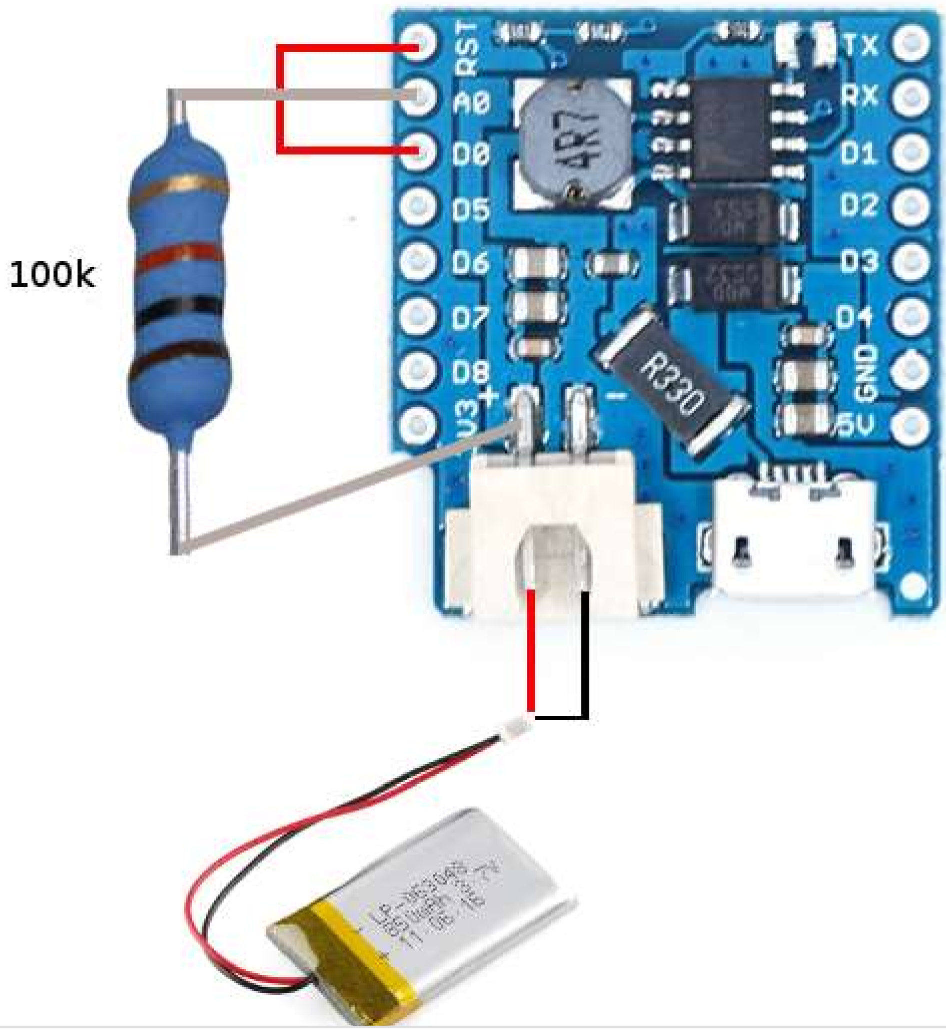
Arduino, ESP8266, ESP32 & Raspberry Pi stuff

Arduino and related stuff (including Attiny and ESP8266) and the Raspberry Pi

Monitoring LiPo battery voltage with Wemos D1 minibattery shield and Thingspeak

There are a million reasons why you would want to monitor the Battery voltage of your Battery fed ESP8266. I will illustrate it with a Wemos D1 mini and the Battery shield



I am using a small 720 mAh LiPo cel. If I just leave the Wemos access the internet continuously it will last 6.5 hours, but for this example I will put the Wemos in Deepsleep for a minute, then read the battery voltage and upload that to Thingspeak. You only need

First, connect RST with GPIO16 (that is D0 on the Wemos D1 mini). This is needed to let the chip awake from sleep.
Then connect the Vbat through a 100k resistor to A0.

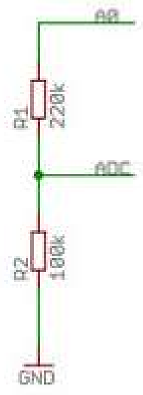
So why a 100 k resistor?

