Project 4.0

-

Team F2

Teamleden:

- Christophe Van Hoof
- Sander Philipsen
- Eloy Boone
- Alexander Meuris
- Yen Aarts,
- Bobby Fonken

Woord vooraf

De installatie handleiding dat voor u ligt is een uitwerking van ons Proof of Concept voor Project 4.0 aan de Thomas More hogeschool te Geel.
In dit rapport vindt u de uitwerking van onze PoC om verschillende zaken te monitoren op één van de productielijnen bij Agricon te Ham.
Aan de hand van dit rapport zou u ons project moeten kunnen reconstrueren.

Graag willen wij een bijzonder woord van dank richten aan de firma's Agricon en 3-it voor hun interessante en leerrijke opdracht. Ook voor hun steun, infosessies en rondleiding op het terrein zelf willen wij hen bedanken.

Secoda

Inhoudsopgave

Inleiding

Het volgende document legt uit hoe iemand onze PoC kan reconstrueren.

De opdracht die wij ontvingen ging als volgt.
We gingen een uitwerking maken waarbij de sensor data vanaf een Arduino verzonden werd naar een Raspberry Pi 3 over Wi-Fi.
De Arduino zou functioneren als een zogenaamde sensor hub. De Raspberry Pi 3 diende als een gateway simulator. Het is de connectie naar de Cloud en zou ook de nodige zwaailichten en buzzers aansturen.
De metingen zouden dan getoond worden in een dashboard.

Het eerste dat er wordt uitgelegd, is hoe de sensor hub is opgebouwd, met de gebruikte technologieën en software.

Dit wordt gevolgd door uitleg over hoe men onze Azure omgeving moet nabouwen.
Dit omvat de volgende onderdelen: Cosmos DB, IoT Hub, App services en Functions.

Daarna wordt er uitgelegd hoe het dashboard is opgebouwd. Uit wat deze bestaat en hoe deze precies werkt. Dit is opgesplitst in drie delen: Front-end, Backend en daarna Chartjs.

Tenslotte wordt er uitgelegd hoe men de gateway kan voorbereiden om metingen te ontvangen en versturen. Met daarnaast ook nog hoe er data zal worden ontvangen.

Achteraan het document zal er een hoofdstuk de referenties bevatten voor eventuele extra uitleg bij onderdelen.
Dat wordt voorafgegaan door een lijst met software en de gebruikte versies ervan.

1 Sensor hub

1.1 Technische gids en handleiding voor her programmatie

De opdracht was alle sensoren koppelen met een platform dat dienst doet als sensor hub, de sensordata wordt erna verstuurd van de hub via wifi naar de gateway. De sensoren en de wifimodule worden gevoed door een aparte breadboard powersupply, dit komt omdat de wifi module redelijk veel stroom gebruikt.
Verder is er ook een 7 segment display op de sensor hub aangesloten, deze geeft foutcodes indien een of meerdere sensoren niet werken.

Dit onderdeel zal diep ingaan op de aansluiting en codering van elke sensor zodat de lezer gemakkelijk aanpassingen in de code kan uitvoeren. Onze POC is immers niet rechtstreeks implementeerbaar en er zullen dus veel veranderingen aan de code moeten gebeuren.

1.1.1 Foutcodes

Elke sensor heeft zijn eigen binair getal. De som van alle niet werkende sensoren zal op de 7 segment display zichtbaar zijn. Door dit getal te ontleden in een binair getal kan je dus zien welke sensoren werken en welke niet.

