Wi-Fi module ESP8266
programmed from Arduino IDE

A fashionable thing for the year 2014-2015 was ESP8266 WiFi board that costs only 2 EUR on Ebay causing Internet of Things revolution. On a single chip is a microcontroller and 2.4 GHz Wi-Fi part with RF coils built into the chip. Miniature circuit board consists of ESP chip, quartz, antenna and flash memory chip. Modification ESP-01 has only a few GPIOs but is perfect for driving a single LED or reading out DS18B20 temperature sensor but in future I would buy ESP-12 with more legs attached.

ESP8266 Remote Serial Wireless Transceiver WIFI Module

$2.55 Esp-12 AP+STA new From China

Buy It Now

Top Rated Plus

Free international shipping

1523 sold

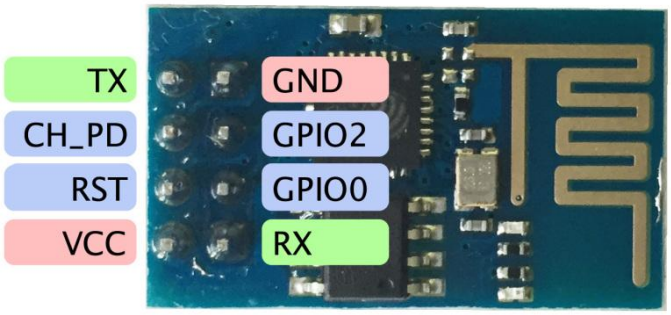
Since 2015 ESP8266 can be programmed from Arduino IDE 1.6.5. Compilation support needs to be installed and ESP libraries downloaded from Github. Some links below how to do it:

http://iot-playground.com/2-uncategorised/41-esp8266-ds18b20-temperature-sensor-arduino-ide

https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/using-arduino-ide

http://makezine.com/2015/04/01/installing-building-arduino-sketch-5-microcontroller/

https://www.hackster.io/rayburne/esp8266-01-using-arduino-ide-67a124

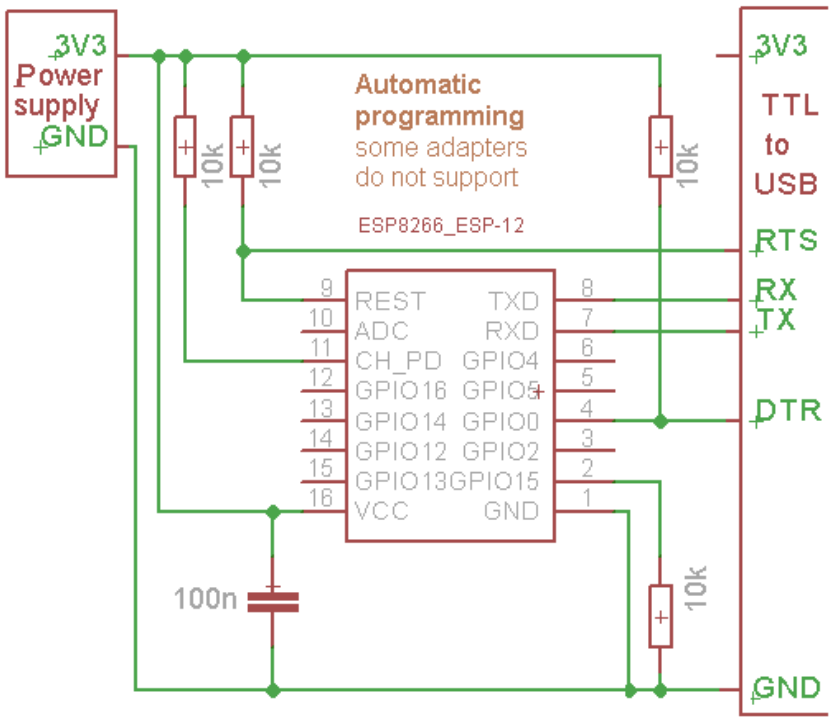


First example to try is to burn the blink example digitalWrite(2, HIGH); and to connect a LED between GPIO2 and GND. During flashing the blue LED on ESP board should be blinking.

Note: GPIO2 should be high during boot, or it will not boot into running mode. This does not allow to connect a switch to the GPIO2, just a pushbutton. Had to connect optical barier read out for electricity meter and magnetic Hall probe for gas meter to another pin than GPIO0 and GPIO2.

Programming

Programming of ESP is done via USB-serial. To enter programming mode GPIO0 needs to be low on boot. This can be done by attaching an external switch. Automatic method of connecting Reset to DSR and GPIO0 to DTR line requires no manual reset, but sometimes did not work for me. Probably, because I did not have pull-up resistors connected. Higher speeds than 115200 sometimes worked, but sometimes did not write flash correctly. Without pull-up resistors board was resetting approximately once an hour.



