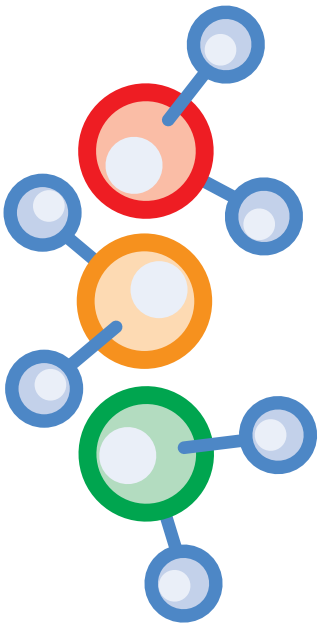


Sigfox CO_2-stoplicht

zonder WiFi-netwerk!

Peter Groppe, Frank Schleking en Bernd vom Berg (Duitsland)

We hebben in Elektor al een aantal CO_2-meters besproken. Bijna allemaal hebben ze een WiFi-interface, zodat je de meetwaarden overal ter wereld kunt bekijken. Dit CO_2-stoplicht is anders; het maakt verbinding met het IoT via het Sigfox-netwerk. Hierdoor heeft de sensor een aanzienlijk groter bereik en kan hij ook worden ingezet op locaties waar geen WiFi-netwerk voorhanden is.

Verschillende studies hebben aangetoond dat de luchtkwaliteit afneemt in niet-geventileerde openbare ruimten; een hoge CO_2-concentratie correleert met een hogere virusbelasting van de lucht die we inademen. Het is belangrijk om een goede luchtstroom te handhaven om het risico van het doorgeven van door de lucht verspreide virussen te helpen verminderen. CO_2-sensormodules zijn overal verkrijgbaar en kunnen met een controller worden gebruikt om de lucht die we inademen continu te controleren. Waarschuwingen en alarmmeldingen kunnen worden getriggerd wanneer de gemeten concentratie een vooraf ingestelde drempelwaarde overschrijdt.

Dat is de taak van een standaard CO_2-'stoplicht'-systeem. Het geeft een visueel en soms ook akoestisch alarm wanneer meer frisse lucht nodig is. De concentratie van CO_2 wordt aangegeven op een LED-display (rood, geel of groen, vandaar de naam 'stoplicht'). In het IoT-tijdperk hebben de meeste van deze systemen ook een WiFi-interface, zodat de meetgegevens naar een cloud-platform kunnen worden overgebracht en op een overal ter wereld toeganklijke webpagina kunnen worden weergegeven.

Het hier beschreven CO_2-stoplicht maakt verbinding met het internet via het Sigfox-radionetwerk en niet via WiFi. Sigfox is bijzonder geschikt voor deze toepassing omdat we slechts kleine hoeveelheden gegevens hoeven te verzenden en de uitstekende radiodekking van Sigfox ons maximale flexibiliteit biedt bij het opstellen van de sensor. Het Sigfox-netwerk heeft doorgaans slechts een handvol basisstations nodig om een hele stad te dekken, en de dekking in veel landen over de hele wereld is al zeer goed. Dit maakt het CO_2-stoplichtsysteem ideaal voor gebruik op plaatsen waar we geen WiFi-toegang hebben.

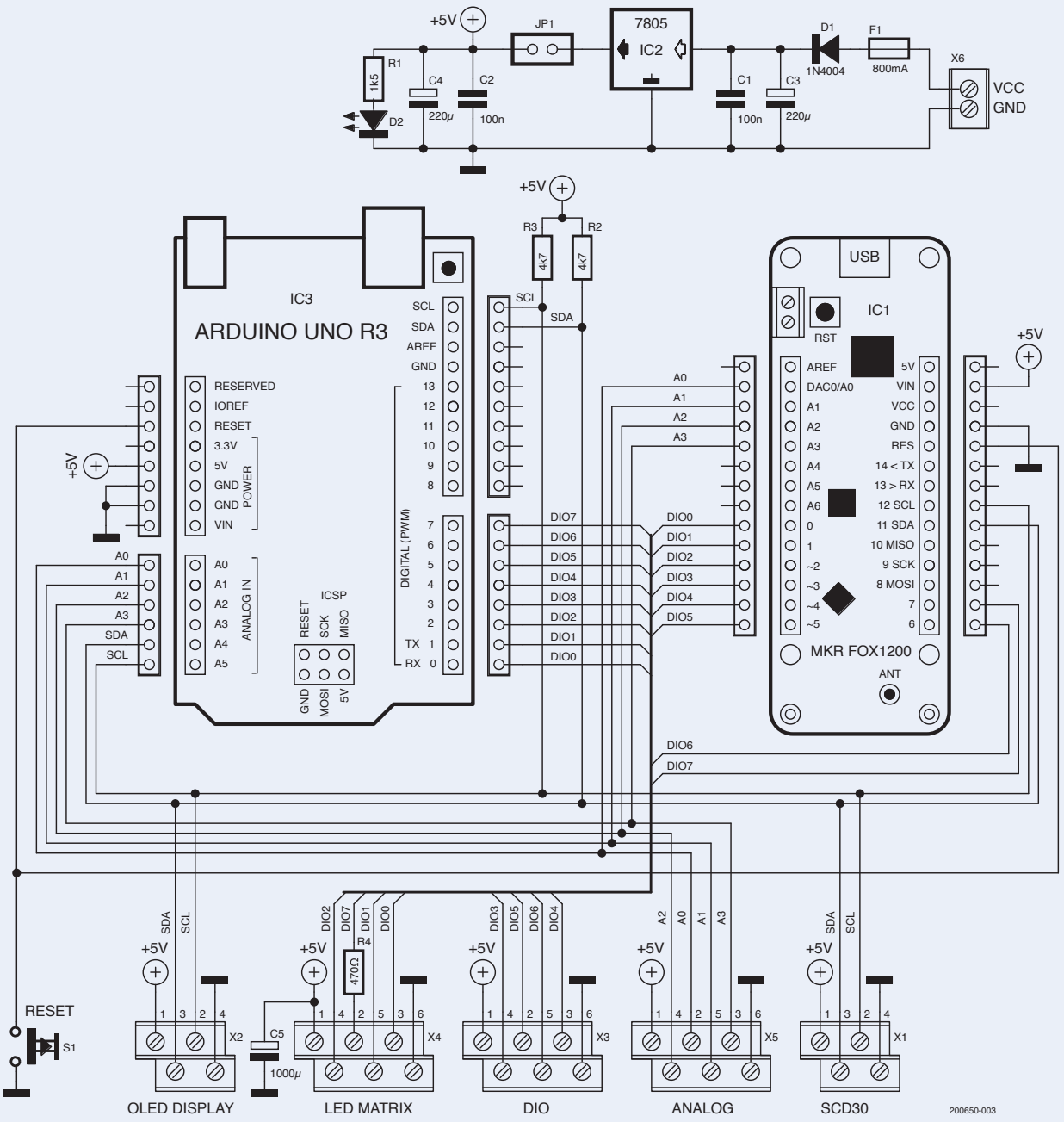
De hardware

De hardware voor het CO_2-stopslicht is vrij eenvoudig; tegenwoordig is er een breed scala aan uiterst krachtige kant-en-klare Arduino-microcontrollerboards verkrijgbaar die in dit project kunnen worden gebruikt. Een zo'n board (dat geschikt is voor Sigfox-communicatie) is de Arduino MKR FOX1200, die we eerder hebben geïntroduceerd in de Elektor-artikelreeks "Sigfox en het IoT" [6].