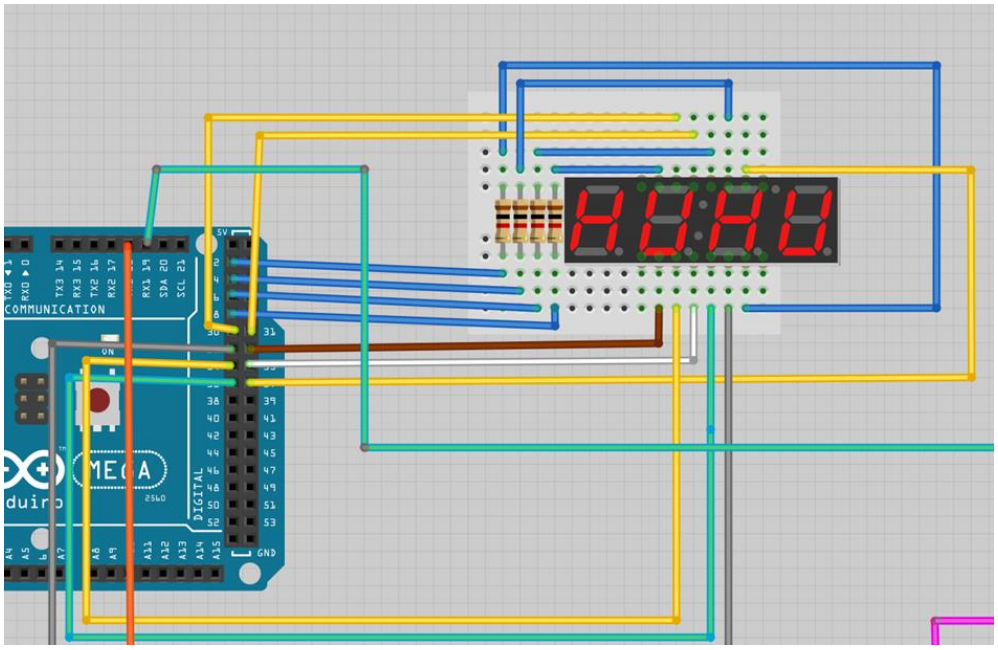
- Wifi: 1
- Ultrasoon sensor: 2
- Temperatuursensor: 4
- Vochtsensor: 8
- Infraroodsensor: 16
- Stofsensor: 32

Als er bv een 11 op de display staat, betekend dit dat de vochtsensor, ultrasoon sensor en wifi niet werken.

1.1.2 7-segment display

Hieronder wordt de werking van de display voor foutcodes uitgelegd.

1.1.2.1 Bedrading

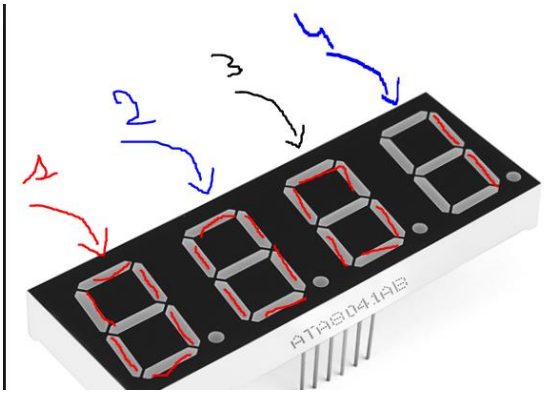


1.1.2.2 Programmatie

Een 7 segment display maakt gebruik van multiplexing. Elk getal wordt dus om de beurt ingesteld. Eerst het eerste, dan het tweede dan het derde en dan het vierde. De getallen worden dus niet tegelijk getoond maar heel snel achter elkaar. Om deze reden heb ik een “timerinterrupt” geïmplementeerd in de code. De interrupt zorgt ervoor dat om de 5000 microseconden een getal naar de display wordt geschreven.

1.1.2.3 Schematische voorstelling

Als er een getal 1 op de display getoond wordt dan zal dus eerst een 0 op positie 1 getoond worden, dan een nul op positie 2 dan een nul op positie 3 en uiteindelijk een 1 op positie 4. Om 1 getal te tonen moet er dus continu op heel korte tijd een lus herhaald worden zodat het voor het oog lijkt dat deze getallen tegelijk getoond worden.



