esp8266, batería, pila, energía, ahorrar

Ahorrar Baterías

Para reducir el consumo de nuestro esp8266 tenemos dos aproximaciones que lo hacen posible. La primera sería modificando el código de programación para que nuestro esp8266 entre en reposo y la segunda sería usando un micro controlador AtTiny13A para que active o desactive nuestro esp8266.

Características del deep-sleep	
WiFI	OFF
System Clock	OFF
RTC	ON
CPU	OFF
Consumo	~0.02mA

El tiempo de reposo se especifica en microsegundos (µs).

El tiempo máximo que se puede especificar según las especificaciones del ESP8266 es de 4,294,967,295 µs, ~71 minutos.

Para habilitar el modo Deep-sleep, necesitamos conectar un cable entre el pin RST y el al pin GPIO 16 (D0 en el ESP8266).

Mediante programación

Lo que hacemos es añadir a nuestro código una variable con el tiempo de reposo y llamamos a la función **ESP.deepSleep**

```
const int sleepTimeS = 10;

Dentro de setup()  añadimos la llamada

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

Un ejemplo completo sería

```
// Library
#include <ESP8266WiFi.h>

// WiFi settings
```

```
const char* ssid = "wifi-name";
const char* password = "wifi-password";

// Time to sleep (in seconds):
const int sleepTimeS = 10;

// Host
const char* host = "dweet.io";

void setup()
{

  // Serial
  Serial.begin(115200);
  Serial.println("ESP8266 in normal mode");
  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  // Print the IP address
  Serial.println(WiFi.localIP());

  // Logging data to cloud
  Serial.print("Connecting to ");
  Serial.println(host);
  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
  // This will send the request to the server
  client.print(String("GET /dweet/for/myesp8266?message=lowpower") + "
HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(10);
  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  Serial.println();
  Serial.println("closing connection");

  // Sleep
```

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

void loop()
{

}
```

Usando el AtTiny13A

Se consigue reducir el consumo a .005mA durante el reposo. El circuito y el código fuente se encuentra en http://homecircuits.eu/blog/battery-powered-esp8266-iot-logger/

Para programa el microcontrolador http://homecircuits.eu/blog/program-attiny13a-via-arduino-board/

Referencias

- https://github.com/openhomeautomation/esp8266-battery
- http://homecircuits.eu/blog/low-power-picopower-attiny13a/
- https://openhomeautomation.net/esp8266-battery/
- http://www.esp8266.com/wiki/doku.php?id=esp8266_power_usage
- https://www.losant.com/blog/making-the-esp8266-low-powered-with-deep-sleep