IoT with ESP8266

Esp8266 blog. Learn how to compile, how to work with the wireless chip esp8266. ESP-01 ESP-03, ESP-07, ESP-12

Thursday, December 21, 2017

Very good offer for books and videos from Packt Pub. On eBook $5

Packt Publishing has a new offer, for $5 you can get a lot of eBooks and videos.

Click here for a list. There are over 4000 books and over 1000 videos to choose from.

at 9:13 AM

Reactions: interesting (0) cool (0)

No comments: Links to this post G+

Labels: esp8266 book, free book ESP8266, packt pub

Saturday, December 2, 2017

New book on how to made a complete ready-to-sell home automated system with ESP8266

I've been busy in the latest months writing a book to cover some key aspects of the ESP8266 ecosystem, the chip, the cloud and a mobile application.

The book will teach you and will give you a ready-to-sell solution for an IoT product.

You will discover how to work with the GPIOs on the ESP8266, how to build your basic thermostat for your house, how to control it from your mobile with your own cloud system based on MQTT.

Securing the data using authentication at the broker level and SSL is explained in a special chapter.

Real-time communication has a dedicated chapter where you will learn how to send real-time data from an ESP8266 to an nodejs server.

You can buy the book from Amazon.

I hope that you will enjoy the book.

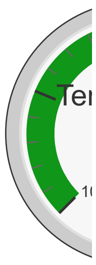
Search This Blog

Banggood

IoT idea

... If you want to devel
email...

Current Temperature



About Me

Catalin B.

I am looking for new
Working in software f
software programmer

View my complete pro

Blog Archive

- ► 2018 (1)
- ▼ 2017 (13)
  - ▼ December (3)
    - Very good offer Pu...
    - New book on ho sell h...
    - BME280 and ES
  - ► November (2)
  - ► October (1)
  - ► September (1)
  - ► August (1)
  - ► July (2)

Catalin Batrinu

ESP8266 Home Automation Projects

Leverage the power of this tiny WiFi chip to build exciting smart home projects

Packt

ESP8266 Home Automation Projects

ESP8266 is a low-cost yet powerful Wi-Fi chip that is becoming a popular choice for building an automated interconnected devices for better living.

With this book, you will learn to create and program home automation projects using the ESP8266 Wi-Fi chip. You will learn how to build a thermostat to measure and adjust the temperature, and how to build a security system using the ESP8266. Furthermore you will learn how to design a complete home automated system from scratch which will allow you to send the values from your ESP8266 modules to your private cloud to monitor your applications.

By the end of the book, you will understand how to completely control and monitor your home from the cloud and from a mobile application. You will be familiar with the capabilities of the ESP8266 and will have successfully designed a complete, ready-to-sell home automation system.

Things you will learn:

- Install and configure an MQTT server
- Use the Wi-Fi connectivity feature to control appliances remotely
- Design a thermostat using ESP8266 to measure and adjust the temperature of your home
- Control and monitor your home from the cloud using ESP8266 modules
- Stream real-time data from the ESP8266 to a server over WebSockets
- Create an Android mobile application for your project

Packt
www.packtpub.com

ISBN 978-1-78728-262-9

at 1:31 AM

Reactions: interesting (0) cool (0)

No comments: Links to this post

Labels: cloud, esp8266 book, MQTT, websockets

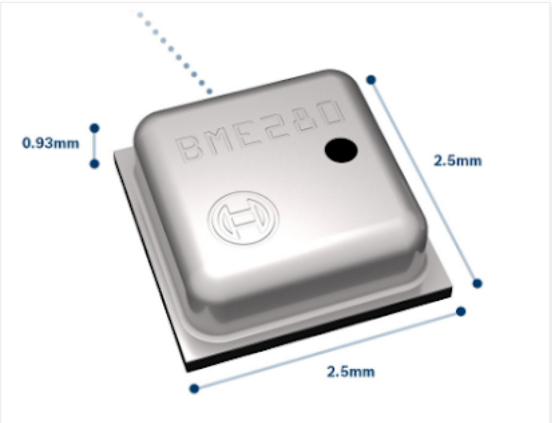
- ► March (1)
- ► February (2)
- ► 2016 (39)
- ► 2015 (38)

BME280 and ESP8266

Latest environmental sensor from Bosh is the BME280 which can measure:

- temperature
- humidity
- pressure

and can be found in mobile phones (Nexus 5). There are rumors that the BMP280 is a BME280 which couldn't be calibrated for humidity, but I have no confirmation on this.

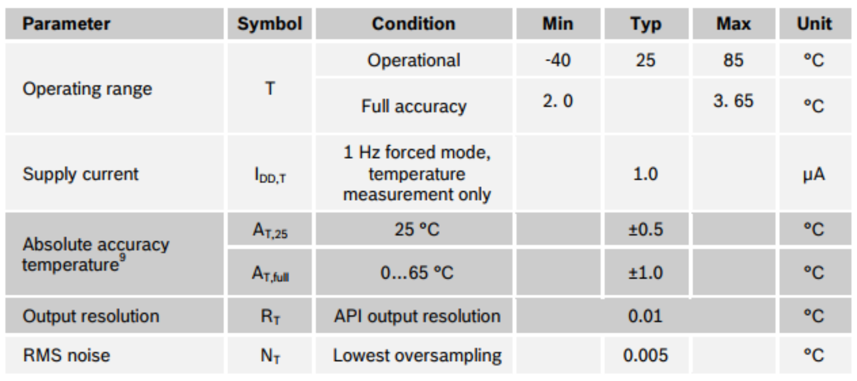


BME280

Can be used for:

- Indoor navigation (based on changing the measured altitude - pressure)
- Outdoor navigation
- Weather forecast
- Home application control
- Context awareness (change room detection)
- Internet of Things.

Temperature precision is ± 1 C degree in 0-60C range.



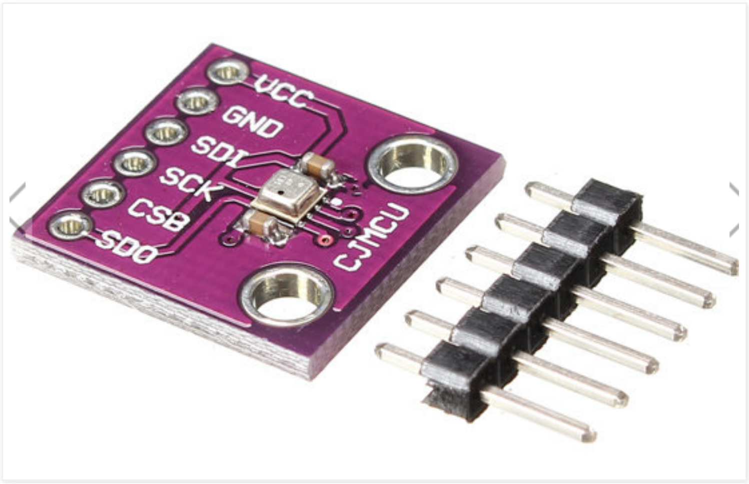
Parameter	Symbol	Condition	Min	Typ	Max	Unit
Operating range	T	Operational	-40	25	85	°C
		Full accuracy	2. 0		3. 65	°C
Supply current	$I_{DD,T}$	1 Hz forced mode, temperature measurement only		1.0		µA
Absolute accuracy temperature[9]	$A_{T,25}$	25 °C		±0.5		°C
	$A_{T,full}$	0…65 °C		±1.0		°C
Output resolution	R_T	API output resolution		0.01		°C
RMS noise	N_T	Lowest oversampling		0.005		°C

Can be found in multiple modules, from SPI connectivity to I2C.

