16.2-5+deb9u1) ...
Processing triggers for man-db (2.7.6.1-2) ...
```

Step 2. After the installation is completed, the SQLite libraries are supplied with an SQLite shell. Use this following command to launch the shell.

```
sqlite3 sensordata.db
```

Step 3. The SQLite is successfully installed! Now let's create a table. Structured Query Language (SQL) is the language that is used for interacting with databases. It can be used to **create tables**, **insert**, **update**, **delete** and **search** for data. SQL statements must end with a semicolon (;). It's common for SQL commands to be capitalized, but this isn't strictly necessary. Most people prefer to do that because it helps readability. Don't worry if you are not familiar with SQL.

a) Let's use the following command to create a table:

```
sqlite> BEGIN;

sqlite> CREATE TABLE dhtreadings(id INTEGER PRIMARY KEY AUTOINCREMENT,
temperature NUMERIC, humidity NUMERIC, currentdate DATE, currenttime TIME,
device TEXT);

sqlite> COMMIT;
```

b) Type the following command to see all the tables:

```
sqlite> .tables

dhtreadings
```

c) It will return the newly created table named 'dhtreadings'. You can see the fullschema of the tables when you type the following commands:

```
sqlite> .fullschema

CREATE TABLE dhtreadings(id INTEGER PRIMARY KEY AUTOINCREMENT,
temperature NUMERIC, humidity NUMERIC, currentdate DATE, currenttime TIME,
device TEXT);
```

Step 4. To insert new temperature and humidity readings in the database, we can do something like this:

```
sqlite> BEGIN;

sqlite> INSERT INTO dhtreadings(temperature, humidity, currentdate,
currenttime, device) values(22.4, 48, date('now'), time('now'), "manual");

sqlite> COMMIT;
```

(In a future blog post, the ESP8266 is going to send the readings to a Python web server that inserts the data in the table.)

Step 5. To access the data stored in the database, we use the SELECT SQL statement:

```
sqlite> SELECT * FROM dhtreadings;

1|22.4|48|2017-01-26|23:43:13>manual

sqlite> COMMIT;
```

PART B: INSTALLING NODE-RED SQLITE

Step 1 <Install Node-RED SQLite>:

- a) We can Run the following list of commands (one at a time) to install node-red-node-sqlite in your Node-RED user directory. These packages give us the basic access to an SQLite database.

```
pi@raspberrypi:~ $ cd ~/.node-red
pi@raspberrypi:~/.node-red $ npm install node-red-node-sqlite
```

- b) Restart your Node-RED software with the next commands for the changes to take effect:

```
pi@raspberrypi:~/.node-red $ cd
pi@raspberrypi:~ $ node-red-stop
pi@raspberrypi:~ $ node-red-start
```

Step 2 < Checking the installation>: When your Node-RED software is back on, you can open it entering the RPI IP address in a web browser followed by :1880 as follows:

http://YOUR_RPI_IP_ADDRESS:1880

A new node called **sqlite** should appear on the left under the storage tab:

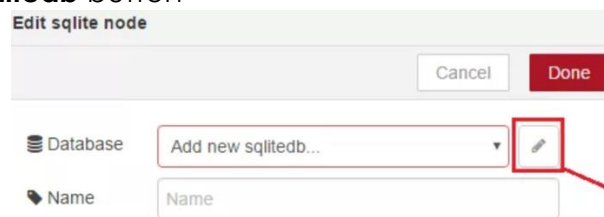


Step 3 < Creating the flow>: In this flow, you're going to send 2 SQL queries (INSERT, and SELECT) to your SQLite database. Follow these next 10 steps to create your flow:

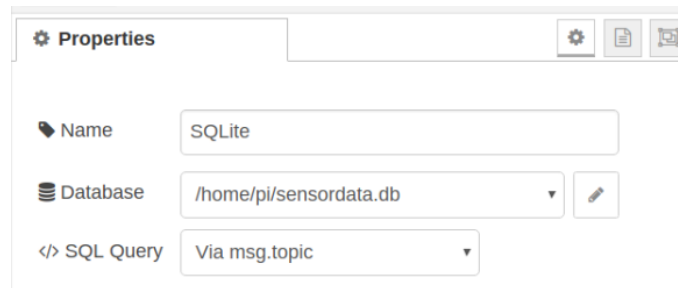
- a) Drag two **inject nodes**: (1) a **sqlite** node and (2) a **debug** node.



- b) Press the **Add new sqllitedb** button

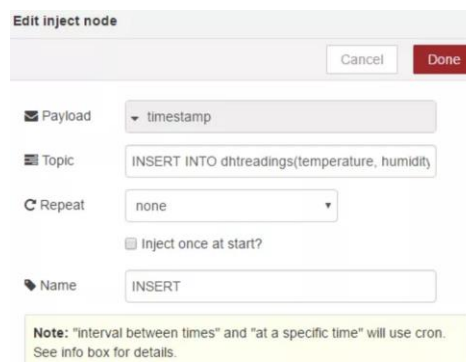


- c) Type **/tmp/sqlite** in the **Database** field



- d) Configure your **INSERT inject** node:

```
INSERT INTO dhtreadings(temperature, humidity, currentdate,
currenttime, device) values(22.4, 48, date('now'), time('now'),
"manual")
```



- e) Configure your **SELECT inject** node with:

SELECT * FROM dhtreadings

Edit inject node
 Cancel Done

☒ Payload timestamp

☒ Topic SELECT * FROM dhtreadings

☒ Repeat none

☐ Inject once at start?

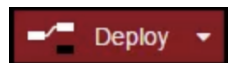
☒ Name SELECT

Note: "interval between times" and "at a specific time" will use cron.
See info box for details.

f) Connect all your nodes



g) To save your application, you need to click the deploy button on the top right corner

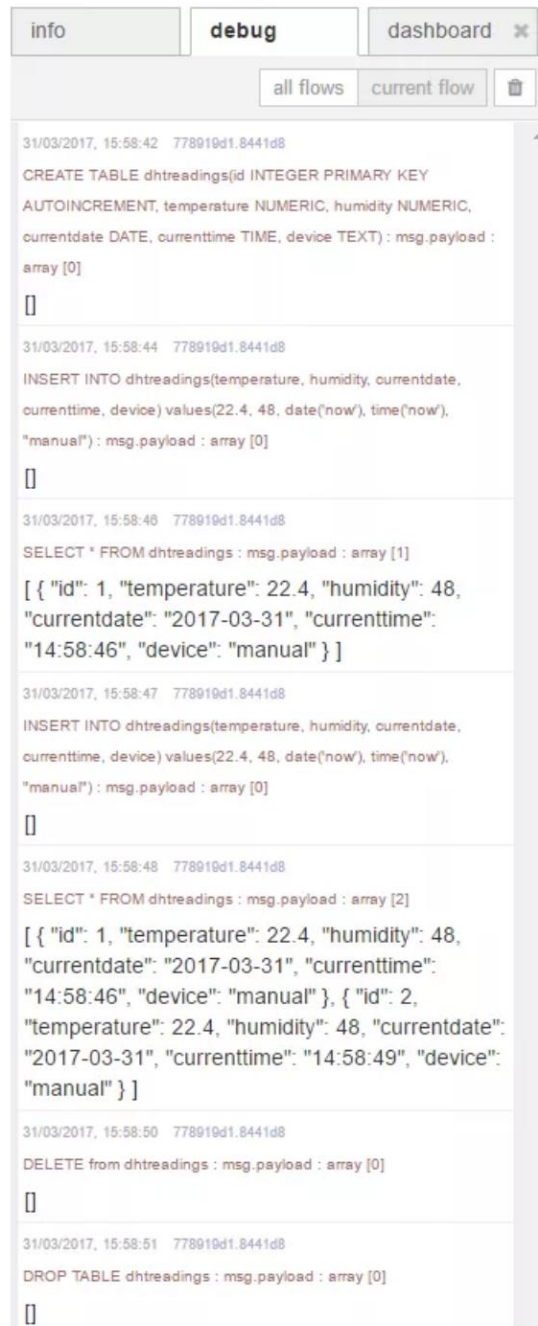


Now, your application is saved and ready.

Step 3 < Testing the flow>: Let's test the simple flow. Open the debug window and press the first inject node to trigger the CREATE SQL query. Then, follow this procedure:

1. CREATE
2. INSERT
3. SELECT
4. INSERT
5. SELECT

As we can see, the database schema was created, and data was inserted. We can select the data from the **dhtreadings** table, delete it and drop the table.



The screenshot shows the Node-RED interface with the 'debug' tab selected. The console displays a series of messages, each containing a timestamp, a unique ID, and a JSON payload. The messages represent a sequence of SQL operations: creating a table, inserting data, selecting data, and finally deleting and dropping the table.

```
31/03/2017, 15:58:42 778919d1.8441d8
CREATE TABLE dhtreadings(id INTEGER PRIMARY KEY
AUTOINCREMENT, temperature NUMERIC, humidity NUMERIC,
currentdate DATE, currenttime TIME, device TEXT): msg.payload :
array [0]
[]

31/03/2017, 15:58:44 778919d1.8441d8
INSERT INTO dhtreadings(temperature, humidity, currentdate,
currenttime, device) values(22.4, 48, date('now'), time('now'),
"manual"): msg.payload : array [0]
[]

31/03/2017, 15:58:46 778919d1.8441d8
SELECT * FROM dhtreadings : msg.payload : array [1]
[ { "id": 1, "temperature": 22.4, "humidity": 48,
"currentdate": "2017-03-31", "currenttime":
"14:58:46", "device": "manual" } ]

31/03/2017, 15:58:47 778919d1.8441d8
INSERT INTO dhtreadings(temperature, humidity, currentdate,
currenttime, device) values(22.4, 48, date('now'), time('now'),
"manual"): msg.payload : array [0]
[]

31/03/2017, 15:58:48 778919d1.8441d8
SELECT * FROM dhtreadings : msg.payload : array [2]
[ { "id": 1, "temperature": 22.4, "humidity": 48,
"currentdate": "2017-03-31", "currenttime":
"14:58:46", "device": "manual" }, { "id": 2,
"temperature": 22.4, "humidity": 48, "currentdate":
"2017-03-31", "currenttime": "14:58:49", "device":
"manual" } ]

31/03/2017, 15:58:50 778919d1.8441d8
DELETE from dhtreadings : msg.payload : array [0]
[]

31/03/2017, 15:58:51 778919d1.8441d8
DROP TABLE dhtreadings : msg.payload : array [0]
[]
```

References:

1. <https://randomnerdtutorials.com/sqlite-database-on-a-raspberry-pi/>
2. <https://randomnerdtutorials.com/sqlite-with-node-red-and-raspberry-pi/>