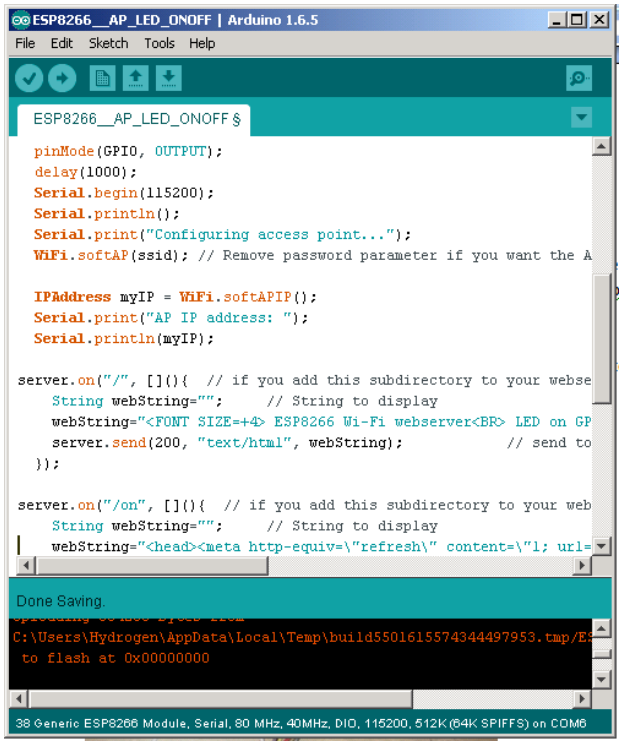
USB-serial port chip does not produce 3.3V reliably even with shorted-out 400 mA fuse and 1000uF capacitor. It is possible to use a Li-Po and a diode in series to decrease voltage from 4.2V to 3.5V. This diode will also give reverse-polarity protection.

Two diodes in series can be used to decrease USB voltage from 5V to 3.6V that is the maximum operating voltage of ESP8266.

A 3.3 V LDO chip can be used, for example, **AS1360** self-consumes only 1 uA of current.

Switch ON/OFF light using your mobile phone over Wi-Fi

Below is a description for ESP8266 WiFi webserver to control a LED attached to GPIO2 via a website.
ESP library has an example for webserwer. ESP as an access point that generated access point something like "ESP8266" and webpage can be browsed at
http://192.168.4.1
It is not convenient to enter such numbers manually, but could not find a better solution. ESP board can be 3.3V battery-driven that makes it a nice demo to show to friends how it is possible to control lights over Wi-Fi (Internet). Works great! See Youtube video:
www.youtube.com/watch?v=uj8UPZ-Qekw



My real need would be to make IP-controlled power sockets to control apartment heating from internet.
For that would need to implement saving the GPIO state to flash, which can be done using EEPROM command similar to on Arduino.

You might want to check http://Blynk.cc They have an app for smartphones that allows to configure ESP pins over internet without need to reflash every time. As communication goes through their server, allows to control ESP board through firewalls. Unfortunately I could not use for anything more than a demo because ESP lost values when Wi-Fi disconnected and did not restore them automatically.

RGB LED stripe “NeoPixels”, individually addressable pixels

Time before Christmas is a typical season for experimenting with lights for decorations.

This year learned how to program "NeoPixel" LED stripe and wanted to share with you.
Bought on Ebay 5 m, 120 pixels, ca 20 EUR. There are LED-spacing and waterproof options.

WS2812B 5050 RGB LED Strip 5M 150 300 Leds 144 60LED/M 5V Individual Addressable

$10.90 to **$42.90**

Buy It Now

Free international shipping

56+ sold

From China

Top-rated seller

Each pixel WS2812 contains RGB LEDs and a controller that receives data serially by a single wire, keeps 3 bytes for himself and the rest of data transmits to the next pixel. So the pixels are individually addressable.

Neopixels (WS2812B) + Arduino (Adafruit library example) worked great!

Thanks to Adafruit for making library for Arduino! Downloaded it, placed in the libraries folder, restarted Arduino IDE and it worked immediately. USB could not supply not more than 32 pixels with 5V power.
In one example there are preprogrammed 9 light effects that can be chosen by a pushbutton.

