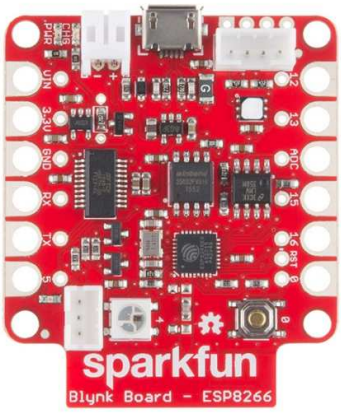
Blynk IoT



Sparkfun Blynk ESP8266 (1) este o placă de dezvoltare care pe lângă avantajele oferite de serviciul cloud Blynk (prezentate în lecția „Cum să realizăm un sistem IoT fără să scriem nici o linie de cod?”) oferă o combinație de componente extrem de interesante. Procesorul WiFi ESP8266, senzorul digital de temperatură și umiditate Si7021 integrat și posibilitatea de alimentare de la un acumulator fac din placa Sparkfun Blynk ESP8266 o excelentă platformă pentru experimente IoT. În cadrul lecției de față vom programa această placă utilizând Arduino IDE și nu vom utiliza platforma cloud specifică ci noul serviciu Robofun IoT (2):

iot.robofun.ro

Pentru punerea în funcțiune și programarea plăcii Blynk se recomandă parcurgerea materialului „Blynk Board Arduino Development Guide” (3). Pentru prima parte a lecției nu avem nevoie decât de placa de dezvoltare și cablul USB de încărcare a programului – senzorul de temperatură și umiditate ce va furniza informațiile transmise către serviciul IoT este integrat pe placă.

Programul a fost dezvoltat și testat utilizând Arduino IDE 1.8.1, extensia esp8266 2.1.0 și bibliotecile Sparkfun Si7021 Humidity and Temperature Sensor 1.0.0 și Adafruit NeoPixel 1.1.1 (pe placa de dezvoltare se află și un led RGB WS2812 pe care îl vom comanda utilizând această bibliotecă). Declarațiile inițiale ale programului sunt următoarele:

```
#include <Wire.h>
#include "SparkFun_Si7021_Breakout_Library.h"
#include <Adafruit_NeoPixel.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
```

În cadrul programului trebuie personalizate datele de conectare WiFi:

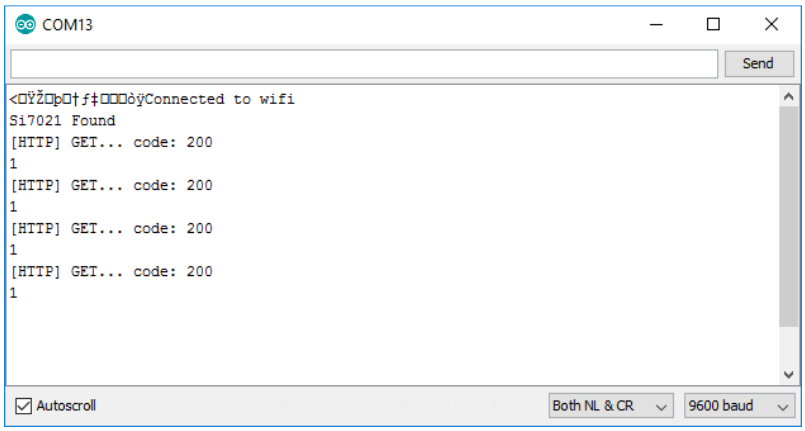
```
char ssid[] = "…";
char pass[] = "…";

WiFiClient client;
float humidity = 0;
float tempC = 0;
Weather sensor;
#define PIN  4
#define NUMPIXELS 1
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS,
                           PIN, NEO_GRB + NEO_KHZ800);
```

Varialbila *postingInterval* indică intervalul de timp între două raportări IoT (în milisecunde, în cazul dat ca exemplu: 10 minute)

```
unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 600L * 1000L;
```

Dacă doriți ca programul să afișeze informații despre execuție se va decomenta următoarea linie:

```
//#define debug
```



https://www.robofun.ro/forum/

În cadrul secțiunii *setup()* se va inițializa conexiunea WiFi – pe durata inițializării led-ul RGB de pe placă va lumina roșu.