Well the Wemos D1 mini already has an internal voltage divider that connects the A0 pin to the ADC of the ESP8266 chip. This is a 220 k resistor over a 100 k resistor

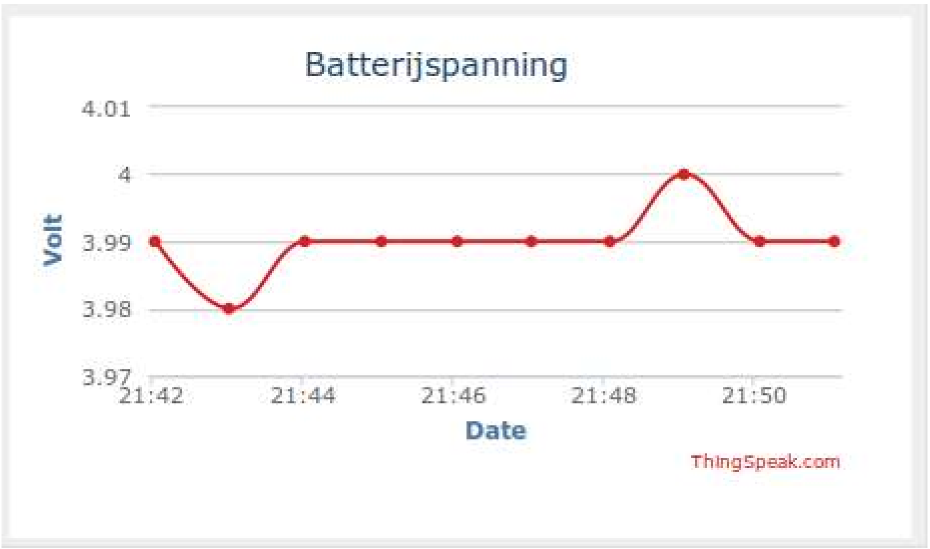
By adding a 100k , it will in fact be a total resistance of 100k+220k+100k=420k.
So if the Voltage of a fully loaded Cell would be 4.2 Volt, the ADC of the ESP8266 would get 4.2 * 100/420= 1 Volt

1 Volt is the max input to the ADC and will give a Raw reading of 1023.

The True voltage then can be calculated by:
raw = AnalogRead(A0);voltage =raw/1023;
voltage =4.2*voltage;
Ofcourse you could also do that in one step, but I like to keep it easy to follow.



Wemos D1 Internal Voltage divider



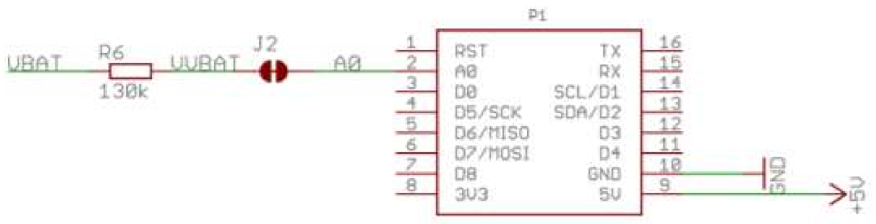
If you do use this possibility, do realise that the resistors drain the battery as well with a constant 10uA (4.2V/420 000ohm). The powerconsumption of an ESP8266 in battery monitor this would be 87uA, which is a sizeable increase. A solution could be to close off the Vbat to the A0 with a transistor, controlled from an ESP8266 pin

A program could look like this:

```
/*
 * Wemos battery shield, measure Vbat
 * add 100k between Vbat and ADC
 * Voltage divider of 100k+220k over 100k
 * gives 100/420k
 * ergo 4.2V -> 1Volt
 * Max input on A0=1Volt ->1023
 * 4.2*(Raw/1023)=Vbat
 */

// Connect RST en gpio16 (RST and D0 on Wemos)
#include <ESP8266WiFi.h>
unsigned int raw=0;
float volt=0.0;
// Time to sleep (in seconds):
const int sleepTimeS = 60;

void setup() {
  Serial.begin(115200);
  Serial.println("ESP8266 in normal mode");
  const char* ssid     = "YourSSID";
  const char* password = "YourPW";
  const char* host = "api.thingspeak.com";
  const char* writeAPIKey="YourAPIkey";
  // put your setup code here, to run once:
  pinMode(A0, INPUT);
  raw = analogRead(A0);
  volt=raw/1023.0;
  volt=volt*4.2;
//  Connect to WiFi network
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  }
  String v=String(volt);// change float into string
  // make TCP connections
  WiFiClient client;
  const int httpPort = 80;
  String url = "/update?key=";
  url += writeAPIKey;
  url += "&field6=";// I had field 6 still free that's why
  url += String(volt);
  url += "\r\n";

// Send request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");

 //Sleep
   Serial.println("ESP8266 in sleep mode");
   ESP.deepSleep(sleepTimeS * 1000000);
}

void loop() {
  //all code is in the Setup
}
```

The new battery shield

There now is a new version (V1.2.0) of the battery shield that has an inbuilt resistor connecting A0 to the battery, through Jumer J2 (‘Jumper’ being a big word for 2 solderpads), so if you want to measure the battery voltage, all you need to do is to put some solder connecting J2
However, rather than a 100k resistor, a 130k resistor was used. The voltage divider thus becomes 100/(130+220+100), so for a full reading of 1023 (=1Volt on A0) a total of (1/100k)*(130+220+100)=4.5Volt would be necessary.
In reality the Lipo cell will not give off 4.5Volt.



ADVERTISING