Make sure that if you are using the I2C version to change the I2C address to 0x76. (Default value for I2C address in Adafruit's library is 0x77).

I2C version

The SPI version can be found around USD 5 here.



Code is similar with the BMP280 , just read the humidity

```
/*******************************************************
 * Catalin Batrinu bcatalin@gmail.com
 * Read temperature, humidity and pressure from BME280
 * and send it to thingspeaks.com
*******************************************************/

#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <ESP8266WiFi.h>


Adafruit_BME280 bme; // I2C
// replace with your channel's thingspeak API key,
String apiKey = "YOUR-API-KEY";
const char* ssid = "YOUR-SSID";
const char* password = "YOUR-ROUTER-PASSWORD";
const char* server = "api.thingspeak.com";
WiFiClient client;


/**************************
 *   S E T U P
 **************************/
void setup() {
  Serial.begin(9600);
  Serial.println(F("BMP280 test"));
  if (!bme.begin()) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
  WiFi.begin(ssid, password);

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
}

  /*************************
 *  L O O P
 *************************/
void loop() {

    if (client.connect(server,80))  // "184.106.153.149" or api.thingspeak.com
    {
        String postStr = apiKey;
        postStr +="&field1=";
        postStr += String(bme.readTemperature());
        postStr +="&field2=";
        postStr += String(bme.readHumidity());
        postStr +="&field3=";
        postStr += String(bme.readPressure() / 100.0F);
        postStr += "\r\n\r\n";

        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
        client.print("Content-Type: application/x-www-form-urlencoded\n");
        client.print("Content-Length: ");
        client.print(postStr.length());
        client.print("\n\n");
        client.print(postStr);
    }
    client.stop();
    //every 20 sec
    delay(20000);
}
```

If your readings are with almost 2 degrees more than the expected value is because the BME280 is to close to the ESP8266. Try to keep at least 10 cm between the BME280 and ESP8266 to eliminate the RF heating and heating produced by ESP8266.

Also is possible that you run the BME280 in normal mode (more samples per second) versus forced mode when you are reading the values exactly when you need them (here the drift is around 0.6 degrees Celsius).

The complete datasheet can be found here.

at 1:09 AM

Reactions: interesting (2) cool (2)

2 comments: Links to this post

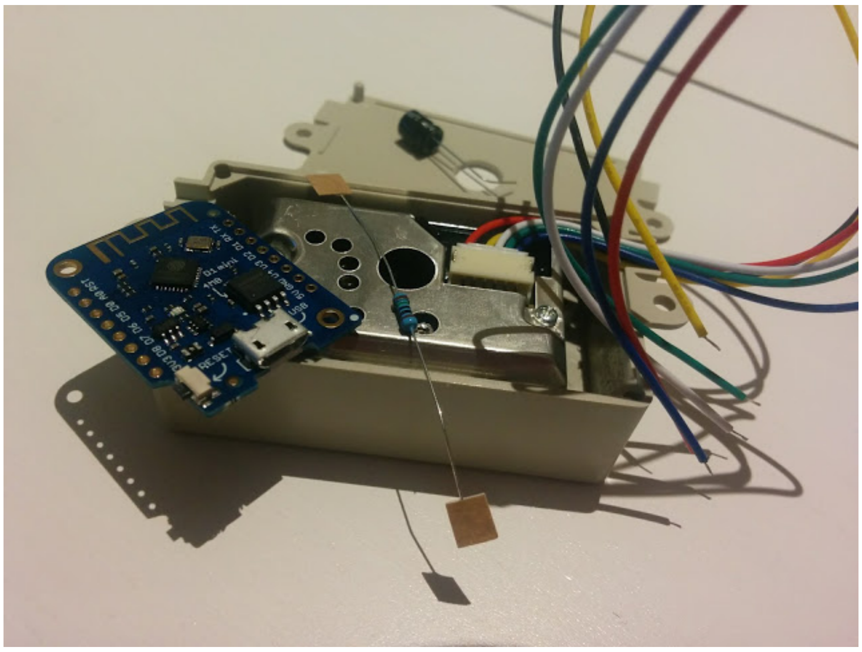
Labels: BME280, BMP280, ESP8266, thingspeak

Sunday, November 12, 2017

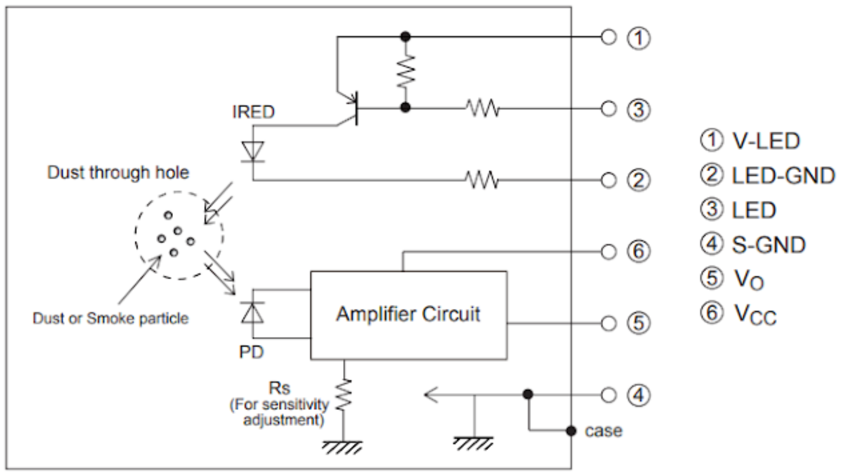
Getting air quality with ESP8266 and Amazon Alexa

If last week I've managed to get the temperature and humidity with ESP8266 and Alexa, now it's time to integrate the air quality sensor GP2Y1010AU0F from Sharp.

The GP2Y1010AU0F it is a compact optical dust sensor that has an infrared emitting diode and a phototransistor that are diagonally arranged. It detects in fact the reflected light of the present dust in the air. It is effective to detect very fine particle like cigarette smoke and it can distinguish from the house smoke from the house dust.


GP2Y1010AU0F sensor.

It has 6 pins and the electronic inside is split in two parts, the IRED emitting part and the phototransistor (receiving part)

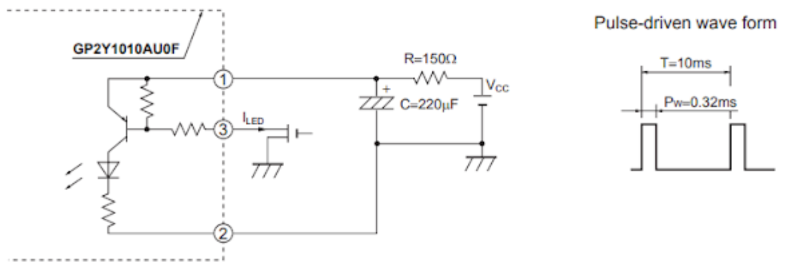


Internal schematic

For the Air quality module I've used an wemos D1 board. The datasheet for the GP2Y1010AU0F is referring as VCC and the provided output by using a 5V power supply. Since the ESP8266 on analog input is limited to 1V and the wemos D1 is having an 220K and a 100K divider, I've choose to power the transmitter part to 5V and the receiver part with 3V3.