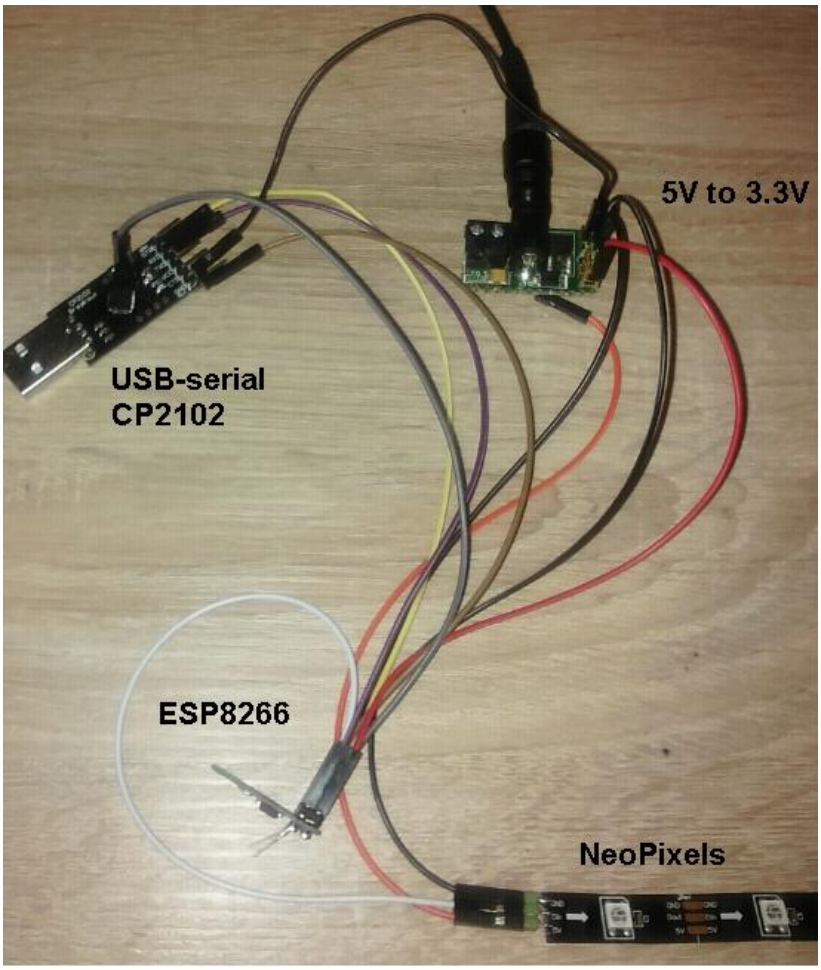
By cutting strips one can make a 2D screen or 3D LED cube. Also light effects for bicycle wheels.

Next thing we will merge Neopixels code with the ESP8266 board and control light effects over Wi-Fi from a smartphone.

Neopixels + ESP8266 with effects switched from website

Burned Adafruit NeoPixels example onto ESP8266. It worked immediately. LED stripe shined correctly colors, but there was some blinking effect probably timing was not perfect. Later fixed this disturbing blinking after I removed lines about using pushbutton from the example.

In the webserver there are links for effects 0…9, when a link is pressed, a corresponding effect is executed.
Youtube:
https://youtu.be/LyBXNlCwjS0

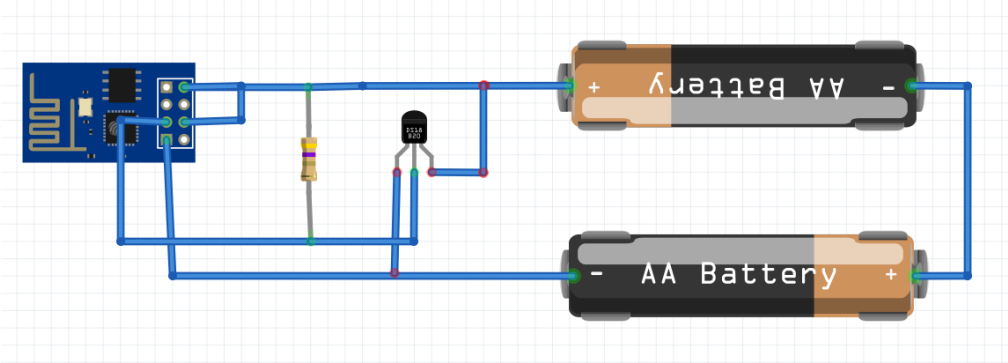


A following problem is observed: while LED animation is moving, the webserver is not responding.

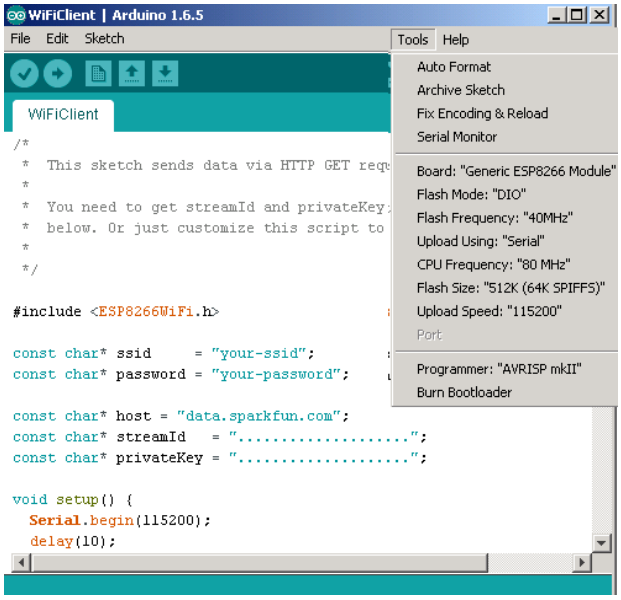
A friend told that he has seen recently a BMW car mod with RGB LED ring around headlights that can be color-adjusted via a mobile phone.