1.1.2.4 Code

Om het bovenstaande effect te krijgen dat de display constant hernieuwt zonder de werking van de rest van het programma te onderbreken hebben we gebruik gemaakt van een “timerinterrupt”. Deze onderbreekt het programma op een afgesproken tijd, voert een taak uit en laat dan het programma verder doen.

Initialisatie van display en interrupt

```
//display

byte numDigits = 4;
byte digitPins[] = {28, 26, 24, 22};
byte segmentPins[] = {30, 37, 36, 34, 33, 31, 32, 35};

bool resistorsOnSegments = true;
bool updateWithDelaysIn = true;
byte hardwareConfig = COMMON_CATHODE;
sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins, resistorsOnSegments);
sevseg.setBrightness(200);


//timer interrupt van 7 seg display
Timer1.initialize(5000); //om de 5000 miliseconden display refreshen
Timer1.attachInterrupt(disp);
```

Functie interrupt

```
void disp(){
    sevseg.setNumber(error, 4);
    sevseg.refreshDisplay();
}
```

1.1.3 Wifi

De data van de sensoren wordt draadloos doorgestuurd via een esp 8266 module gekoppeld aan de Arduino. Op de esp 8266 staat AT firmware geïnstalleerd.
Op die manier kan de Arduino at-commando's sturen via rx en tx. In het setup deel van de code wordt de wifi module geconfigureerd als client.
Daarna wordt er een UDP-connectie opgezet met de gateway.
In het "loop" gedeelte wordt er een string in JSON-formaat doorgestuurd naar de gateway.

1.1.3.1 Code setup

In dit stuk code wordt de esp module geïnitialiseerd, op het netwerk gezet en wordt er een UDP connectie naar de gateway opgezet.
De esp module communiceert op lijn Serial1.

```
Serial.begin(9600);
Serial1.begin(115200);
Serial.print("staat op");
sendCommand("AT", 5,100);

//esp module resetten
sendCommand("AT+RST",3,500);
//runnen van configuratie (netwerk, sockets en udp openzetten)
sendCommand("AT+CWMODE="+ mode, 5, 500);
;
sendCommand("AT+CWJAP=\"" + network + "\",\"" + passwd +"\"", 5,2500);
//checken naar wifi connectie
checkwifi();
sendCommand("AT+CIPMUX=1", 5,500);


//udp transmssie opzetten
Serial.println("test2");
sendCommand("AT+CIPSTART=1,\"UDP\",\""+ server + "\","+port+",1112,0",1,5000);
```

Functie sendCommand

De functie "sendCommand" hebben we geprogrammeerd, zodat er commando's kunnen gestuurd worden naar de esp module. Deze worden kunnen dan worden gecheckt of het commando succesvol is uitgevoerd.
Er wordt hierbij meegegeven hoeveel het commando opnieuw verstuurd moet worden na een fail (max time) en hoe lang er gewacht moet worden op een antwoord (answer).

```
void sendCommand(String command, int maxTime, int awnser){
  //werkt voor alle commands die OK returnen
  //commandos worden tot maxTime keer verzonden

  while(countcommand < maxTime){
    Serial1.println(command);
    delay(awnser);

    if(Serial1.find("OK"))

    {

      break;
    }
    countcommand ++;
  }
  countcommand = 0;


}
```

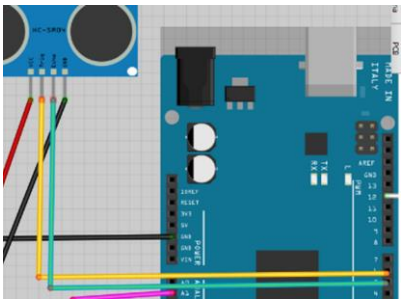
Code doorsturen

Hiermee wordt de JSON string met alle info over de sensoren en de gemeten waardes in totaliteit doorgestuurd naar de gateway.

```
sendCommand("AT+CIPSEND=1,"+ lendata, 1, 0);
Serial1.println(totaal);
delay(10000);
```

1.1.4 Ultrasoon sensor

De ultrasoon sensor wordt gevoed door 5V en is op de volgende manier aangesloten.