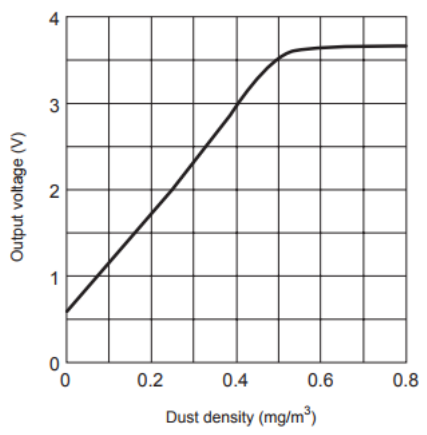
According to the specification in the datasheet also you will need an 150 ohm resistor and a 220uF capacitor, both came with the package.

Connect them according to the datasheet.



Resistor and capacitor

The provided analog output is proportional with the number of the dust particle in the air. Powering the phototransistor part with 3V3 will not get exact the output graph from the datasheet unless you will not use the 5V Vcc. If you need to be 100% to the datasheet and you are using an wemos D1, add a resistor in series with the output VO (pin 5) so the maximum voltage on the A0 input will not exceed 1V. Probably an 180k resistor will be fine.



Output voltage vs dust density

Based on the output voltage you can setup some steps for the air quality like excellent, very good, good, fair or dusty.

Here is a video clip on how is working with iotcentral.eu platform and Amazon Alexa.

Now eNVi-A is online. Total cost: 18.50 euro.

at 5:08 PM

Reactions: interesting (0) cool (0)

2 comments: Links to this post G+

Labels: air quality, amazon alexa, ESP8266, iotcentral.eu, Sharp

Thursday, November 9, 2017

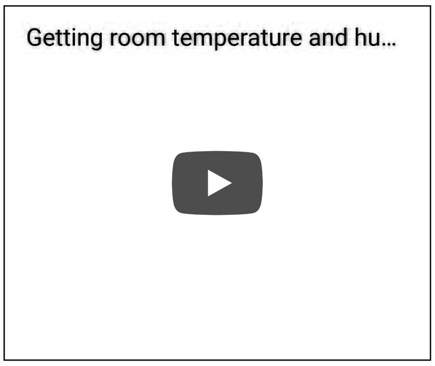
Getting temperature from ESP8266 over MQTT with Amazon Alexa

With the help of the Iotcentral.eu platform now I can get room temperature or outside temperature with just an ESP8266 and a temperature sensor like DHT22 or DS18B20 or BMP280.

The Alexa is questioning the iotcentral.eu server that will ask my local broker that will publish a message the the ESP8266 to ask the temperature for my room. The values for temperature and humidity will travel the other way around to Alexa, that is so nice and is letting me know about them.

On Iotcentral.eu you will find also a demoapp (source code on git) for a plug, but you can convert it easy to other IoT device. Also there is a mobile application in Google Play named Homy4(source is on git) that connects to the iotcentral.eu and allows you to turn on/off the plug device from demoapp.

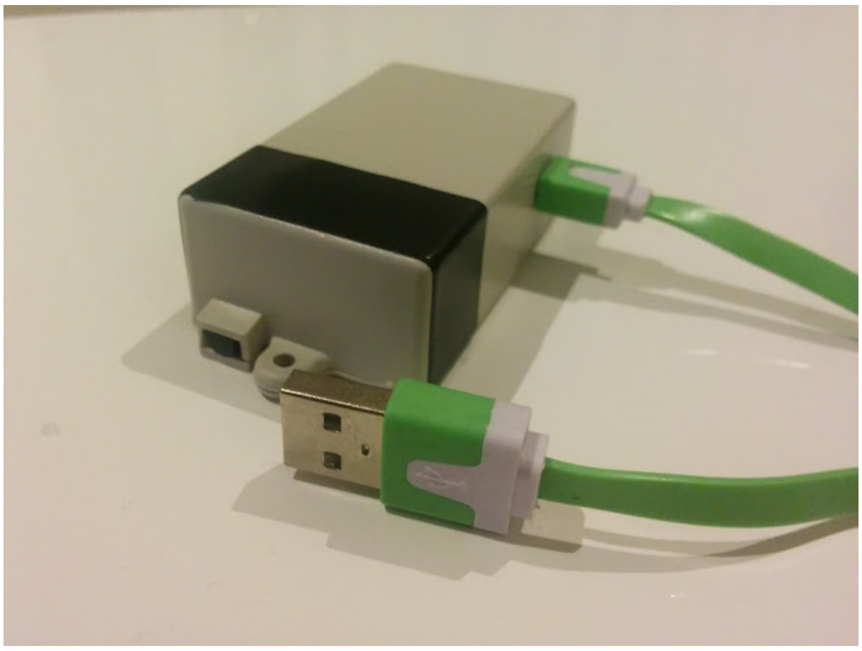
Let me know what you think and what modules do you want to integrate on iotcentral.eu, on Alexa, ESP8266 and on the mobile application.



Everything is done encrypted and authenticated against 2 servers, so the entire communication is secured.

I will publish soon this skill to Amazon so any of you can have it. Just use the demoapp from iotentral.eu and adapt it to send temperature. Maybe will be a good option to create also a demoapp for the temperature, not only for a plug.

The final box is small and compact. I've choose the DS18B20 since it can be integrated well in the box I have. To bad that there are not some enclosures for wemos D1 or nodeMcu which are most popular boards on the market. A printed 3D one it is an option, but takes time and I don't know if they can be produced in a large number.



Final box with eNVi - Alexa thermometer

So now the eNVi-T is up and running in 3 rooms. Next will be an air quality module and maybe an thermostat.

Total cost:10 euro.

at 12:09 AM

Reactions: interesting (0) cool (0)

2 comments: Links to this post

Labels: Alexa, banggood, BMP280, DHT22, DS18B20, ESP8266, temperature

Monday, October 2, 2017

MQTT Broker on ESP8266

Running an MQTT broker to serve all devices for a house (20pcs) it requires at least a Raspberry Pi or an equivalent SOC.

SD card fails from time to time and requires some Linux skills to make it work and the final price is over USD 70. (Good power supply, case, HDMI cable, keyboard)

But how about running an MQTT broker on this USD 3.99 ESP8266 board ? It is possible ? Yes, it is ! Go to http://iotcentral.eu and with few clicks you will be able to flash your ESP8266 (4Mb) with the binary.

No need to compile code, to solve errors , just click and run. More than that your MQTT broker will communicate with your http://iotcentral.eu instance so you still get your messages on your phone if your mobile app is connecting to the websockets to your http://iotcentral.eu cloud instance.