Als je geen IoT-connectiviteit nodig hebt en de CO_2-metingen alleen lokaal wilt weergeven, kun je ook een gewone Arduino Uno gebruiken die ook op het hier gepresenteerde moederbord-ontwerp kan worden aangesloten. **Figuur 1** geeft het schema, waarin je ziet hoe ofwel IC3 (een Arduino Uno R3) of IC1 (Arduino MKR FOX1200) op het moederbord wordt aangesloten.

Het microcontrollerboard registreert de door de sensor (een SCD30) gemeten waarden (CO_2-concentratie, luchttemperatuur en -vochtigheid) met regelmatige (instelbare) tussenpozen en toont deze waarden op een 1,3" OLED-display. De stoplicht-indicatie bestaat uit een veelkleurig NeoPixel LED-matrixdisplay.

De CO_2-sensor SCD30 [1] maakt gebruik van een I²C-bus (SDA, SCL, +5V en GND) en wordt aangesloten via de ruimtebesparende tweerijige printkroonsteen X1. Indien



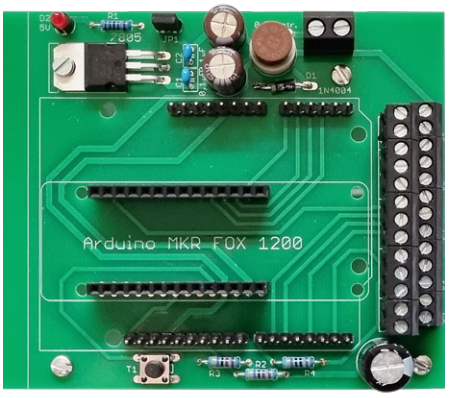
Figuur 1. Het schema van het CO_2-stoplicht met de bedrading van zowel de Arduino Uno als het MKR FOX1200-board.

nodig kunnen 4,7-kΩ pull-up weerstanden worden aangesloten op SDA en op SCL. Het OLED-display [2] wordt eveneens aangesloten via de I²C-bus, en wel op X2.

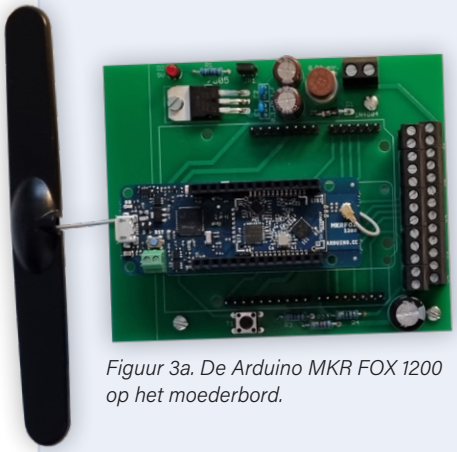
Naast een voedingsaansluiting hebben we slechts een enkele digitale poortpin nodig om het NeoPixel LED-matrixdisplay aan te sturen [4]. We gebruiken hier DIO7 en het display wordt aangesloten op X4. NeoPixel-displays met verschillende aantallen LED's kunnen via deze enkele poortpin worden aangestuurd.

Op de ongebruikte digitale/analoge aansluitingen X3, X4 en X5 kunnen extra sensoren of actuatoren worden aangesloten. Deze aansluitingen kunnen worden gebruikt om het systeem in de toekomst van extra functionaliteit te voorzien, bijvoorbeeld om de airconditioning of een ventilator in te schakelen wanneer de gemeten waarden de drempelwaarden overschrijden.

Figuur 2 toont de compleet gemonteerde print waarop ofwel de Arduino MKR FOX1200-module ofwel het Arduino Uno-printje (met de voorkant naar beneden) in de corresponderende connectoren wordt geprikt. **Op het moederbord mag op elk moment slechts één van beide worden geplaatst! Probeer niet beide printen te monteren met verlengde headers.** De layout van de print, de componentenopstelling, de onderdelenlijst

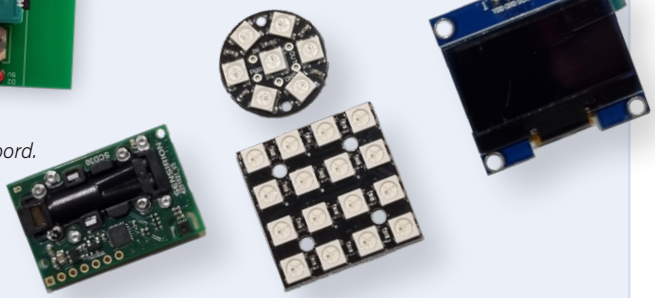


Figuur 2. De print van het moederbord van het CO_2-stoplicht.



Figuur 3a. De Arduino MKR FOX 1200 op het moederbord.

Figuur 3b. De Arduino Uno 'ondersteboven' op het moederbord.



Figuur 3c. De CO_2-sensormodule (links) en verschillende displays.

en de gebruikte Arduino-firmware kunnen worden gedownload van de projectpagina bij dit artikel [3]. De platte Sigfox-antenne kan met dubbelzijdige plakband worden vastgezet. **Figuur 3** toont beide versies in volle glorie.

De voeding

Een **belangrijke overweging** bij het gebruik van NeoPixels is hun stroomverbruik. In het ergste geval, wanneer alle drie individuele LED's van een NeoPixel op maximale helderheid zijn ingesteld (waarde 255) om fel wit licht te produceren, zal elk LED-element 20 mA trekken, zodat één enkele NeoPixel 60 mA consumeert. Een matrix van 16 NeoPixels trekt dan maar liefst 16 × 60 mA = 960 mA! Dat is meer dan kan worden geleverd door de (ongekoelde) 5-V SMD-spanningsregelaar op het Arduino-board. Voeding vanaf het Arduino-board zou de regelaar overbelasten, waardoor de beveiliging ervan kan aanspreken of in het ergste geval de regelaar kan overlijden.