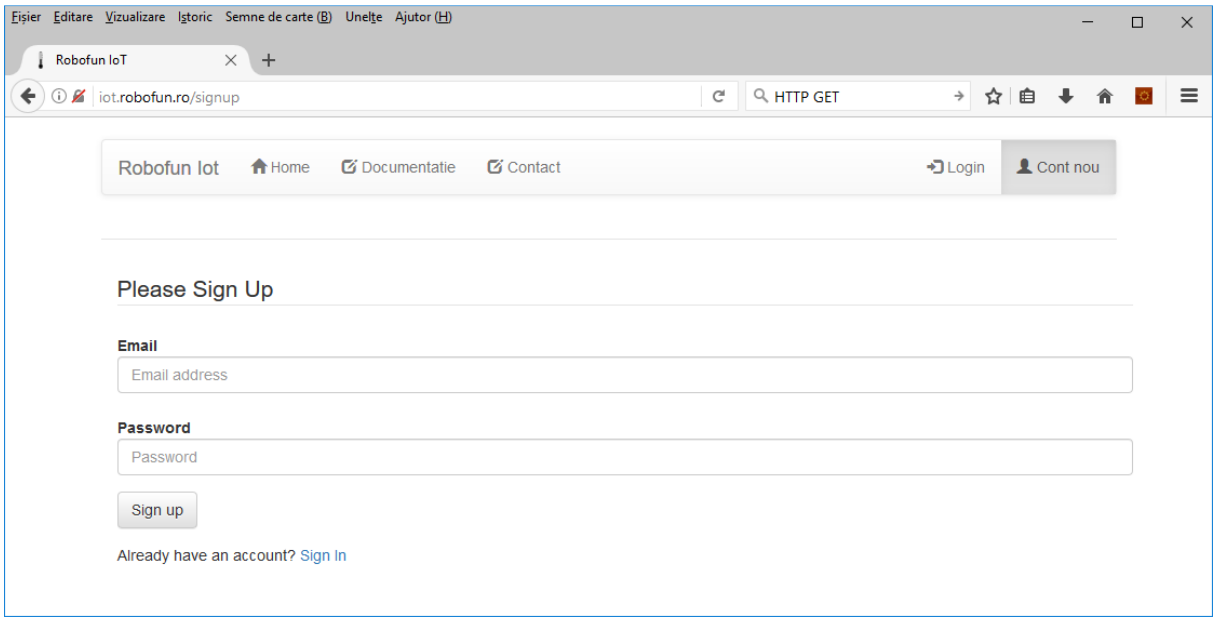
```
void setup() {
  pixels.begin();
  pixels.setPixelColor(0, pixels.Color(150,0,0));
  pixels.show();
  #ifdef debug
    Serial.begin(9600);
  #endif
  int status = WL_IDLE_STATUS;
  while (status != WL_CONNECTED)
  {
    status = WiFi.begin(ssid, pass);
    delay(10000);
  }
  #ifdef debug
    Serial.println("Connected to wifi");
  #endif
  sensor.begin();
  pixels.setPixelColor(0, pixels.Color(0,0,0));
  pixels.show();
}
```

Secțiunea *loop()* va verifica timpul scurs de la ultima raportare și dacă este depășit intervalul specificat se va apela procedura *IoTPublish()* care se ocupă de citirea senzorului Si7021 și de trimiterea datelor către Robofun IoT. Pe durata execuției procedurii led-ul de pe placă va lumina verde.