But what I am at home and my internet connection is down ? Can I communicate with my devices or I am isolated? Don't worry, your mobile app will still be able to connect to the ESP8266 MQTT Broker over the websocket so you can control your devices.

at 11:14 PM

Reactions: interesting (0) cool (0)

No comments: Links to this post

Labels: broker, ESP8266, iotcentral.eu, mosquitto, MQTT, websocket

Monday, September 18, 2017

Control any TV with your voice

Few months ago I was writing about the Broadlink RM3 mini in this article.

The big news is that nows supports the Amazon "Alexa" voice control.

So saying "Alexa, change to channel 77 on Television" will change your TV to channel 77.

You can also rename your favorite channel to a specific name and call it like this: "Alexa, change to channel Sport on Television" and the Broadcom RM3 mini will do that for you.

To have all this features you need to install this application from Google Play (I don't know for iPhone) and make sure you have the latest version of firmware on your Broadlink RM3 mini.

If you have an STB Like I have and you are using your TV just as a panel you need to assign the virtual remote buttons to your STB remote buttons. After that everything will work fine.

The last step is to add the "BroadLink Remote Control" skill to your Alexa app and do start a device discovery.

BroadLink RM3 mini

What I miss from the previous e-Control application is the possibility to share the configuration of devices with other phones.

And a video (sorry for accent :-)) for demo.

at 4:23 PM

Reactions: interesting (0) cool (0)

No comments: Links to this post

Labels: Alexa, Amazon Echo Dot, amazon esp8266, BroadLink Alexa

Tuesday, August 1, 2017

ESP8266 anti cloning system

If you are investing time and money in developing a new module you want to have confidence that no one will clone it.

Assuring the authenticity of a product is a good for your brand image. No one wants that his nice and cheap power socket to also mine bit coins for someone else or do a DoS attack based on an external command. So, be aware on cheap IoT devices.

Since the flash memory of ESP8266 is external to the MCU then you will need an external encryption chip (ATSHA204 EUR 0.5) that helps to authenticate your module.

Features:

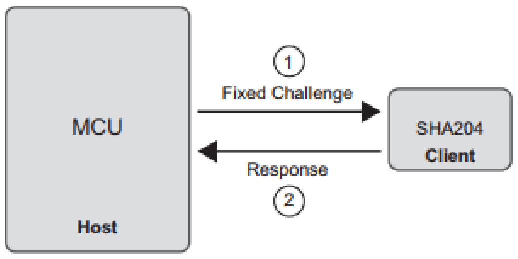
- Crypto Element with Protected Hardware-based Key Storage
- Secure Symmetric Authentication Device Host and Client Operations
- Superior SHA-256 Hash Algorithm with Message Authentication Code (MAC) and Hash-Based Message Authentication Code (HMAC) Options

- Best-in-class, 256-bit Key Length; Storage for Up to 16 Keys
- Guaranteed Unique 72-bit Serial Number
- Internal, High-quality Random Number Generator (RNG)
- 4.5Kb EEPROM for Keys and Data 512 bit OTP (One Time Programmable) Bits for Fixed Information
- Multiple I/O Options–UART-compatible High-Speed, Single-Wire Interface–1MHz I2 C Interface
- 2.0V to 5.5V Supply Voltage Range
- 1.8V to 5.5V Communications Voltage Range
- <150nA Sleep Current
- Secure Download and Boot–Ecosystem Control–Message Security–Anti-Cloning
- 8-lead SOIC, 8-lead TSSOP(2), 3-lead SOT23, 8-pad UDFN, 8-pad XDFN, and 3-lead CONTACT Packages

Applications:

- Secure Download and Boot
- Ecosystem Control
- Anti-cloning
- Message Security

Basically the ESP will compute a challenge and pass it to the encryption chip. Solution for the challenge is then compared with the expected one and if both matched then the board is authentic. The ESP8266 will not start if the challenge is not solved correctly.



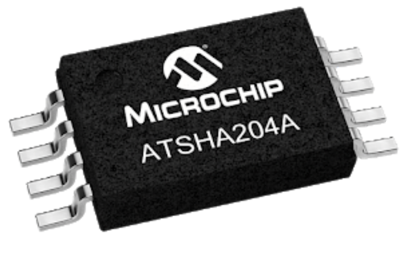
ATSHA204 Challenge Response

The chip I've used is the ATSHA204 and it has I2C and SWI as interfaces.

You can read the Application Notes here.

SWI interface

And the I2C version.

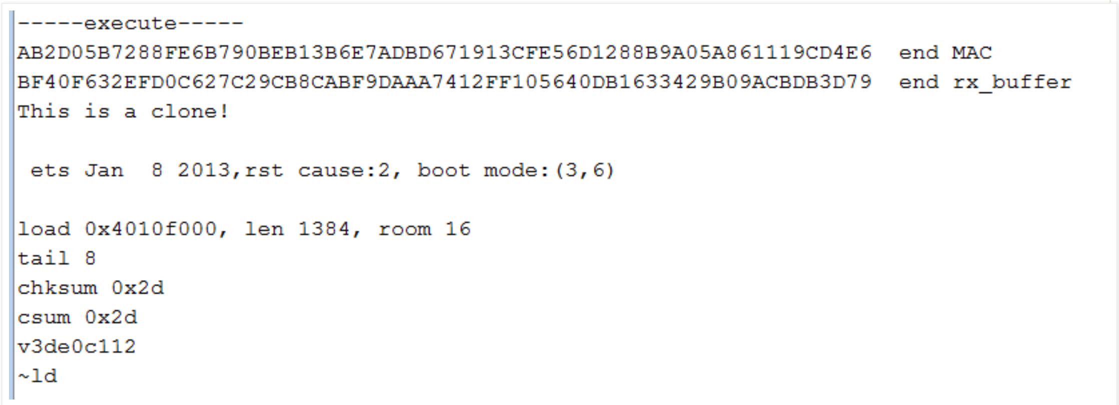


I2C version

First check if some how the chip is not altered before arrives to you. Check the default registers values with the default values from the manual, so between factory and you nobody altered the chip.

Here you will find all documentation related to the chip, how to configure it, how to debug it and more.

Trying to clone an ESP8266 module:



```
-----execute-----
AB2D05B7288FE6B790BEB13B6E7ADBD671913CFE56D1288B9A05A861119CD4E6  end MAC
BF40F632EFD0C627C29CB8CABF9DAAA7412FF105640DB1633429B09ACBDB3D79  end rx_buffer
This is a clone!

 ets Jan  8 2013,rst cause:2, boot mode:(3,6)

load 0x4010f000, len 1384, room 16
tail 8
chksum 0x2d
csum 0x2d
v3de0c112
~ld
```

Anti-cloning system in action

at 3:53 PM

Reactions: interesting (0) cool (0)

2 comments: Links to this post G+

Labels: anti cloning system, ESP8266 clone, SHA256

Sunday, July 23, 2017

18650 Li-Ion battery powered Wemos D1 with Wemos Battery Shield - DONE 48 days achieved

>>> For the version with a solar panel attached see this post <<<

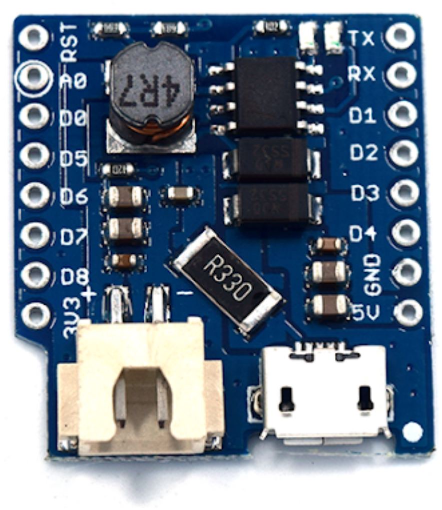
Wemos has developed a series of shields for the Wemos D1 :

- DHT Shield
- Battery Shield
- Matrix LED Shield
- Buzzer Shield
- Dual Base
- SHT30 Shield
- WS2812B RGB Shield
- ProtoBoard Shield
- 1-Button Shield
- Micro SD Card Shield
- Relay Shield
- DC Power Shield
- Tripler Base
- Motor Shield
- OLED Shield

The good thing is that you can stack them on top of another.