Onder normale omstandigheden zul je de NeoPixels waarschijnlijk niet zo bruut aansturen, maar reken op een gemiddeld stroomverbruik van 25...30 mA per NeoPixel, afhankelijk van de weergegeven kleur. Met een 4×4 LED-matrix zit je nog steeds op 400...480 mA, en dat is nog altijd te veel voor de spanningsregelaar op het Arduino-board.

Daarom moet het volgende in acht worden genomen bij het gebruik van NeoPixel-displays:

- Monteer direct bij de aansluiting van de NeoPixel-voeding een bufferelco om fluctuaties in de voedingsspanning te beperken als de LED's worden geschakeld. Gebruik een elco van 470...1000 µF voor kleinere displays. Hier gebruiken we 1000 µF (C5).
- Monteer een laagohmige weerstand in serie met de besturingslijn van het display om storingen te reduceren. Wij gebruikten een waarde van 470 Ω (R4).
- Er is een externe (geregelde) voeding nodig met een goede spanningsregelaar (eventueel gekoeld) en met voldoend dikke buffercondensatoren (hier gebruiken we een conventionele lineaire 7805-regelaar op het moederbord die gevoed wordt uit een 9V/1A-netadapter). Schakelende DC/DC-converters kunnen hier ook worden gebruikt; deze hebben een beter rendement en zijn niet veel duurder.
- In geen geval mag de ongekoelde SMD-regelaar op het Arduino-board worden gebruikt, aangezien dat snel fout kan gaan ten gevolge van oververhitting of spikes.
- De NeoPixel LED-displaymatrix mag niet groter zijn dan absoluut nodig en mag niet bij maximale helderheid worden gebruikt. De helderheid van het ronde 7-element NeoPixel-display dat we hier gebruiken, wordt door de software aangestuurd met waarden van 4 tot 16 (van de 255 mogelijke niveaus).

Als het systeem wordt gebruikt zonder NeoPixel-display (voor test- of andere doeleinden) en als het totale stroomverbruik niet te hoog is, kan de voeding eenvoudig worden verzorgd door de on-board SMD-spanningsregelaar. In dat geval moet jumper JP1 op het moederbord worden verwijderd, zodat de 7805 niet 'omgekeerd' gevoed wordt.

Extra bibliotheken voor de Arduino IDE

De CO_2-stoplichtsoftware vereist enkele extra bibliotheken om externe componenten zoals de SCD30 CO_2-sensor, het 1,3" OLED-display en de NeoPixel LED-matrix aan te sturen. Er zijn ook twee extra bibliotheken nodig voor de samenwerking van Sigfox met het MKR FOX1200-board en een optionele voor de on-chip RTC van de SAMD21-microcontroller.

De SCD30 CO_2-sensorbibliotheek

Ga, om de SparkFun driver-bibliotheek voor de SCD30-sensor te installeren, naar *Tools -> Manage Libraries...* in de Arduino IDE, en zoek dan naar *scd30* in de rechter bovenhoek. De twee meest populaire SCD30-bibliotheken zullen worden vermeld, een van Adafruit en een van SparkFun. Zoals uit **figuur 4** blijkt, gebruiken wij de SparkFun-bibliotheek.

Na installatie wordt de bibliotheek automatisch geïntegreerd in ons programma in de IDE via *Sketch -> Include Library -> ... SparkFun SCD30 Arduino Library*. Je krijgt een gedetailleerde beschrijving van de bibliotheek door op *More info* te klikken (linksonder in het scherm van figuur 4.

De 1,3" OLED display-bibliotheek

De krachtige en complete *U8g2*-bibliotheek van Oli Kraus wordt gebruikt om het 1,3" OLED-display met 128*64 pixels aan te sturen. Dit complete softwarepakket bestaat eigenlijk uit vier afzonderlijke bibliotheken:

- *U8g2:* bibliotheek voor display-toepassingen die gebruik maken van een groot aantal grafische functies en tekensets.
- *U8x8:* bibliotheek voor eenvoudige, puur tekstgebaseerde display-toepassingen

met een beperkte reeks tekensets.
- *MUIU8g2:* speciale functies voor de realisatie van interactieve, grafisch georienteerde Monochrome User Interfaces (MUI).
- *U8log:* functies voor het emuleren van een terminal, vergelijkbaar met de seriële monitor-functie van de Arduino IDE.

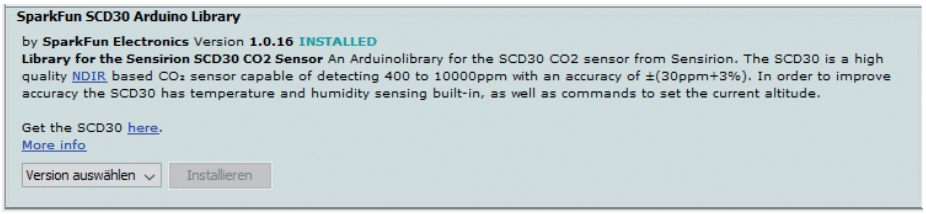
De U8g2-driver-bibliotheek voor het OLED-display wordt weer via *Tools -> Manage Libraries…* in de Arduino IDE geïnstalleerd; zoek naar *u8g2* in de rechter bovenhoek (**figuur 5**).
Na de installatie worden de drie kernbibliotheken automatisch in ons programma geïntegreerd met *Sketch\Include Library\… U8g2*.

```
#include <MUIU8g2.h> // delete
#include <U8g2lib.h> // delete
#include <U8x8lib.h>
```

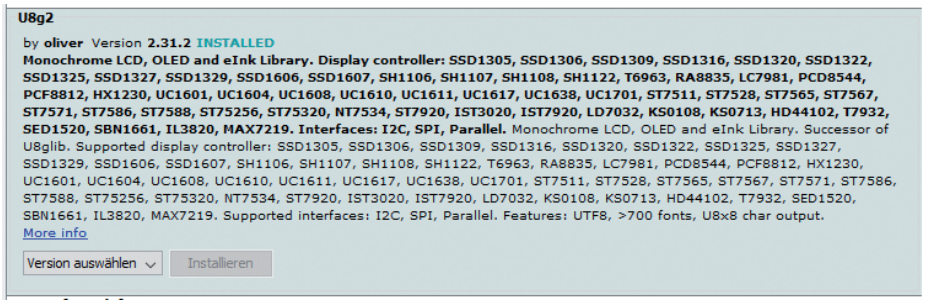
We hebben alleen de tekstgebaseerde *U8x8lib* nodig, dus de `#include` statements die naar de andere twee bibliotheken verwijzen, kunnen worden verwijderd.
Het voordeel van het U8g-bibliotheekpakket is (onder andere) het grote aantal ondersteunde displaytypes, waaronder:

- LC- en OLED-displays met grafische mogelijkheden;
- diverse pixelresoluties;
- besturing via verschillende bussen: I2C/SPI (hard- of softwarematig), parallelle bussen conform de 8080- en 6800-specificaties;
- diverse driver-componenten.