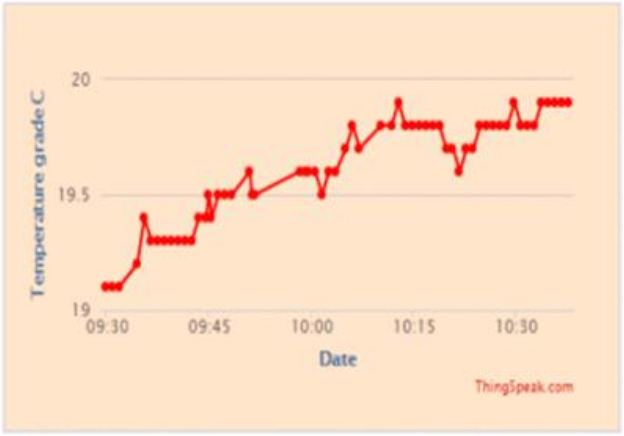
ESP8266 Wi-Fi thermometer storing to Internet cloud

Digital thermometer DS18B20 can be attached to the GPIO2 of ESP8266. First it is a good idea to reproduce an existing example:
http://iot-playground.com/2-uncategorised/41-esp8266-ds18b20-temperature-sensor-arduino-ide



DS18B20 sensor uses the same OneWire library as used for Arduino.
There is a nice example for uploading data to Sparkfun. I modified code to upload to *ThingsSpeak* or Xively. *Thingspeak* allows to perform mathematical manipulations with the data on the server side. For example, add to add togather electricity meter immediate power consumption into daily consumption.





Next step would be to implement battery operation and deep-sleep mode where power consumption is reduced below 0.1 mA.

Deep-sleep power saving mode on ESP8266

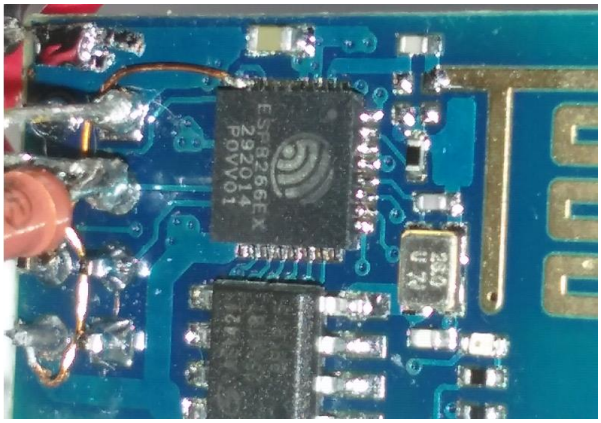
This section addresses the deep-sleep mode on ESP8266 module allowing to make a battery-powered WiFi thermometer with a DS18B20 sensor uploading to Internet cloud.

Construction

1) In deep-sleep mode the main quartz is shut down and only a slow oscillator runs that produces a pulse on the 'post-sleep-reset-pin' XPD_DCDC (pin nr 8 of the ESP chip) after a desired time. This pin can be connected to the reset pin of the module. On the ESP-03 module there are two pads for soldering a bridge.
On ESP-01 module one needs to solder a thin wire between the pin nr 8 of the ESP chip (at the corner of the chip) and the reset pin of the module. This requires a thin soldering iron tip, experience, good eyes and a steady hand. Enameled transformer wire of 0.1-0.2 mm diameter is heated in a droplet of solder until the enamel burns off and wire gets tinned nicely.
With this modification, the module is able to make use of the deep-sleep mode and successfully resets and restarts after waking.

2) A second thin wire is used to solder CH_PD pin of the module with the 3.3V pin.

3) Temperature sensor DS18B20 and a 3.3k pull-up resistor can be soldered directly to the ESP8266 board pins:
GND, GPIO2 and CH_PD (soldered to 3.3V in the previous step).
DS18B20 with a pull-up resistor seems not to contribute to the power consumption in sleep mode.



4) To further reduce consumption the red power LED is also removed by prying it off with a tiny screw driver.

5) Now we have made a thermometer board that can be programmed and placed in a box with two AA cells.

Deep sleep program description

In the Arduino IDE program a following line is added where we want to put controller to sleep:

```
ESP.deepSleep(300000000, WAKE_RF_DEFAULT); // Sleep for 300 seconds
```

The time passed to ESP.deepSleep is in micro seconds, not milliseconds. For one second you would have to use 1000000.