Let's take a look for now to the Battery Shield that is now on version 1.2.0 and it has a retired version 1.1.0

Get the schematics from here version 1.1.0 and 1.2.0 to see the differences. An immediate visual difference is the inductor on the V1.1.0 is much higher than the one on V1.2.0 but is not all.



Battery Shield V1.1.0

Since I didn't found yet the version 1.2.0, I used the V1.1.0 from here. and the Wemos D1 mini from here. I've also added an BMP180 pressure and temperature sensor.

Characteristics:

- Charging Voltage: max: 10V, recommend: 5V
- Charging Current: max: 1A
- Lithium Battery voltage: 3.3-4.2V
- Boost Power Supply: 5V(max: 1A)

Connections between the Wemos D1 mini and Battery shield:

D1 mini	Shield
5V	5V(max: 1A) Power Supply
GND	GND
A0	Vbat f the jumper J2 is closed.

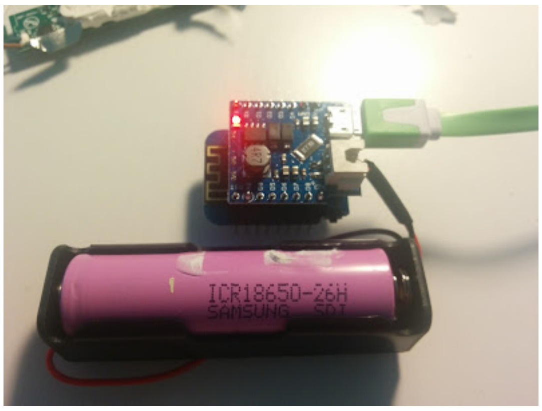
Now for the battery. Since I didn't find a decent LiIon flat battery on a good price I had to improvise and I've dismounted a EUR 5 Innergie 2600mA power bank.

Original Power Bank

I have disassembled the power bank and I've get the battery from it.

Samsung 18650 Li-Ion battery. Nice !!!

Then I've added the battery to this EUR 1 battery holder from tinytronics.nl.

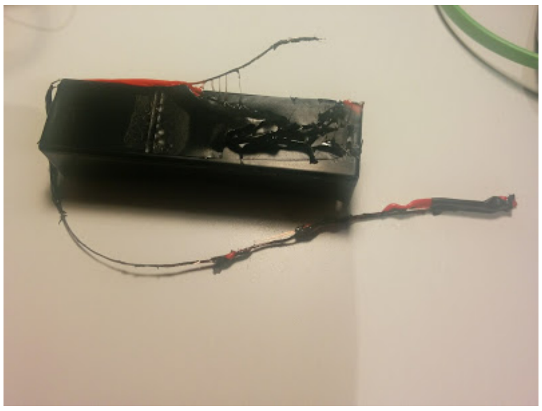


Battery is charging

I've connected the Battery Shield's miniUSB to 5V to charge the battery and the red LED start to light. When the battery was charged the LED red turned off and LED green turned ON signaling that the charging is done. After I've removed the USB cable from the Battery Shield the green LED turned off not to consume from battery. At this moment I've added the BMP180 and flash the firmware.

BE AWARE OF SHORTING THE BATTERY WIRES. ALWAYS DOUBLE CHECK THEM..

This is happening when the battery is shorted. Don't try it at home. This was done so you not have to try it.


Melted wires and battery holder.

To log the battery level I've used again the good service offered by the thingspeak.com and log the A0 value, a computed voltage which is not accurate at this time since I didn't had a voltmeter with me to do it.

```
#include <ESP8266WiFi.h>

#include <Wire.h>

#include <Adafruit_BMP085.h>


Adafruit_BMP085 bmp;
```

```
unsigned int raw=0;
float volt=0.0;
String apiKey = "YOUR_THINGSPEAK_API_KEY";

void setup() {
  Serial.begin(115200);
  if (!bmp.begin()) {
    Serial.println(F("Could not find a valid BMP180 sensor, check wiring!"));
  }
  Serial.println(F("\n"));
  const char* ssid     = "WIFI_SSID";
  const char* password = "WIFI_PASSWORD";
  const char* host = "api.thingspeak.com";

  pinMode(A0, INPUT);
  raw = analogRead(A0);
  volt=raw/1023.0;
  volt=volt*4.2;

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
     delay(500);
  }
  Serial.println(F("Connected to WiFi"));
  String v=String(volt);// change float into string
  // make TCP connections
  WiFiClient client;
  if (client.connect(host,80))  // "184.106.153.149" or api.thingspeak.com
  {
      String postStr = apiKey;
      postStr +="&field1=";
      postStr += String(v);
      postStr +="&field2=";
      postStr += String(raw);
      postStr +="&field3=";
      postStr += String(bmp.readPressure()/100);
      postStr +="&field4=";
      postStr += String(bmp.readTemperature());
      postStr +="&field5=";
      postStr += String(bmp.readAltitude());
      postStr += "\r\n\r\n";

      client.print("POST /update HTTP/1.1\n");
      client.print("Host: api.thingspeak.com\n");
      client.print("Connection: close\n");
      client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
      client.print("Content-Type: application/x-www-form-urlencoded\n");
      client.print("Content-Length: ");
      client.print(postStr.length());
      client.print("\n\n");
      client.print(postStr);
  }
  client.stop();
  Serial.println(F("Sleep......"));
  ESP.deepSleep( 10 * 60 * 1000000 ); //10 minutes
}

void loop() {

}
```