Hoewel het ongetwijfeld handig is om een overvloed aan ondersteunde displays te hebben, moet je er wel voor zorgen dat je de juiste kiest, anders krijg je problemen. In **figuur 6** zie je slechts een deel van de talloze mogelijke ondersteunde displaytypes (die zijn weggecommentarieerd).
Het is daarom zinvol om eerst een blik te werpen op een van de kant-en-klare voorbeelden die samen met de bibliotheek in de Arduino IDE zijn geïnstalleerd – bijvoorbeeld het *HelloWorld*-voorbeeld dat te vinden is onder *File -> Examples -> U8g2 -> U8x8*. Let erop dat je alleen voorbeelden selecteert voor de U8x8-bibliotheek!
In de lange lijst in deze *HelloWorld*-sketch, moeten we zoeken naar het hier gebruikte



Figuur 4. Selecteer de SCD30-bibliotheek van SparkFun.



Figuur 5. Kies de U8g2-bibliotheek om het 1,3" OLED-display aan te sturen.

Figuur 6. Kies de juiste uit dit grote assortiment van ondersteunde U8x8-displays.

(OLED-)display. Je activeert het door het `//` commentaarteken te verwijderen. We kopiëren gewoon de overeenkomstige (uitgecommentarieerde) regel 64 van de juiste display-constructor naar regel 119 [3] van onze CO_2-stoplichtsketch:

```
U8X8_SH1106_128X64_NONAME_HW_I2C
u8x8(/* reset=*/ U8X8_PIN_NONE);
```

en dat komt neer op:

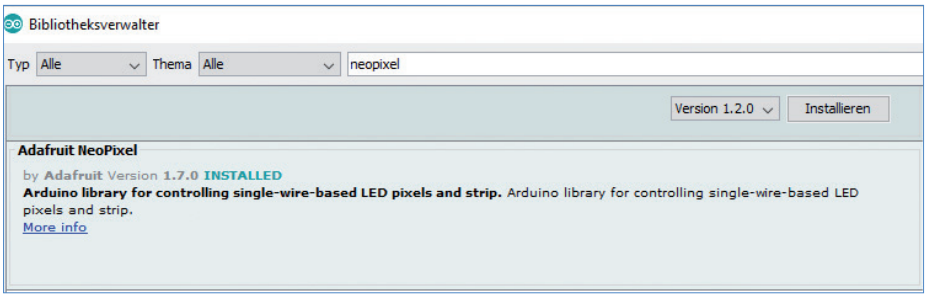
- driverbibliotheek: `U8X8`
- displaycontroller: `SH1106`
- aantal pixels: `128*64`
- fabrikant van het display: `NONAME` (we gebruiken een generiek merkloos display)
- hardware-interface: I²C
- naam van het object voor verwijzing in het programma: `u8x8`
- uitdrukking tussen haakjes: de resetpen van het display wordt niet gebruikt

Je kunt een gedetailleerde beschrijving van de bibliotheek krijgen door op *More info* te klikken (linker benedenhoek van figuur 5).

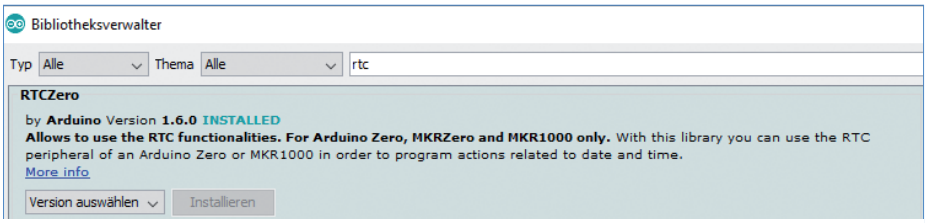
De NeoPixel-LED bibliotheek

Het NeoPixel LED-concept (ontwikkeld door Adafruit) maakt het mogelijk om grote, gekleurde LED-strips, vierkanten, cirkels en andere vormen samengesteld uit individuele RGB LED-elementen te construeren. Alle LED-elementen worden in serie geschakeld en gestuurd via een enkele digitale I/O-lijn. De WS2812 LED-controller maakt het mogelijk om

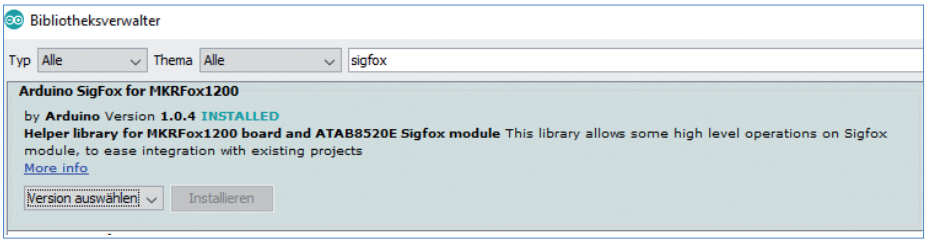
elk van de drie verschillend gekleurde individuele LED's (rood, groen, blauw) aan te sturen met programmeerbare 8-bit helderheid, zodat nagenoeg elke kleurencombinatie kan worden gegenereerd.
Het voordeel van het schrijven van software voor NeoPixel-toepassingen van willekeurige omvang is dat er een bijzonder handige Arduino-bibliotheek van Adafruit is die de aansturing verzorgt. Deze bibliotheek is ook opgenomen in het Arduino-programmapakket en wordt geïnstalleerd via de bibliotheekmanager die u al kent (zoek naar *neopixel*), zoals in **figuur 7**. Na installatie wordt de bibliotheek automatisch in ons programma geïntegreerd via *Sketch -> Include Library -> Adafruit NeoPixel*. Ook hier vindt u een gedetailleerde beschrijving van de bibliotheek door op de link *More info* te klikken.



Figuur 7. Installeer de Adafruit NeoPixel-bibliotheek.



Figuur 8. Selecteer de RTC-bibliotheek voor de SAMD21-microcontroller.



Figuur 9. Kies de Sigfox-bibliotheek voor het MKR FOX1200-microcontrollerboard.