The program first reads out the sensor value, then connects to a router and sends away the value to cloud. During this time the blue on-board LED blinks irregularly.
For debugging it is important to look at serial communication which looks like this:

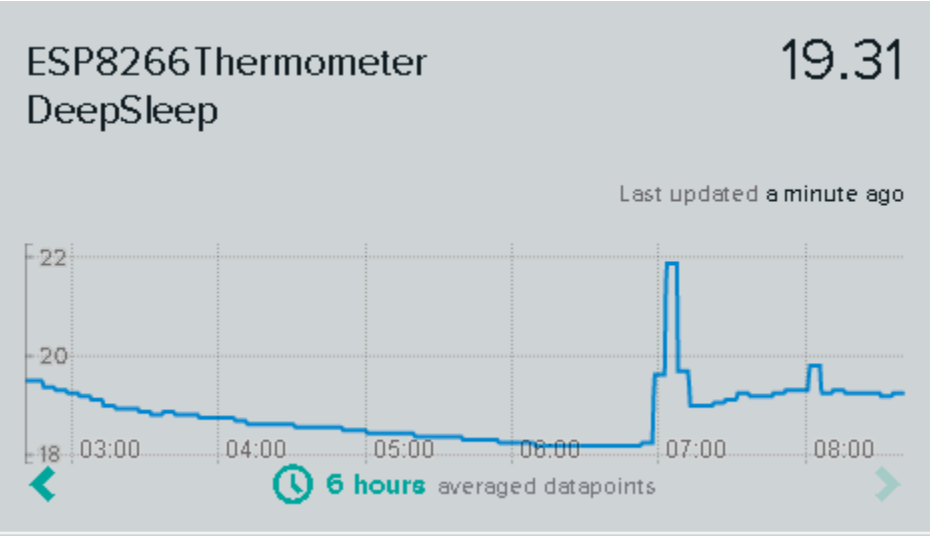
```
Chip = DS18B20  Data = 1 A1 1 4B 46 7F FF F 10 D9   Temperature = 26.06

Connecting to: "LMT-89B0","331M372E70J"...
.............................
WiFi connected IP address: 192.168.1.210

Post string: {"version":"1.0.0", "datastreams": [{"id": "ESP8266ThermometerDeepSleep",
"current_value": "26.06"}]}
String length: 102
HTTP/1.1 200 OK
Date: Sun, 06 Dec 2015 22:58:00 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 0
Connection: keep-alive
X-Request-Id: dff45a7367d2886b389a9e2ef9271c58ada61066
Cache-Control: max-age=0
Vary: Accept-Encoding

Sleeping...
```

I prefer Xively because it generates a png picture that can be saved or embedded in another webpage. Disadvantage of ThingsSpeak javascript plot is that horizontal axis are not updated when new data are not arriving.
https://api.xively.com/v2/feeds/2023977927/datastreams/ESP8266ThermometerDeepSleep.png?width=702&height=250&colour=F15A24&duration=10hours&detailed_grid=true&show_axis_labels=true&timezone=Riga



Power consumption during operation and deep sleep

The program is running 10-15 seconds with average power consumption of 75 mA. Largest time consumption is connection to a router.

During deep-sleep the ESP module consumption was measured to be about 30...70 microamps.
DS18B20 sensor with the pull-up resistor was not measurably contributing to the power consumption.

Normal AA batteries have a capacity of ca 2500mAh. The expected AA battery lifetime is ca 3 months waking up once in 5 minutes. This is once or twice during winter season, which is probably OK. By increasing the cycle time to 10 minutes one would sacrifice time resolution, but would need to change batteries only once per winter.

A better idea is to use rechargeable batteries. 2AA NiMH would not give enough voltage.

A Li-Po could be recharged via USB, but would need a step-down regulator to 3.3V and deep-discharge protection circuit. Discharge protection is sometimes built into the Li-Po cell of mobile phones.

3.3 V LDO chip **AS1360** self-consumes only 1 uA of current. Instead of using a 3.3V LDO one can use a single diode in series to decrease Li-Po max voltage 4.2V to 3.5V. This will also give reverse-polarity protection.

Two diodes in series can be used to decrease USB voltage from 5V to 3.6V that is the maximum operating voltage of ESP8266.

Due to connection time to the router a Wi-Fi thermometer will never be as power-saving as a 432 MHz thermometer that just transmits data during 100 ms once a every minute and can run on a set of AA batteries for several years.

Inside the home in many places it is actually not necessary to run on batteries, one could use a switching 220V to 3.3V DC power supply and such Wi-Fi sensors practically would not contribute to the electricity bill compared to 0.3W by ESP chip, if deep-sleep mode is not implemented.

Advanced Wi-Fi thermometer configurable via a webpage

For consumer-grade gadgets SSID and login details should be possible to change by booting the gadget in access point mode and connecting to it via a web browser.

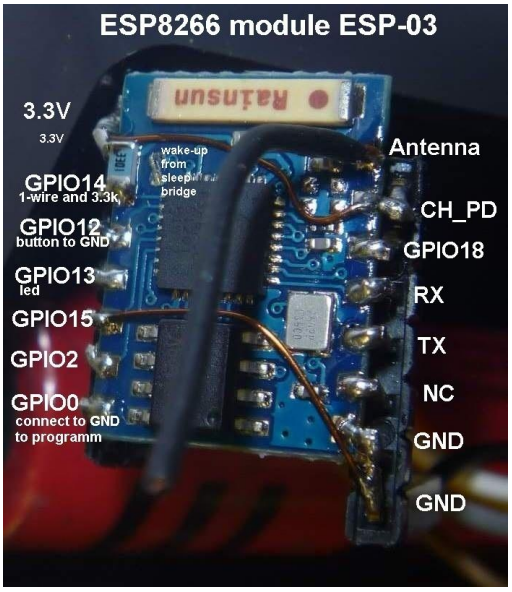
Below is the final version. Plastic enclosure hosts ESP and Li-Po with a diode in series, power switch, a button and a led. Sensor can be put either inside or outside the box. Enclosure dimensions are 80x 45 mm.