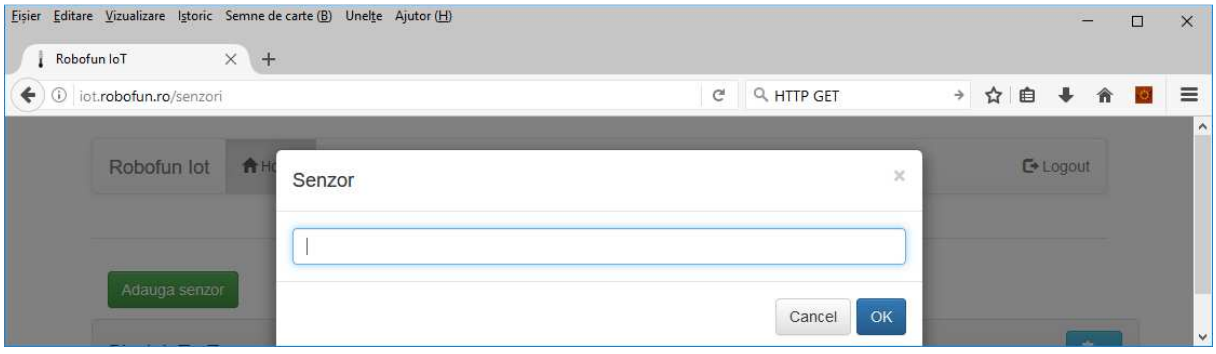
```
void loop() {
  if (millis() - lastConnectionTime > postingInterval)
  {
    pixels.setPixelColor(0, pixels.Color(0,150,0));
```

```
    pixels.show();
    IoTpublish();
    pixels.setPixelColor(0, pixels.Color(0,0,0));
    pixels.show();
  }
}
```

Procedura *IoTpublish()* va posta datele oferite de senzorul de temperatură și umiditate către serviciul IoT. Postarea se face prin două apeluri HTTP GET (4), câte unul pentru fiecare parametru trimis (temperatură și umiditate). Apelurile HTTP vor returna răspunsul *1* dacă totul este în regulă sau alte valoare dacă a apărut o eroare. Pentru utilizarea serviciului Robofun IoT este necesară înregistrarea gratuită a unui cont.



După înregistrare și conectare este necesară definirea a doi noi senzori.



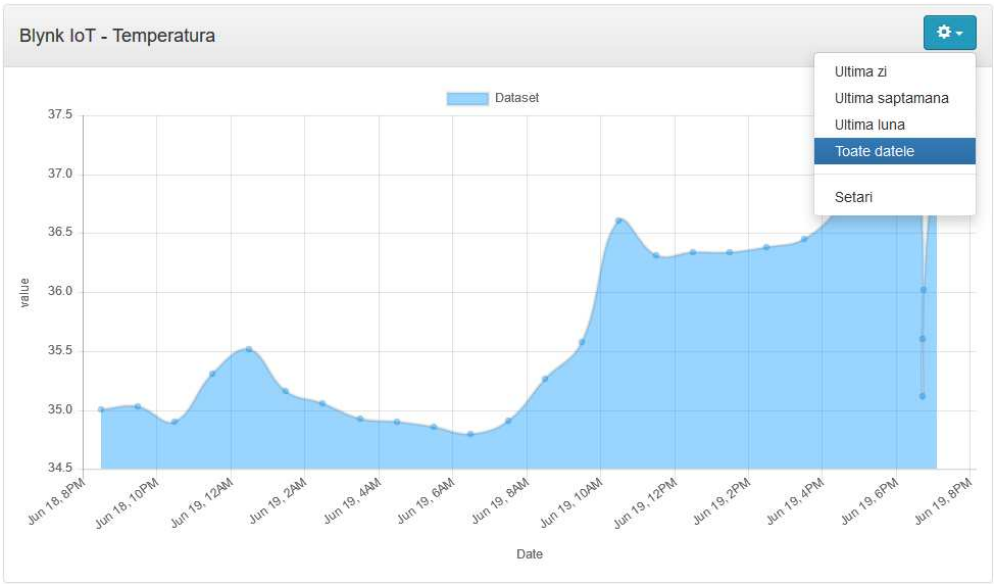
https://www.robofun.ro/forum/

Fiecare senzor va avea propria cheie de autentificare (token) și pentru fiecare dintre cei doi senzori este necesar să copiem această cheie în program (*SENZOR_TOKEN1* și *SENZOR_TOKEN2*).