De SAMD microcontroller RTC-bibliotheek
De SAMD21-microcontroller op het MKR FOX1200-board heeft een on-chip RTC die kan worden gebruikt voor 'systeembrede' tijd- en datumtoepassingen. Dit vereist een back-up batterij voor de print zodat de RTC niet telkens de tijd vergeet wanneer de voedingsspanning wordt uitgeschakeld. Wij hebben ervoor gekozen om geen back-up batterij in het CO_2-stoplicht op te nemen, maar een passende kleine schakeling kan gemakkelijk worden aangesloten op de MKR FOX1200-print.
Als je deze RTC wilt gebruiken, is daar natuurlijk weer een handige driverbibliotheek voor, die zoals gebruikelijk via de bibliotheekmanager geïnstalleerd kan worden (zoek naar *rtc*, **figuur 8**). Zoals gebruikelijk is een uitgebreide beschrijving van deze bibliotheek te vinden onder *More Info*. De bibliotheek wordt geïntegreerd via *Sketch -> Include library -> RTCZero*.

De MKR FOX1200 Sigfox-bibliotheek
Nu komen we bij de aansluiting van het CO_2-stoplicht op het Internet of Things met behulp van het Arduino MKR FOX1200-microcontrollerboard, dat met het Sigfox-netwerk 'praat'. Bekijk voor meer informatie over Sigfox de vierdelige introductie in Elektor [6].
Met Sigfox worden de meetwaarden van het CO_2-stoplichtsysteem draadloos doorgestuurd naar het internet, waar ze overal ter wereld met een PC, laptop, tablet of smartphone kunnen worden opgevraagd en weergegeven. Ook hier kunnen we gebruik maken van alle kant-en-klare functies in de Sigfox-bibliotheek die beschikbaar is in het Arduino-programmapakket. De bibliotheek wordt geïnstalleerd via de bibliotheekmanager, net zoals we dat met de andere hebben gedaan (zoek naar *sigfox*, **figuur 9**). Gebruik nu *Sketch -> Include Library -> Arduino Sigfox for MKRFox1200* om deze te installeren. Klik op *More info* om meer over de beschikbare Sigfox-functies te weten te komen.
Belangrijke fragmenten van het geannoteerde Arduino programma (voor het MKR FOX1200 bord) zijn te zien in **listing 1**. Het volledige programma in het Duits en Engels [3] is in detail gedocumenteerd en bevat nog enkele seriële uitvoerroutines die op veel plaatsen zijn ingevoegd voor debug-doeleinden (voor de seriële monitor in de Arduino IDE). Als je eenmaal tevreden bent dat alles werkt zoals het zou moeten, kun je deze codesecties wegcommentariëren of verwijderen.
Je bent helemaal vrij in het toekennen van drempelswaarden voor de gemeten CO_2-concentraties en de bijbehorende stoplichtkleur. Wij stellen deze drempelwaarden voor:

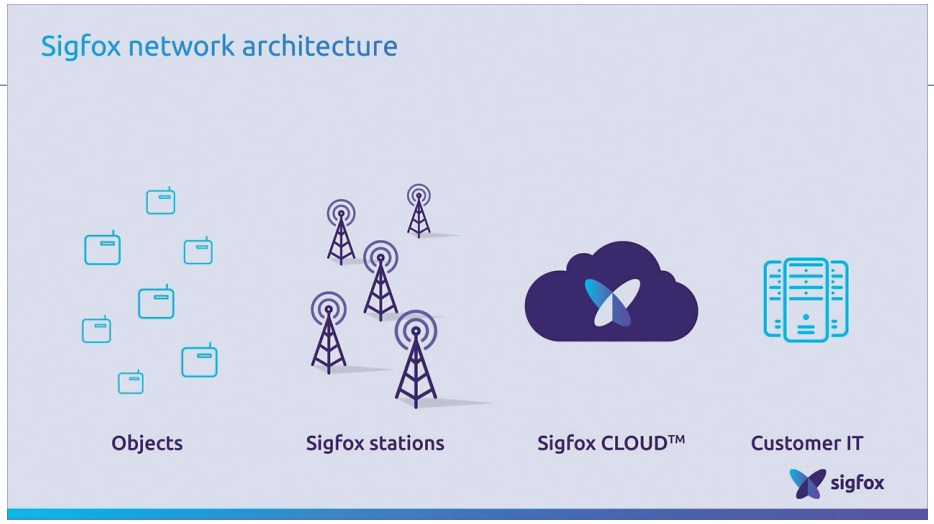
- 0 tot 1000 ppm: heldergroen
- 1001 tot 2000 ppm: geel
- 2001 tot 5000 ppm: oranje
- 5001 tot 12000 ppm: helderrood

Gebruik met het Sigfox-netwerk

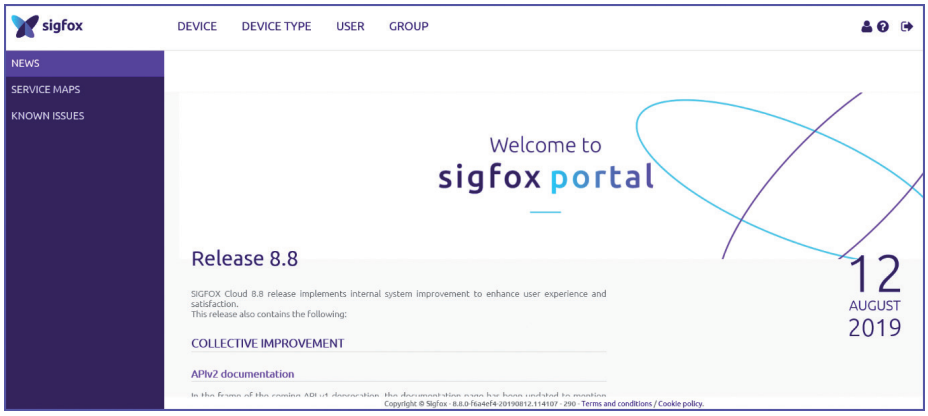
In [6] hebben we in detail laten zien hoe de parametrering en de werking van de MKR FOX1200-module in het Sigfox-netwerk in zijn werk gaat. Hier beperken we ons tot een korte beschrijving van het principe en de

Listing 1. Gedeelten van de Arduino-sketch.