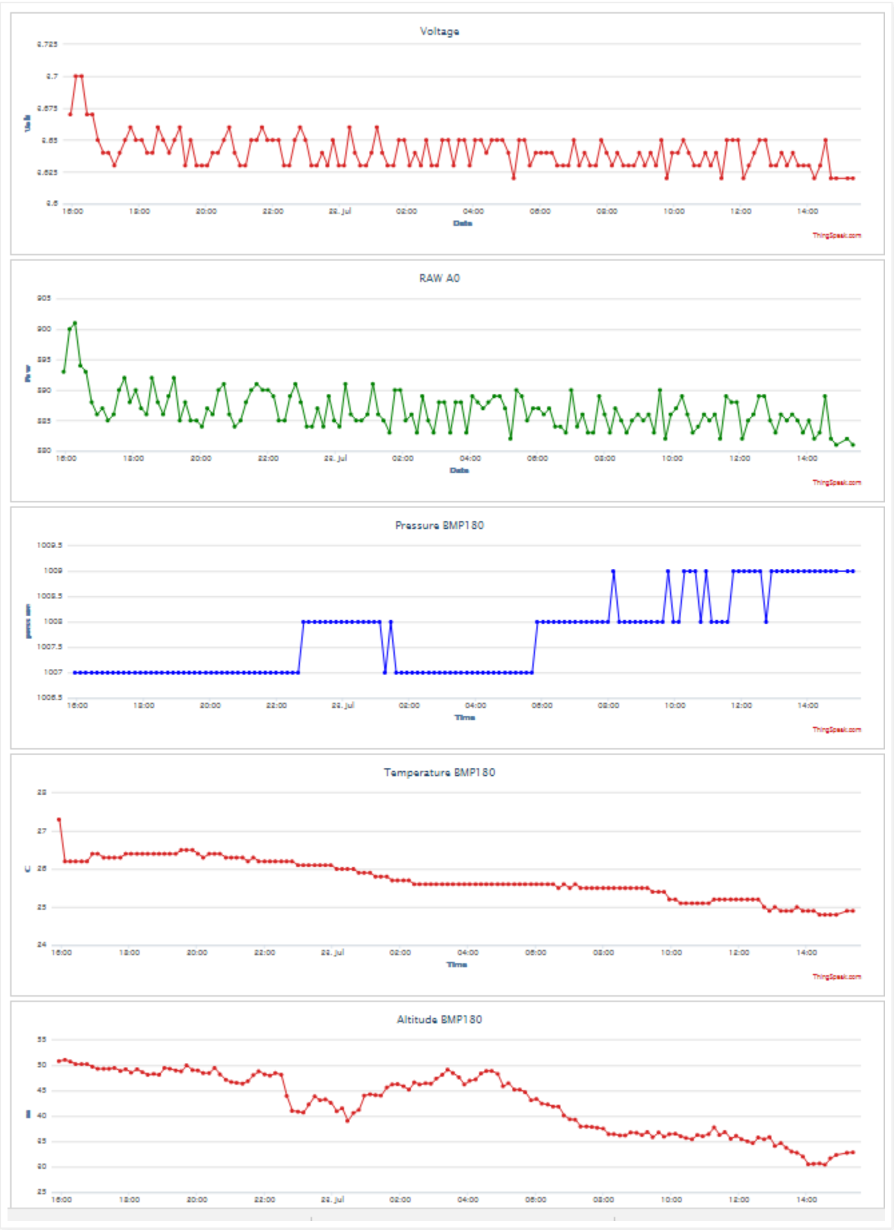
For connecting the BMP180 to Wemos D1 mini see this post.

The Wemos will enter in sleep mode and every 10 minutes will wake and log the data to cloud.

Don't forget to connect the RST pin to D0 to activate the deep sleep.

The 5V to 3V3 on the Wemos is always powers so there is some energy lost on it but is ok.

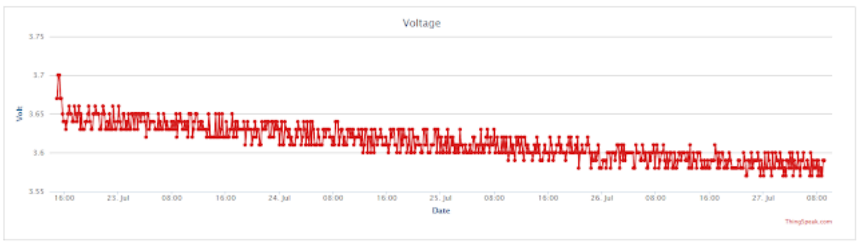
After one day the graph is looking like this:



Total cost: EUR 14.

[EDIT 1] 5 days later after 700 measurements the graph is:

I am considering adding a solar panel to charge the battery during the day.

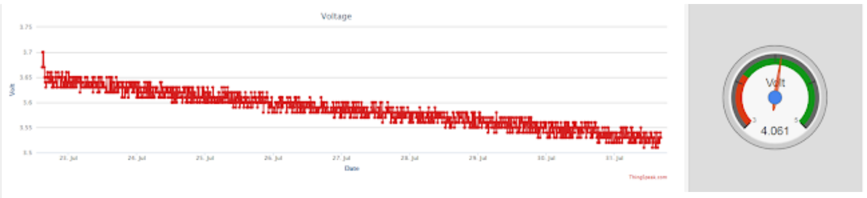


5 days later

[EDIT 2] I've measured battery's voltage today and the deviation is 0.5V. So I need to add 0.5V over the calculated value. Voltage measured on battery after 5 days is 4.11 V.

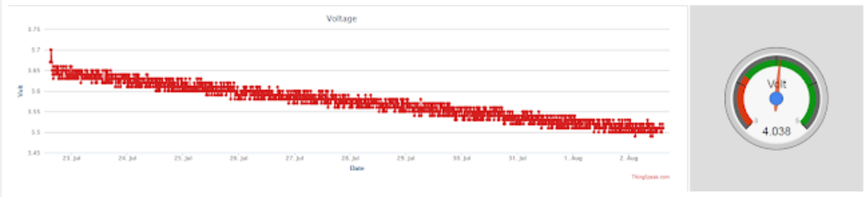
I don't find the any information if the shield module will stop providing voltage to Wemos D1 if the battery voltage drop under the 3.6 V or will dry the 18650 battery, but my guess is that will stop at 3.6 V.

[EDIT 3] After 9 days the battery has 4.06 V



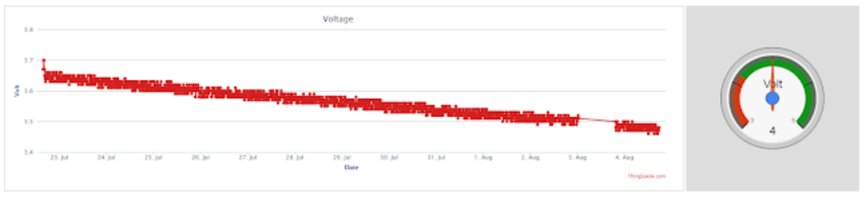
9 days later

[EDIT 4] After 11 days the battery has 4.03 V



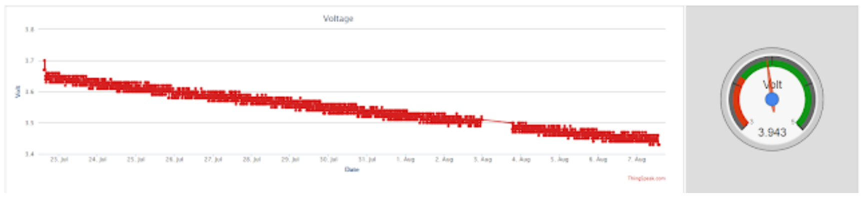
11 days

[EDIT 5] After 13 days the battery has 4.00 V. For one day the router decided to take a break. It looks like the battery still have 0.4 V left and it is enough for the rest of the month. I predict that the module can run 30 days without problems with only one charge. Discharging rate is is pretty linear and I hope it will stay linear.