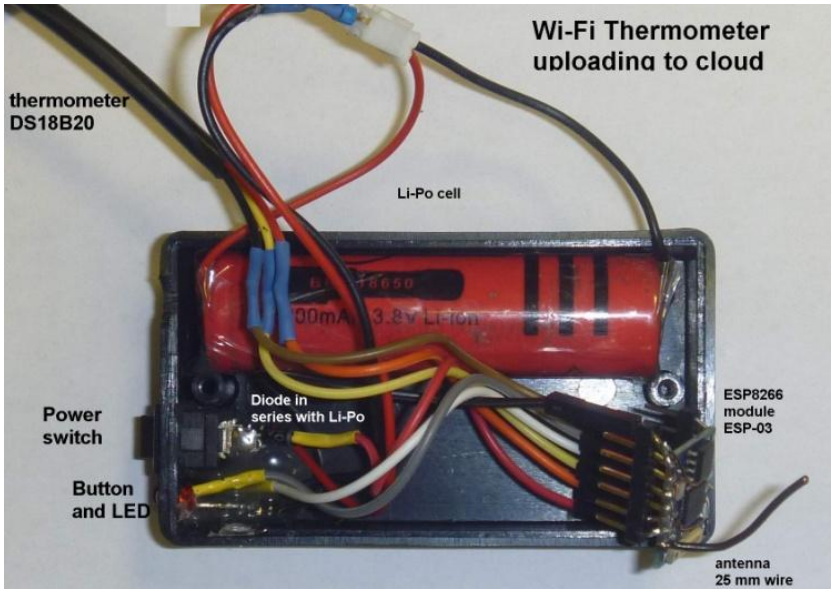
Thermometer with Wi-Fi upload to Internet cloud
ESP8266 Wi-Fi module ESP-03, temperature sensor DS18B20.
Button pressed during power ON - entering access point mode. Webpage at 192.168.4.1
Wi-Fi client uploads temperature to Internet cloud Xively or ThingsSpeak at regular intervals.
LED blinks shortly after a successful upload. Blinks 1s if error has occured.
Li-Po powered via a Si diode to drop voltage to max 3.6 V. Consumes 75 mA while running (ca 10s).
Deep sleep mode implemented. Consumption 30 uA.

Construction

ESP-03 module version was used in this project as it has more available GPIO pins compared to ESP-01.

- ESP-03 has no ADC pin and no Reset pin routed from the chip to the board.
- A bridge needs to be soldered to lead wakeup timer to reset the board from deep sleep.
- Solder gpio15 to gnd.
- Solder CH_PD to 3.3V.
- No red led.
- No blue led.
- Chip antenna. People wrote in a blog that a 25 mm long wire soldered to antenna connector increase signal strength.