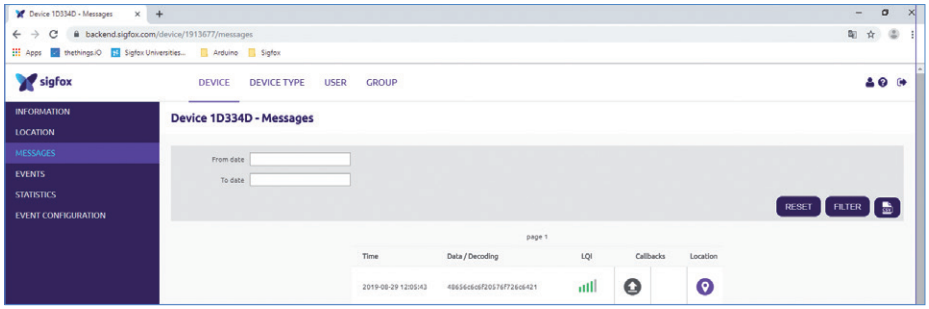
```
void loop()
{
  unsigned char i;
  char text[15];

  // loop
  while(1)
  {

    /* The SCD30 sensor measures automatically every 2 seconds.
       If data is available, then read it out and store it in variables*/

    if (AirSensor.dataAvailable())      // Check if new data is available
    {
      // If yes, then read the 3 measured values
      co2 = AirSensor.getCO2();
      temperatur = AirSensor.getTemperature();
      luftfeuchte = AirSensor.getHumidity();

      // Increase counter
      mess_zae = mess_zae + 1;

      // Print values on the serial monitor
      Serial.print("Measurement-Nr: "); Serial.print(mess_zae); Serial.print("   //   ");

      // Print time
      print2digits(rtc.getHours());
      Serial.print(":");
      print2digits(rtc.getMinutes());
      Serial.print(":");
      print2digits(rtc.getSeconds());
      Serial.println();

      // print measurements
      Serial.println("CO2:      " + String(co2) + " ppm");
      Serial.println("Temp:     " + String(temperatur-temp_cor) + " °C");
      Serial.println("Hum:  " + String(luftfeuchte) + " %rh");
      Serial.println();

      // Processing the CO2 value for the 'LED monitor': NeoPixel display
      switch(co2)
      {
        case 0 ... 1000:  pixels.fill(gruen_satt,0,7);
                          pixels.show();
                          break;

        case 1001 ... 2000: pixels.fill(gelb_1,0,7);
                            pixels.show();
                            break;

        case 2001 ... 5000: pixels.fill(orange_1,0,7);
                            pixels.show();
                            break;

        case 5001 ... 12000: pixels.fill(rot_satt,0,7);
                             pixels.show();
                             break;
      }

      // Output of the measured values on the OLED display
      // Output: CO2 value
      u8x8.setCursor(6,2);
      sprintf(text,"%5d",co2);   // sprintf only works with integers
      u8x8.print(text);

      // delete Phantoms
      u8x8.setCursor(11,2);
      u8x8.print(" ppm");
```



Figuur 10. Basisstructuur van het Sigfox-netwerk (bron: Sigfox).



Figuur 11. Het Sigfox-portaal (Sigfox Backend).



Figuur 12. Het berichtenvenster van het apparaat.

noodzakelijke stappen. We beginnen met een overzicht van het Sigfox-netwerk (**figuur 10**). De *Objects*, zoals onze CO_2-meter of een ander sensorboard, verzenden hun telegrammen conform het "fire-and-forget" principe via de licentievrije 868-MHz ISM-band. Naast een identificatie (afzender-ID) bevat elk telegram een gebruiker-dataveld (user data field) met een maximale grootte van 12 bytes, de zogenaamde payload. Een gebruiker kan bij elke transmissie maximaal 12 bytes aan meet-, status- of andere gegevens verzenden. Aangezien het Sigfox-netwerk in de licentievrije ISM-band werkt, mag elk object, om binnen de wettelijke voorschriften te blijven, slechts maximaal 140 transmissies per dag uitvoeren. Ons MKR FOX1200-board mag dus gemiddeld elke elf minuten een telegram versturen.

Afhankelijk van de dekking worden deze transmissies vervolgens ontvangen door alle Sigfox-basisstations (*Sigfox Stations*) binnen het bereik. Deze basisstations sturen alle ontvangen gegevens via internet of GSM-verbinding door naar de *Sigfox Cloud*, van waaruit de gebruiker zijn gegevens kan opvragen en verder evalueren en verwerken (*Customer IT*).

De interface om het gebruikersaccount te configureren wordt *Sigfox Backend* genoemd. Hier worden de Sigfox-objecten (apparaten) geregistreerd, groepen toegewezen en het doorsturen van gegevens naar *Customer IT* (via Callbacks) ingesteld.

Eerst moeten de unieke identificator *ID* en *PAC* van het Sigfox-apparaat worden uitgelezen met behulp van een kleine Arduino-sketch [3]. Deze twee parameters zijn nodig om het Sigfox-apparaat te registreren in de Sigfox Cloud op [7].

Tijdens bedrijf heb je wereldwijd toegang tot de telegrammen. Je meldt je aan met je e-mailadres en het gekozen wachtwoord bij het Sigfox-backend [7], dat je naar de startpagina van het Sigfox-portaal (Sigfox-backend) van **figuur 11** brengt. Hier klik je op het tabblad Device zodat er een lijst verschijnt van de actieve Sigfox devices. Als je vervolgens op het *Id*-veld van een apparaat klikt, kom je op de info-pagina van dat apparaat. Hier klik je nu aan de linkerkant op *MESSAGES* en kom je in het venster terecht waarin alle telegrammen opgesomd staan die het Sigfox-backend van dit apparaat ontvangen heeft (**figuur 12**).

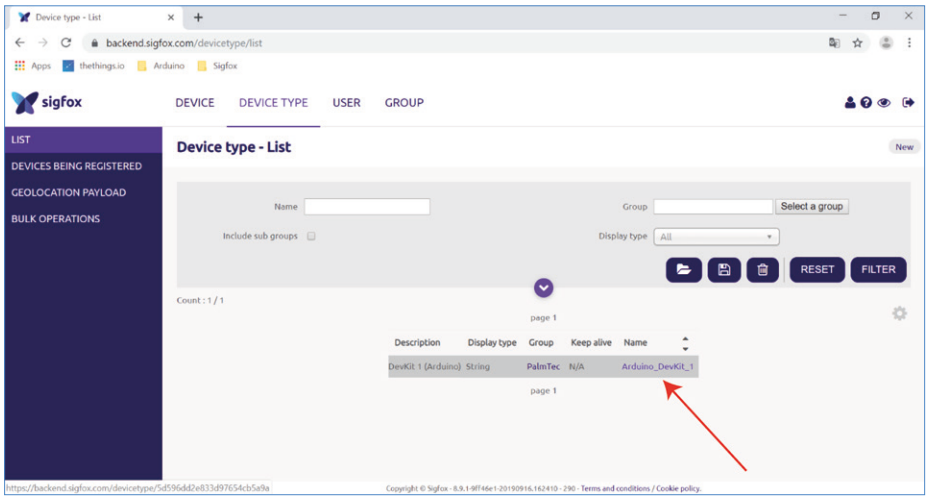
Callbacks creëren in het Sigfox-backend

Deze ruwe data is niet bijzonder zinvol. Zodra de MKR FOX1200 gegevens naar het Sigfox-backend stuurt, willen we dat deze automatisch worden doorgestuurd naar het *thinger.io* dashboard-programma. Het Sigfox-backend heeft hiertoe zogenaamde *Callbacks* in de aanbieding.