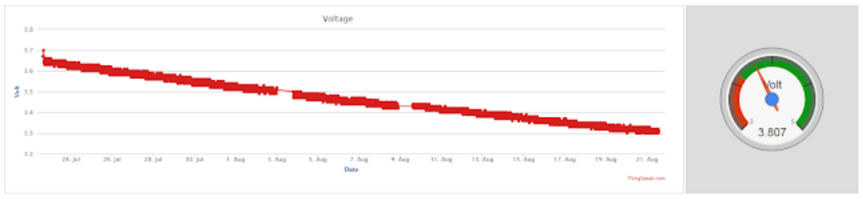


13 days later

[EDIT 6] After 16 days - 3.943 V



[EDIT 7] After 30 days - 3.8 V

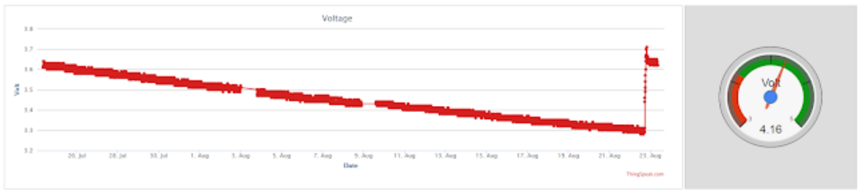


The second interruption was deliberate (I had to travel).

Conclusion:

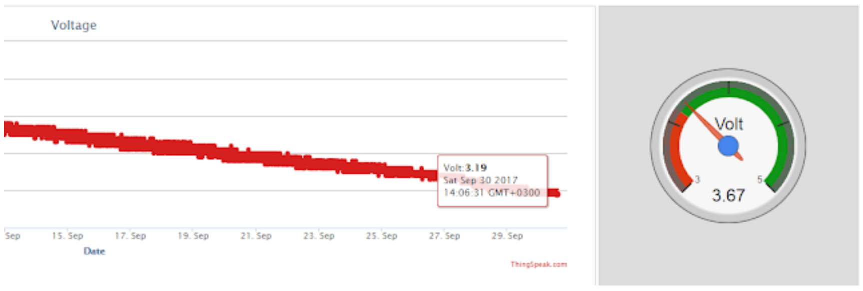
After 4160 records logged with a single charge of one 18650 battery there is still some power left for at least one week. In conclusion you can use the WeMos D1 , Wemos Battery Shield and one 18650 to get and log data at 10 minutes interval with a single charge for more then one month.

[EDIT 8] After 2 hours charge the module is ready for another month of working. I've added a trigger to be notified via twitter when the voltage is bellow 3.7 V to charge it again.



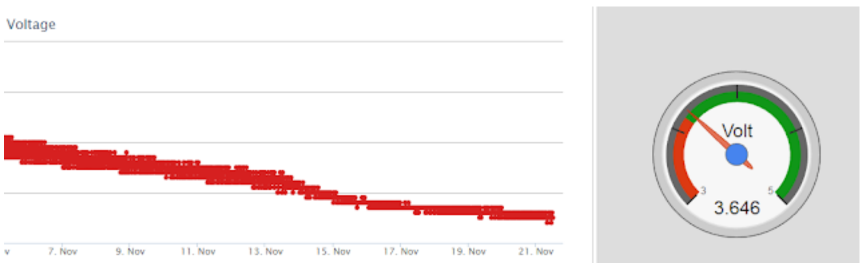
Ready for a new month

[EDIT 9] I have now 38 days with one charge. Remember, every 10 minutes is measuring the temperature and post it on thingspeak.com then is entering again is deep sleep



38 days with one charge

[Edit 10] I have now 48 days with one charge. In the last days I've notice that the value for the battery voltage is quite flat comparing it with the previous values. I am sure that today I will receive a tweet that the voltage have dropped under 3.65V.



48 days with one charge

>>> For the version with a solar panel attached see this post <<<

at 3:43 PM

Reactions: interesting (1) cool (1)

4 comments: Links to this post G+

Labels: Battery powered ESP8266, battery shield ESP8266, Battery wemos, BMP180, deep sleep, measure temperature with ESP8266, MQTT broker

Saturday, July 22, 2017

BMP180 and wemos D1 reading temperature, pressure and altitude

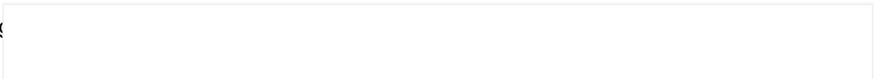
To read the pressure, temperature and altitude you can use an BMP180 cheap i2c breakout.

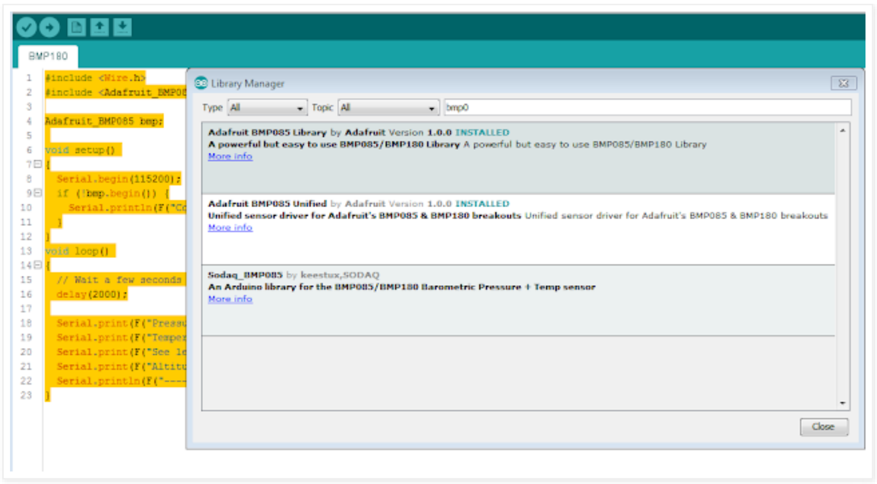
1. Materials:

Wemos D1	USD 3.99
BMP180	USD 1.97
Total:	USD 5.96

2. Libraries:

Install Adafruit BMP_085 libraries from Library Manager:





Adafruit BMP085 Library by **Adafruit** Version **1.0.0** INSTALLED
A powerful but easy to use BMP085/BMP180 Library A powerful but easy to use BMP085/BMP180 Library
More info

Adafruit BMP085 Unified by Adafruit Version 1.0.0 INSTALLED
Unified sensor driver for Adafruit's BMP085 & BMP180 breakouts Unified sensor driver for Adafruit's BMP085 & BMP180 breakouts
More info

Sodaq_BMP085 by keestux,SODAQ
An Arduino library for the BMP085/BMP180 Barometric Pressure + Temp sensor
More info

3. Connections:

VCC ----3V3
GND ----GND
SDA ----D2
SCL ----D1

4. Code:

```
#include <Wire.h>
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;

void setup()
{
  Serial.begin(115200);
  if (!bmp.begin()) {
    Serial.println(F("Could not find a valid BMP180 sensor, check wiring!"));
  }
}
void loop()
{
  // Wait a few seconds between measurements.
  delay(2000);

  Serial.print(F("Pressure: "));  Serial.println(bmp.readPressure()/100);
  Serial.print(F("Temperature: "));  Serial.print(bmp.readTemperature()); Serial.println(F("*C"));
  Serial.print(F("See level pressure: "));  Serial.println(bmp.readSealevelPressure());
  Serial.print(F("Altitude: "));  Serial.println(bmp.readAltitude());
  Serial.println(F("----------------------------------------"));
}
```

at 4:40 PM
Reactions: interesting (0) cool (0)

No comments: Links to this post G+
Labels: banggood.com, BMP180 code, ESP826 temperature, Wemos

Friday, March 10, 2017

Magical Black Bean - Infrared control over WiFi/3G/4G

Few weeks ago I've received the Broadlink RM mini 3 Black Bean Smart Home Wifi Universal IR and I've took some time to test it before I can give more details.