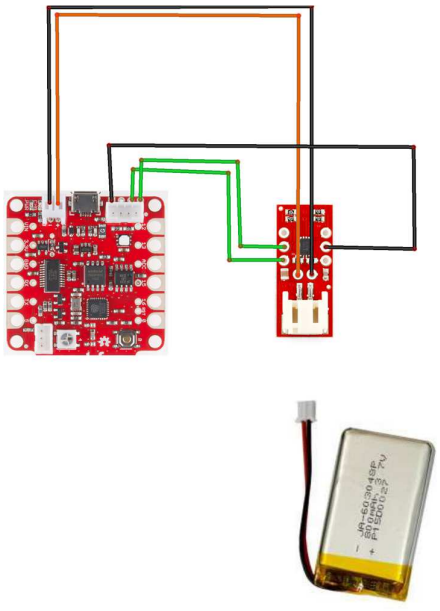
```
void IoTpublish() {
  String SENSOR_TOKEN1="…";
  String SENSOR_TOKEN2="…";
  HTTPClient http;
  humidity = sensor.getRH();
  tempC = sensor.getTemp();
  String data =
    String("http://iot.robofun.ro/api/v1/input/") +
    SENSOR_TOKEN1 + "/" + String(tempC, DEC);
  http.begin(data);
  int httpCode = http.GET();
  #ifdef debug
    if(httpCode > 0) {
      Serial.printf("[HTTP] GET... code: %d\n", httpCode);
      if(httpCode == HTTP_CODE_OK) {
        String payload = http.getString();
        Serial.println(payload);
      }
    } else {
      Serial.printf("[HTTP] GET... failed, error: %s\n",
                http.errorToString(httpCode).c_str());
    }
  #endif
  http.end();
  data = String("http://iot.robofun.ro/api/v1/input/") +
    SENSOR_TOKEN2 + "/" + String(humidity, DEC);
  http.begin(data);
  httpCode = http.GET();
  #ifdef debug
```

```
    if(httpCode > 0) {
      Serial.printf("[HTTP] GET... code: %d\n", httpCode);
      if(httpCode == HTTP_CODE_OK) {
        String payload = http.getString();
        Serial.println(payload);
      }
    } else {
      Serial.printf("[HTTP] GET... failed, error: %s\n",
            http.errorToString(httpCode).c_str());
    }
  #endif
  http.end();
  lastConnectionTime = millis();
}
```

După încărcarea programului pe placa de dezvoltare și scurgerea unui interval de funcționare mai lung decât *postingInterval* vom putea vedea deja pe graficul oferit de interfața serviciului valorile achiziționate de senzorul plăcii. În capturile de ecran următoare putem vedea un grafic pe o perioadă mai lungă de timp (temperatura – vizualizare a tuturor valorilor, umiditate – vizualizare a valorilor medii la nivel de săptămână).



Un atu foarte important al plăcii de dezvoltare Sparkfun Blynk ESP8266 este posibilitatea de alimentare utilizând un acumulator LiPo de 3.7V (5). Asta conferă sistemului portabilitate – vom dispune de un sistem de achiziție IoT de dimensiunea a două cutii de chibrituri. Totuși, o problemă majoră care se naște în acest caz este evaluarea autonomiei de funcționare a sistemului – determinarea nivelului de încărcare a acumulatorului. Utilizarea convertorului analog-numeric al plăcii în acest scop presupune construirea unui divizor de tensiune (convertorul poate măsura tensiuni între 0V și 3.3V) dar această soluție conduce la descărcarea mai rapidă a acumulatorului. O soluție mult mai elegantă pentru rezolvarea acestei probleme este utilizarea unei componente specializate de măsurare: LiPo Fuel Gauge (6). Conectarea acestei componente, între conectorul de alimentare al plăcii și acumulator, este prezentată în diagrama următoare:



https://www.robofun.ro/forum/

Circuitul integrat MAX17043G+U (7) al componentei de măsurare va supraveghea nivelul de tensiune al acumulatorului și îl va raporta către placa de dezvoltarea prin intermediul magistralei I2C – placa de dezvoltare are un conector special I2C la care se poate conecta componenta de măsurarea. Introducerea componentei de măsurare nu va influența în nici un fel posibilitatea de încărcarea a acumulatorului prin intermediul mufei microUSB de pe placă și nici nu va scădea durata de viață a acumulatorului.