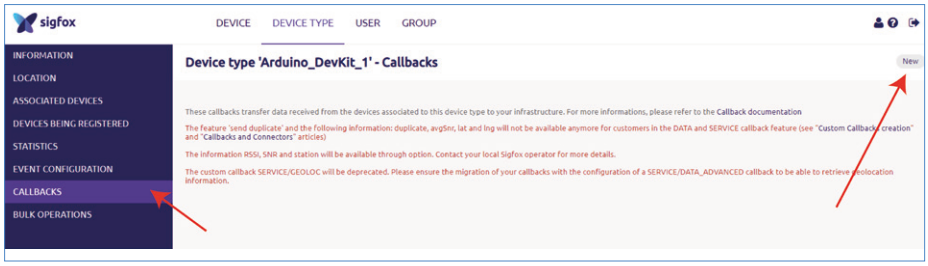
Zo'n callback is niets anders dan het automatisch doorsturen van data naar een ontvanger; dit vindt onmiddellijk plaats zodra het Sigfox-backend gegevens ontvangt van een Sigfox-apparaat – bijvoorbeeld van onze MKR FOX1200-module. Het aanmaken en configureren van een Callback werd in detail behandeld in het derde deel van de artikelreeks [6], daarom stippen we hier slechts de essentie aan.

Om een callback aan te maken, klik je op het tabblad *Device Type* op de hoofdpagina van het Sigfox-backend en in de lijst die verschijnt (met slechts één item) op de naam van het *Device Type*, in ons geval *Arduino_DevKit_1*

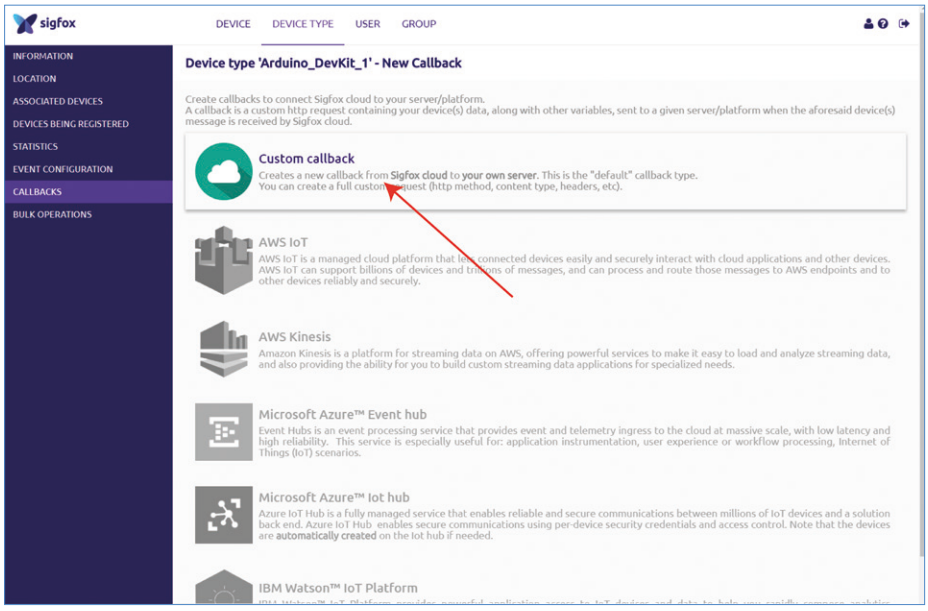
```
      // Output of the temperature value = float number
               oled_float(6,4,temperatur-temp_cor,1);

      // Output of the humidity value = float number
      oled_float(6,6,luftfeuchte,1);

      // Waiting times for measured value acquisition and transmission of the Sigfox telegram
      delay(w_zeit * 60000);            // (w_zeit * 1 Minute) wait between the measurements
      min_zae = min_zae + w_zeit;       // Minutes count until the next Sigfox transmission
      if (min_zae == SF_zyk)            // Now: send Sigfox telegram
      {
        SF_send_data();
        min_zae = 0;                    // Reset counter
} } } }

/*** Send data via Sigfox ***/

void SF_send_data(void)
{
  Serial.print("Sigfox - Start ... \n");

  // Write measured values to the data structure variable
  SF_Ampel.CO2    = co2;
  SF_Ampel.Temp   = temperatur;
  SF_Ampel.Feucht = luftfeuchte;


  // If required: Debug outputs
/*  Serial.println();
  Serial.print("CO2:     "); Serial.println(SF_Ampel.CO2);
  Serial.print("Temp:    "); Serial.println(SF_Ampel.Temp);
  Serial.print("Feucht:  "); Serial.println(SF_Ampel.Feucht);
  Serial.println();
*/

  // Activate Sigfox modem and query errors
  if (!SigFox.begin())      // Initializing the modem
  {
    Serial.println("Sigfox modem not found! - Continue with RESET!");
    while (1);  // loop
  }
  else
  {
    Serial.println("Sigfox modem initialization OK!");
  }

  // Enable debug LED and disable sleep modes
  SigFox.debug();

  // Clears all pending interrupts
  SigFox.status();

  // Send payload via Sigfox
  SigFox.beginPacket();               // Preparing to send a packet

  // Send structure variable to the Sigfox backend
  SigFox.write((char*)&SF_Ampel, sizeof(SF_Ampel));

  // Error checking: If endPacket() returns 'true': error
  SF_error = SigFox.endPacket()
  if(SF_error > 0)
  {
    Serial.println("Sigfox-Error !!");
  }

  // Sigfox end
  SigFox.end();

  Serial.println("Sigfox - End .... !\n");
}
```



Figuur 13. Kies het gewenste apparaattype.



Figuur 14. Het Callback-venster.



Figuur 15. Veel verschillende Callback-types.

(**figuur 13**). Klik in de overzichtspagina van het *Device Type* op *Callbacks* in de lijst aan de linkerkant (**figuur 14**) en rechtsboven in dit venster op de kleine knop *New*. Er verschijnt een lijst met alle mogelijke soorten callbacks voor de meest voorkomende typen dashboard-/cloud-programma's (**figuur 15**). Selecteer *Custom Callback*.

In het venster dat nu verschijnt (**figuur 16**) wordt de callback voor het verzenden van de gegevens van het Sigfox-backend naar *thinger.io* geconfigureerd.