Is absolutely fantastic !

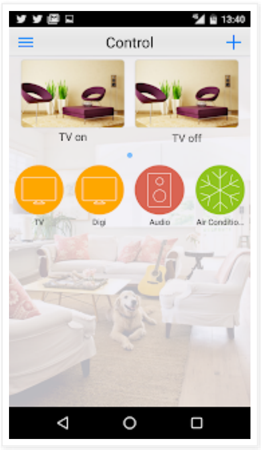
As features:

- -I was able to record any remote I had around
- -I was able to connect to it from WiFi or 4G from my mobile
- -I was able to make scenes for multiple remotes.

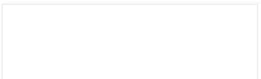
The scenes part is very handy since allows me to control multiple devices with one touch. For example:
-start TV
-wait 1 sec
-start Set-box
-wait 1 sec
-put TV on HDMI 3 input
-change SetBox to channel 5.
-start amplifier

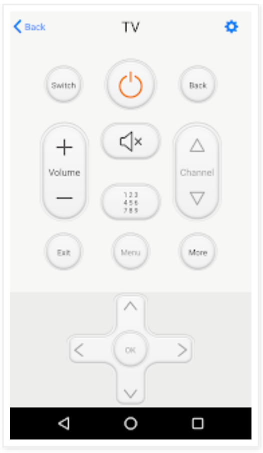
All of that with just one touch from the mobile app.

Learning procedure from the original remotes is very simple, just select the TV to emulate and set which button from virtual remote will be assigned to physical remote.

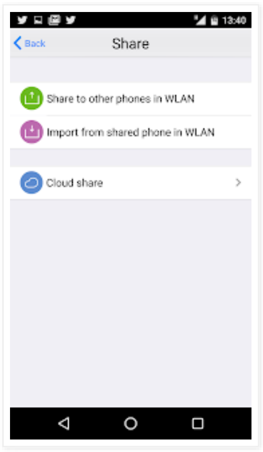


And for example the TV virtual remote is:

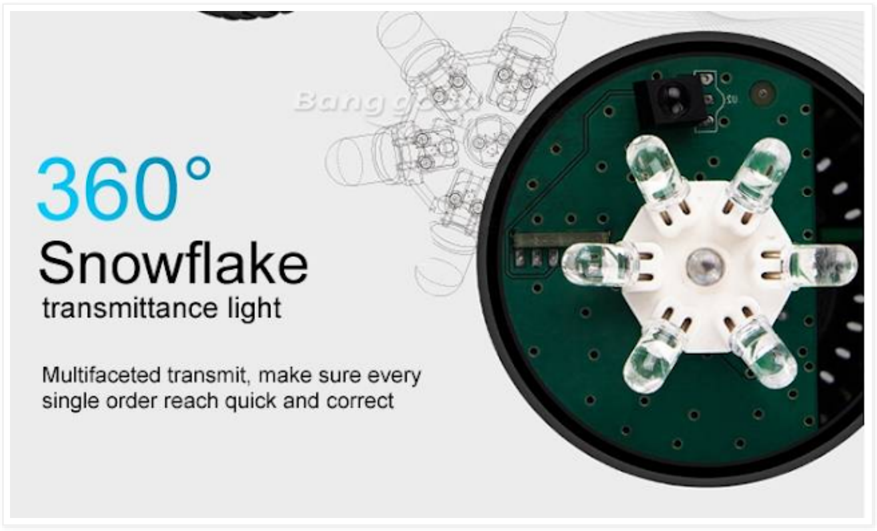




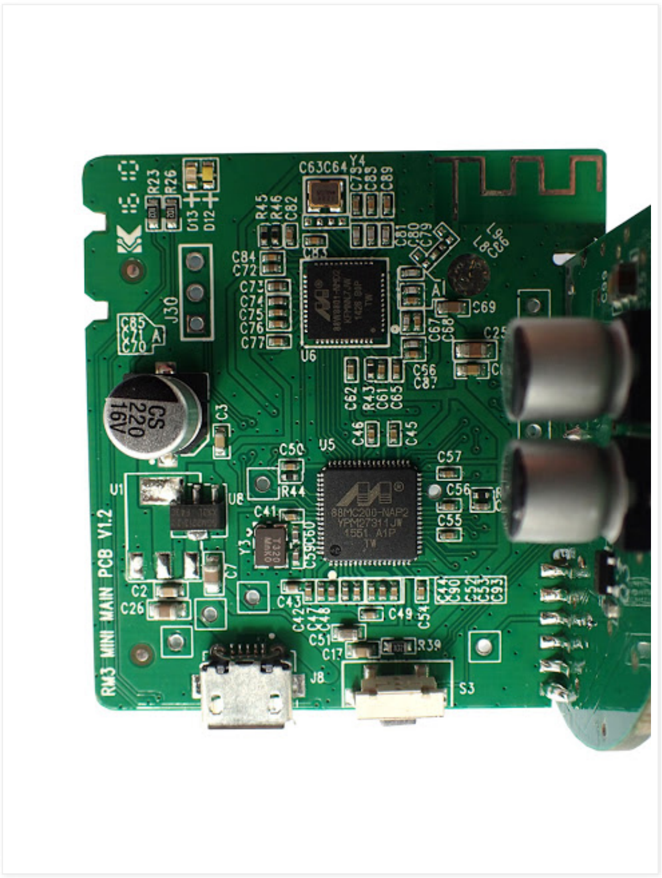
Other important feature was to move all defined remotes to another phone. For this there is a special menu and on one phone we will need to select the "Share to other phones in WLAN" options and on the receiver phone "Import from shared phone in WLAN". In one second I had all my remotes, scripts in the second phone, so no more relearning process. Super !!!.


13:40
Back
Share
Share to other phones in WLAN
Import from shared phone in WLAN
Cloud share

As for what is inside:
-main processor is a 200MHz Cortex M3 Marvel 88MC200 and for WiFi a 88W8801 chip.
-6 snowflake positioned IR emitter LEDs.
-one IR Receiver for learning the codes from your remotes.


360°
Snowflake
transmittance light
Multifaceted transmit, make sure every single order reach quick and correct
Banggood

The PCB is looking like:



Make sure that:
- there is direct line of sight between the mini R and your appliances, otherwise the IR reflection will not always work.
-you have a power adapter to deliver 5V since you will not find one in the box. An old phone charger will be just fine.

at 2:46 PM
Reactions: interesting (0) cool (0)

No comments: Links to this post G+
Labels: ESP8266 air conditioning, IR receiver/transmitter for WiFi.

Newer Posts Home Older Posts

Subscribe to: Posts (Atom)