În cadrul programului vom adăuga următoarele funcții și proceduri necesare lucrului cu circuitul MAX17043:

```
unsigned int vcellMAX17043() {
  unsigned int vcell;
  vcell = i2cRead16(0x02);
  vcell = vcell >> 4;
  return vcell;
}

float percentMAX17043() {
  unsigned int soc;
  float percent;
  soc = i2cRead16(0x04);
  percent = (byte) (soc >> 8);
  percent += ((float)((byte)soc))/256;
  return percent;
}

void qsMAX17043() {
  i2cWrite16(0x4000, 0x06);
}

unsigned int i2cRead16(unsigned char address) {
  int data = 0;
  Wire.beginTransmission(MAX17043_ADDRESS);
  Wire.write(address);
```

```
  Wire.endTransmission();
  Wire.requestFrom(MAX17043_ADDRESS, 2);
  while (Wire.available() < 2)
    ;
  data = ((int) Wire.read()) << 8;
  data |= Wire.read();
  return data;
}

void i2cWrite16(unsigned int data, unsigned char address)
{
  Wire.beginTransmission(MAX17043_ADDRESS);
  Wire.write(address);
  Wire.write((byte)((data >> 8) & 0x00FF));
  Wire.write((byte)(data & 0x00FF));
  Wire.endTransmission();
}
```

La începutul programului vom defini adresa I2C a circuitului MAX17043:

```
#define MAX17043_ADDRESS 0x36
```

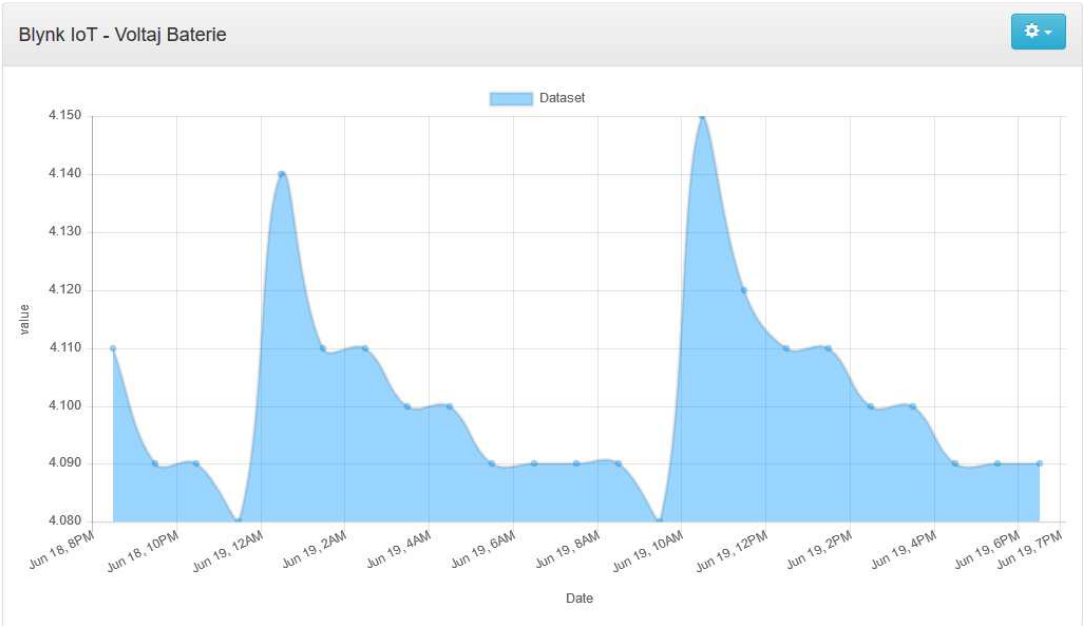
Iar în cadrul procedurii *IoTpublish()* vom adăuga:

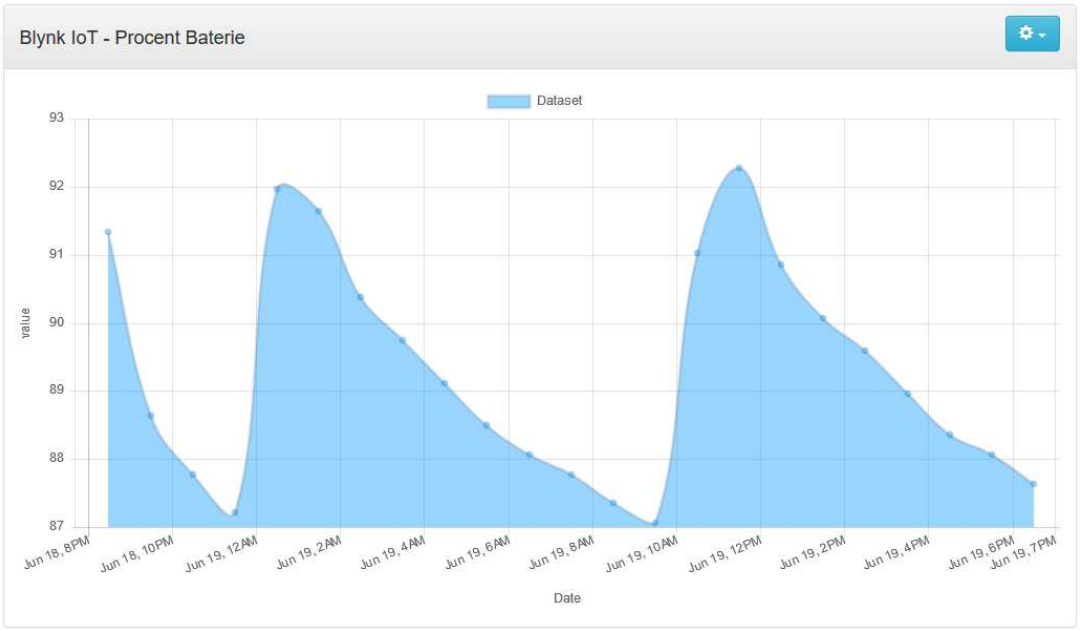
```
float batVoltage;
float batPercentage;
…
String SENSOR_TOKEN3="…";
String SENSOR_TOKEN4="…";
….
qsMAX17043();
batPercentage = percentMAX17043();
```

```
 batVoltage = (float) vcellMAX17043() * 1/800;
…
data = String("http://iot.robofun.ro/api/v1/input/") +
    SENSOR_TOKEN3 + "/" + String(batVoltage, DEC);
  http.begin(data);
  httpCode = http.GET();
  #ifdef debug
    if(httpCode > 0) {
      Serial.printf("[HTTP] GET... code: %d\n", httpCode);
      if(httpCode == HTTP_CODE_OK) {
        String payload = http.getString();
        Serial.println(payload);
      }
    } else {
      Serial.printf("[HTTP] GET... failed, error: %s\n",
            http.errorToString(httpCode).c_str());
    }
  #endif
  http.end();
  data = String("http://iot.robofun.ro/api/v1/input/") +
        SENSOR_TOKEN4 + "/" + String(batPercentage, DEC);
  http.begin(data);
  httpCode = http.GET();
  #ifdef debug
    if(httpCode > 0) {
      Serial.printf("[HTTP] GET... code: %d\n", httpCode);
      if(httpCode == HTTP_CODE_OK) {
        String payload = http.getString();
        Serial.println(payload);
      }
    } else {
      Serial.printf("[HTTP] GET... failed, error: %s\n",
```

```
        http.errorToString(httpCode).c_str());    }
#endif
http.end();
```

Înainte de încărcarea noii versiuni de program este necesar să definim doi noi senzori (unul pentru tensiunea acumulatorului altul pentru procentul de încărcare) și să copiem cheile de autentificare în variabilele asociate: *SENZOR_TOKEN3* și *SENZOR_TOKEN4*. După rularea programului vom putea vedea evoluția parametrilor acumulatorului folosit:





Referințe on-line

(1) Blynk Board - ESP8266 Wifi

https://www.robofun.ro/iot/blynk-board-esp8266?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(2) Robofun IoT

http://iot.robofun.ro/

(3) Blynk Board Arduino Development Guide

https://learn.sparkfun.com/tutorials/blynk-board-arduino-development-guide

(4) HTTP Methods: GET vs. POST

https://www.w3schools.com/tags/ref_httpmethods.asp

(5) Acumulatori

https://www.robofun.ro/surse_de_alimentare/acumulatori?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(6) LiPo Fuel Gauge

https://www.robofun.ro/surse_de_alimentare/acumulatori/lipo_gauge_fuel?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(7) 1-Cell/2-Cell Fuel Gauge with ModelGauge and Low-Battery Alert

https://datasheets.maximintegrated.com/en/ds/MAX17043-MAX17044.pdf