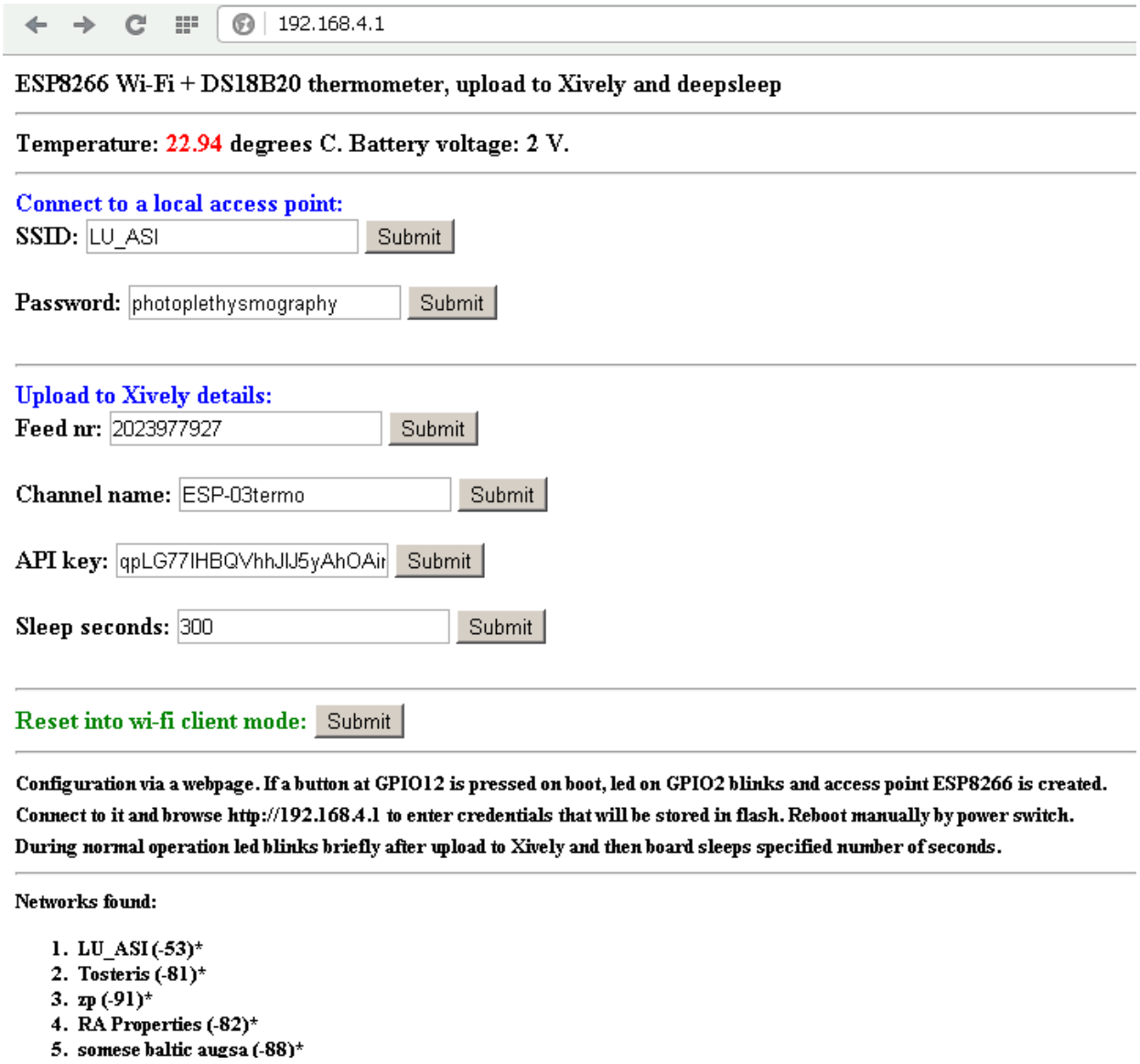
A button is added to one of GPIOs. If the button is pressed during powering on the board it starts access point. If the button has not been pressed then ESP reads out temperature sensor, starts Wi-Fi client, composes data string and uploads it to the Internet cloud, for example Xively or ThingsSpeak.

A LED is added to indicate a successful upload. The board then disconnects from the AP and sleeps in low power consumption mode for specified number of seconds. A piezo-speaker could be attached to warn about low battery voltage like it is done in smoke-alarms.

Configuration website

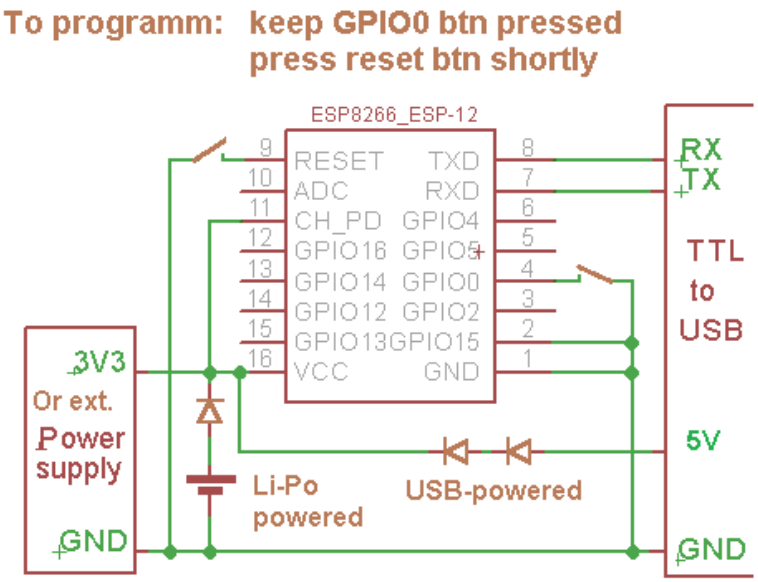
If the button is pressed during the power-up an access point names for example ESP8266 is created and one can connect via a web browser. From the web page several settings can be configured and stored permanently . This is very convenient when moving thermometer from one internet AP to another. Forms are pre-filled with data from flash EEPROM.

