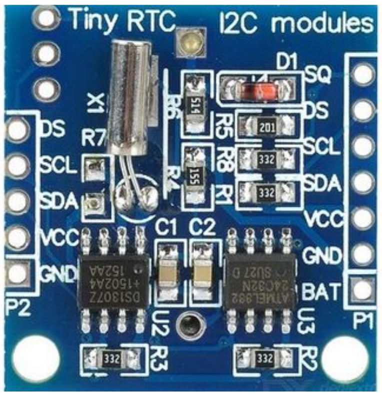
NODEMCU Lua ESP8266 With Real Time Clock (RTC) & EEPROM

by JohnL142

Getting the correct time is essential if you want to keep a data log. There are various ways to get the time from sources on the Internet.

You may ask why not use the ESP8266 to keep time for you? Well you can, it has its own internal RTC (Real Time Clock), but the ESP8266 has 3 different operating clock frequencies - 52MHz when it boots, 80MHz during regular operation, and 160MHz if boosted. If you need more accurate time keeping, especially over longer periods, then an external RTC might provide a solution. These modules also have a battery backup in case of power loss. An RTC is not terribly accurate as it counts the time elapsed since it was set and although it may do for most applications, it may not be good enough for critical time keeping. It is possible to get the an accurate time from a SNTP time server from which the RTC can be updated at regular intervals if required.



The DS1307 Tiny RTC I2C Module (above) is an example of these items and can be purchased on Ebay and other suppliers for less than £2. There are also others like the DS1302 and DS3231 which work in a similar way and cost from 99p upwards.

The DS1307 module uses an I2C interface and for an ESP-01 should be connected as:

```
Vcc - 3.3v, Gnd - Gnd, SDA - D3, SCL - D4
```

SDA and SCL can be connected to any of the I/O pins on the larger ESP8266's (alter the code accordingly). Only the left hand side pins need to be connected on this module.

Step 1: Google Time

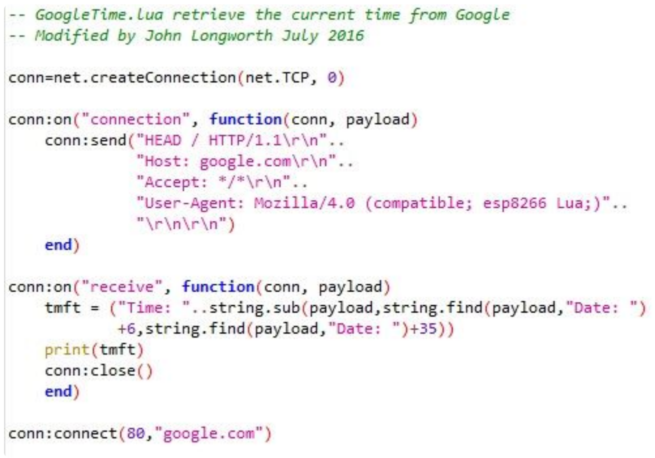
There are many examples of getting the time from Google and look something like this. When you run the GoogleTime.lua program you get a result like this:

```
>
```

dofile("GoogleTime.lua")

```
> Time: Fri, 15 Dec 2017 11:19:45 GMT
```

The problem with this method is that you get the time in string format and you have to split the string into its individual bits for hours, minutes, seconds etc. The RTC accepts the time in a special format i.e. UNIX time stamp. In layman's terms this is the number of seconds that have elapsed since Thursday 1st January 1970 to the present day and time. The UNIX Epoch (1970/01/01 00:00:00) is used by most computer operating systems and the elapsed time is stored as a signed 32 bit number. This means that this system will work up to 19th January 2038 when the number will become too big to store this way. One solution is to store the number as 64 bits, but for now the 32 bit method will suffice.



```lua
-- GoogleTime.lua retrieve the current time from Google
-- Modified by John Longworth July 2016

conn=net.createConnection(net.TCP, 0)

conn:on("connection", function(conn, payload)
    conn:send("HEAD / HTTP/1.1\r\n"..
              "Host: google.com\r\n"..
              "Accept: */*\r\n"..
              "User-Agent: Mozilla/4.0 (compatible; esp8266 Lua;)"..
              "\r\n\r\n")
    end)

conn:on("receive", function(conn, payload)
    tmft = ("Time: "..string.sub(payload,string.find(payload,"Date: ")
            +6,string.find(payload,"Date: ")+35))
    print(tmft)
    conn:close()
    end)

conn:connect(80,"google.com")
```

https://www.instructabl...

Download

To set the time to 2015 July 9, 18:29:49 on the internal RTC you would use this line of code:

```
> rtctime.set(1436430589, 0)
```

The 2 parameters are seconds and micro seconds.

You can find more information reading the NodeMCU Documentation.

Step 2: SNTP Time Servers

Simple Network Time Protocol (SNTP) is provided from many sources on the Internet, and many countries throughout the world have this service.

The program, SNTPTime2.lua sets the the time on the internal RTC. You need to have the **rtctime** & **sntp** modules in your build when you flash your ESP8266. The program gets the time from the server in seconds and micro seconds and sets the internal RTC with **rtctime.set(sec, usec).**

The program then displays the date and time in different formats.

There are many SNTP Servers around the world and some are as follows:

- **sntp.sync({"216.239.35.0"},**
- **sntp.sync({"0.uk.pool.ntp.org", "0.uk.pool.ntp.org"},**
- **sntp.sync({"3.uk.pool.ntp.org","143.210.16.201"},**
- **sntp.sync({"0.uk.pool.ntp.org","1.uk.pool.ntp.org","3.uk.pool.ntp.org"},**

All the above lines of code can be substituted into the the SNTPTime2.lua program.