Met deze callback stuurt de Sigfox backend altijd de variabelen `Device-ID`, `Telegr-Nr`, `CO2`, `Temp` en `Feucht` (vochtigheid) met hun waarden naar het dashboard-programma, en wel onmiddellijk na ontvangst van het telegram van het CO_2-stoplicht.

Een dashboard met Thinger.io

Thinger.io is een open-source IoT-visualisatieplatform waarmee je snel en eenvoudig een duidelijke weergave van data kunt creëren. Voor kleinere projecten kan het gratis worden gebruikt. De waarden worden gevisualiseerd op een dashboard, dat ook openbaar kan worden gemaakt voor webbrowsers.
Er zijn slechts een paar eenvoudige stappen nodig om een dashboard op maat te ontwikkelen:

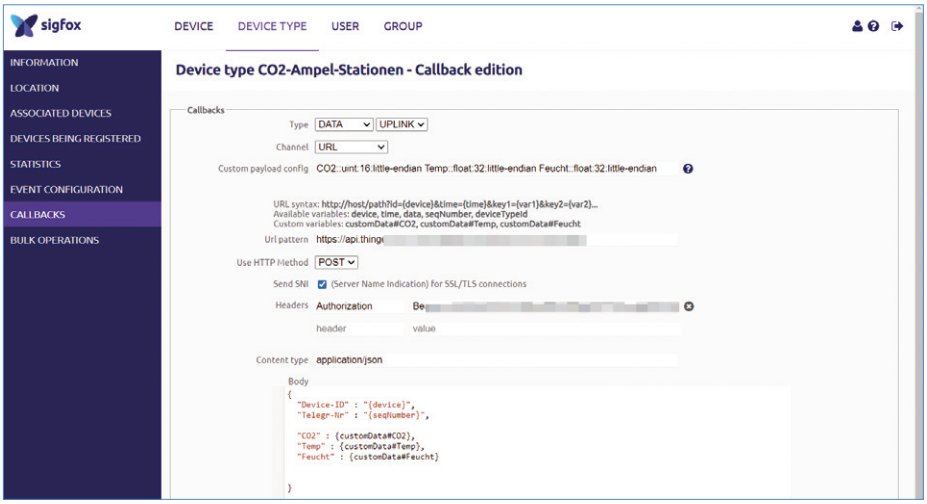
- maak een gratis *Account* aan bij thinger.io;
- creëer een gegevenscontainer (*Data Bucket*) bij thinger.io om de meetwaarden te bewaren die vanuit de Sigfox-cloud worden verzonden;
- definieer het toegangspunt (*Token*) voor de Sigfox-cloud en creëer een ontvangstverificatie (*Access Token*) bij thinger.io, zodat de Sigfox-cloud toestemming krijgt om gegevens naar thinger.io door te sturen;
- configureer een Callback in de Sigfox-cloud om de gegevens van die cloud via het internet naar de *Data Bucket* in thinger.io over te brengen;
- ontwerp een fraai dashboard op de thinger.io-site om de gegevens te visualiseren.

Ook hier verwijzen we naar [6], waar een meer gedetailleerde beschrijving van de afzonderlijke stappen wordt gegeven. Na het aanmaken van een gratis account bij thinger.io [8] log je in en kom je bij het hoofdscherm, dat het uitgangspunt is voor alle verdere acties (**figuur 17**).
Om een dashboard te maken, klik je links op het item *Dashboards* en in het dan verschijnende venster op *Add Dashboard*. Voor de individuele vormgeving van een dashboard staat een groot aantal vrij configureerbare widgets (de componenten van de gebruikersinterfacen) ter beschikking, bijvoorbeeld grafieken die de meetwaarden als functie van de tijd tonen, taart- en staafdiagrammen, analoge displays, Google Maps om de

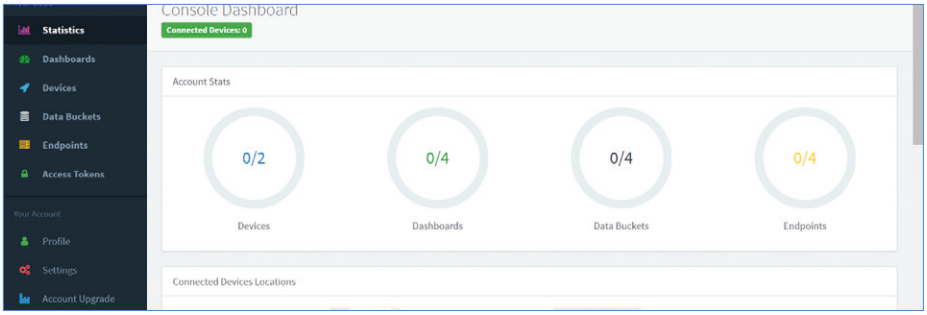
locatie te tonen, afbeeldingen, tekstdisplays, LED's en een klok.

Figuur 18 toont een voorbeeld van een dashboard dat is ontworpen met thinger.io voor ons CO_2-stoplicht. De gemeten waarden worden weergegeven in de vorm van een tijdgrafiek. Naast deze gegevens kunnen ook de tijd, het telegramnummer en andere informatie worden getoond. Wereldwijde toegang tot het dashboard is met een paar muisklikken mogelijk, zodat iedereen die de link ontvangt vervolgens het dashboard kan lezen en de actuele waarden kan controleren.

200650-03



Figuur 16. Callback-configuratie.



Figuur 17. Accountstatistieken op het console-dashboard.

Figuur 18. Het CO_2-stoplichtdashboard.

GERELATEERDE PRODUCTEN

- **Arduino MKR FOX 1200 (SKU 19096)**
  www.elektor.nl/19096
- **Arduino Uno Rev3 (SKU 15877)**
  www.elektor.nl/15877
- **Seeed Studio Grove SCD30 CO2, Temperature & Humidity Sensor for Arduino (SKU 20012)**
  www.elektor.nl/20012

WEBLINKS

[1] CO_2-sensor SCD30: https://bit.ly/34XbL5o
[2] 1,3" OLED-display: https://bit.ly/3fEP7AX
[3] Projectpagina bij dit artikel: www.elektormagazine.com/200650-03
[4] Informatie over NeoPixel LED-displays: https://bit.ly/3qf1Y2k
[5] Details van je dev-kit: https://buy.sigfox.com/activate/devkit/DE
[6] Frank Schleking, Bernd vom Berg, "Met de vos in het IoT (1)", Elektor november 2019 (delen 2...4 in de drie daarop volgende nummers): www.elektormagazine.nl/magazine/elektor-116/56888
[7] Sigfox backend login-pagina: https://backend.sigfox.com/auth/login
[8] Bouw je eigen dashboard met thinger.io: www.thinger.