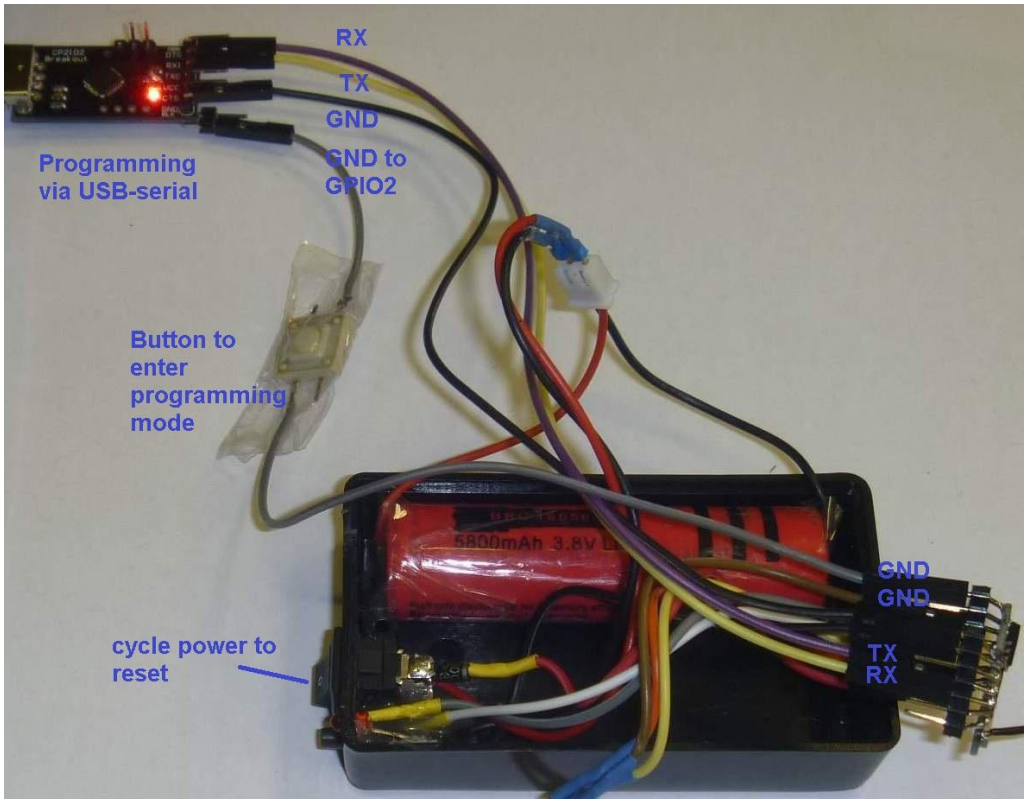
ESP8266 has no real EEPROM, but data can be stored permanently to flash using *EEPROM.write* command similarly like using Arduino. Actually data are stored in RAM and written to flash only with EEPROM.commit(); command.



Programming

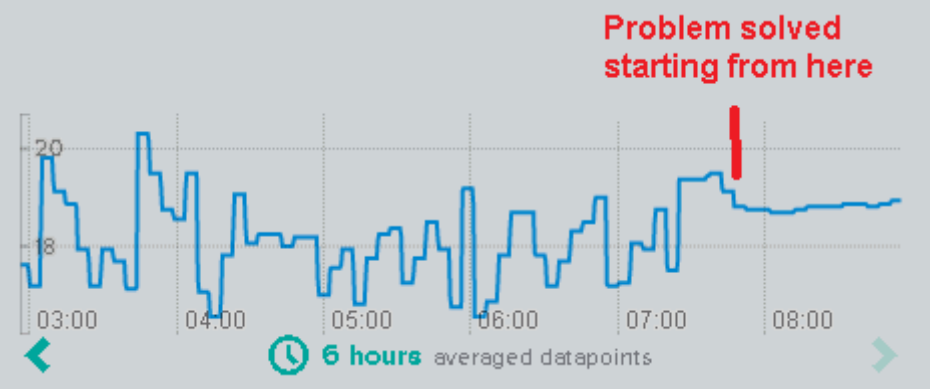
Programming was done via a USB-serial adapter. To enter programming mode GPIO0 was connected to gnd by a pushbutton during powering the board up. Probably it is a good idea to add a second pushbutton to reset to wake up board from deep sleep.





Troubleshooting

Temperature sensor readout should be performed before the Wi-Fi transmitter is started. Or it might look extra noisy like in the plot below.



Outlook

Surfing Internet brought me to Rayshobby website and I became a fan of **ESPtoy** board for 15$:
http://rayshobby.net/cart/esptoy
It allows to program ESP via USB and also charges a Li-Po from USB. It has onboard RGB LED and one line pin header to plug into breadboards.
ESPtoy is open source and both code and Eagle design files available at github.
https://github.com/rayshobby/ESPToy
Thanks to the Rayshobby for making available ingenious design!

In the attachment is ESP8266 code for
1) ESP8266 AP webserver allowing to switch ON/OFF a LED on GPIO2.
2) ESP8266 + NeoPixels on GPIO2 control via webserver and smartphone.
3) ESP8266 + DS18B20 thermometer sending to ThingsSpeak.
4) ESP8266 + DS18B20 thermometer sending to Xively with deep-sleep mode.
5) AP credentials entering from webpage and and storing into EEPROM.

Will be happy for your feedback. Author, Janis Alnis. 2015.12, Riga, Latvia.