There are more SNTP Servers at the addresses below which again can be used in the program.

- **93.170.62.252, 130.88.202.49, 79.135.97.79, ntp.exnet.com**

Google also provides time servers at these addresses:

- **216.239.35.0, 216.239.35.4, 216.239.35.8, 216.239.35.12**

You need to remember to get the time from the country you are in or you might have to modify it for the different world time zones. Also some countries have daylight saving time, so you might have to deal with that as well.

https://www.instructabl...

Download

Step 3: Getting the Time From RTC Module

The program GetRTCTime.lua reads the time from the internal RTC.

The first part reads the time and displays it in seconds and microseconds.

The second part converts it into more human readable format.

when calling **tm = rtctime.epoch2cal(rtctime.get())** it returns:

- year - 1970 ~ 2038
- mon - month 1 ~ 12 in current year
- day - day 1 ~ 31 in current month
- hour
- min
- sec
- day - day 1 ~ 366 in current year
- wday - day 1 ~ 7 in current week (Sunday is 1)

Each item can be accessed as **tm["day"] , tm["year"]...**

You can find more information reading the NodeMCU Documentation.

DisplaySNTPtime.lua is a more elaborate way of showing the date and time on a LCD 128 x 64 OLED display, as it is easily connected and can be used with these programs.

https://www.instructabl... Download

https://www.instructabl... Download

Step 4: RTC User Memory

A little diversion from the time keeping is the internal RTC on the ESP8266 has 128 x 32 bit memory addresses that can be accessed by the programmer. They are especially useful as they can survive the deep sleep cycle of the ESP8266. It is up to the programmer to control their use and ensure that they are not overwritten accidentally.

I have included RTCmem.lua, a simple program which demonstrates its use. You need to have **rtcmem** module in your build.

https://www.instructabl... Download

Step 5: External RTC Modules

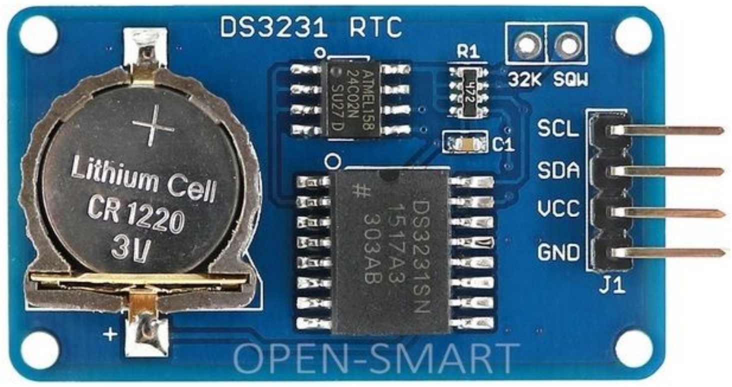
The external RTC modules connect to the ESP8266 through the I2C interface, which only uses two I/O pins and so works with the ESP-01 as well as most of the other ESP8266 devices.

The RTC module address is 0x68 and is accessed using the normal I2C commands. There is, however, something to bear in mind, the data in the RTC registers is stored in BCD format (base 16), so your programs have to deal with this. The time and date are stored in 7 registers within the RTC. On the internal RTC, the BCD conversions are taken care of by the **rtctime** module.

SetExtRTC.lua converts the data to BCD and sets the time.

ReadExtRTC.lua reads the time data and prints it out. **NOTE:** the data is printed out in hexadecimal.

I have not spent a lot of time formatting the display as you may have your own ideas about what you want to do with date and time. This is the basic engine in its simplest form, so that you can develop it further if you wish.



https://www.instructabl... Download

https://www.instructabl... Download

Step 6: Data Logging

If you look closely at the RTC modules, you will notice that they have an AT24C32 EEPROM IC or similar built into them, or you can use a 24C256 board as above. Most of these EEPROM ICs have similar pin outs as above. They come with various amounts of storage, but they are all accessed in the same way. As the AT24C32 is already soldered onto the board, it can be used directly from the I2C of the external RTC.

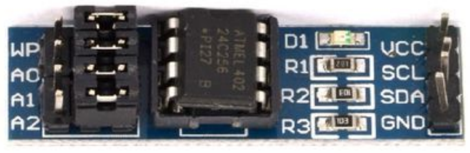
If you only have a 24C256 IC or similar, you can set it up in a bread board, connect A1, A2 and A3 to Gnd, Vcc to 3.3V and SDA AND SCL to I2C, WP can be left floating. Some EEPROM ICs only operate at 5V, so check the relevant data sheet first.

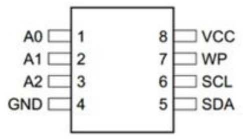
ByteWR.lua writes 1 byte of data to memory location 0x00 of the EEPROM and reads it back.

Desiderata.lua writes a couple of lines from famous text to the EEPROM.

eeRead.lua reads data from the EEPROM and prints it out.

NOTE: These programs should work with other EEPROM boards as well.





https://www.instructabl… Download

https://www.instructabl… Download

https://www.instructabl… Download

Step 7: Conclusion

I have tried to show how the RTC and EEPROM works for data logging. This is just a starter for you to develop further. You can connect various devices to the I2C bus such as light sensors, barometric pressure sensors, temperature and humidity sensors and record the data on the EEPROM.