Code

Voor het programmeren is er geen gebruik gemaakt van enige library, enkel van deze zelf geschreven functie.

Code

Voor het programmeren is er geen gebruik gemaakt van enige library, enkel van deze zelf geschreven functie.

```
int distance1(){

  noInterrupts();
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  interrupts();
  distance= duration*0.034/2;
  if (distance !=0){
    error+=0;
    statdist = "1";
  }
  else{
   error+=2;
   statdist = "0";
  }
  return distance;


}
```

LET OP: Als er aanpassingen gebeuren aan deze code of het is gewenst een library te gebruiken om een ander type ultrasoon sensor te benutten.
Let dan goed op om de lijnen “noInterrupts()” en “interrupts()” te behouden.
Een ultrasoon sensor werkt immers steeds met een strikte timing.
Het uitvoeren van een interrupt kan dus grote gevolgen hebben voor de werking van de sensor.

1.1.5 DHT sensor

De DHT sensor wordt gevoed door 5V en is een "1 wire sensor".
Dit betekend dat alle data over één draad verstuurd wordt naar de hub. De sensor meet zowel temperatuur als vocht. Om de sensor uit te lezen hebben we gebruik gemaakt van de "DHT library".
Bijgevolg is de code voor het uitlezen van deze sensor bijzonder simpel. In de opstelling is de sensor gekoppeld op pin 3.

1.1.5.1 Initialisatie sensor

```
#include "DHT.h"
#define DHTPIN 3
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

1.1.5.2 Uitlezen data

```
String temp(){
  float t = dht.readTemperature();
  String Temp = String(t);
  //aanpassenn na tussen waardes
  if (Temp !="NAN"){
    stattemp = "1";
  }
  else {
    error +=4;
    statdist = "0";
  }
  return String(Temp);
}
String hum(){
  float h = dht.readHumidity();
  String hum = String(h);
  if (hum !="0.00"){
    stathum = "1";
  }
  else{
    error+=8;
    stathum = "0";
  }
  return String(hum);
}
```

Aangezien we gebruik maken van een library specifiek voor deze sensor zal er dus bij gebruik van een andere sensor ook een andere library gebruikt moeten worden.
Indien er voor deze sensor geen library beschikbaar is , kan je deze library aanpassen (deze is immers Open Source) om een andere "1 wire sensor" mee uit te lezen.
De broncode van deze library vindt u op deze volgende webpagina:
https://github.com/adafruit/DHT-sensor-library

1.1.6 Infrarood sensor

De IR-sensor meet of er iets ervoor aanwezig is of niet.
Deze zal naast de magneet hangen om zo een obstructie aan de magneet te kunnen waarnemen.
De code voor deze sensor is zeer simpel. Er worden geen libraries gebruikt, dus deze vervangen door een andere variant zou niet veel aanpassing in de code mogen teweegbrengen. Tenzij deze een ander protocol gebruikt.

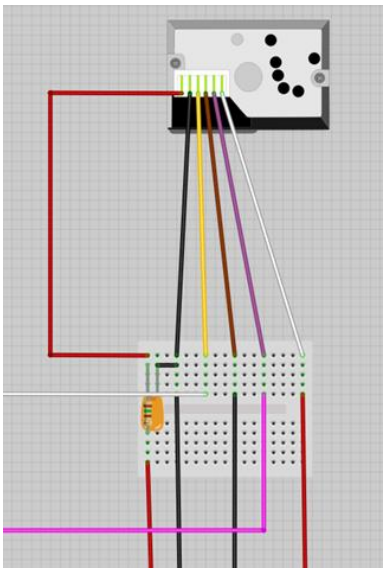
De gebruikte sensor is van het type "grove ir distance interrupter" en wordt aangesloten op pin 2 en wordt gevoed met 5V.

```
String ir(){
  String IRsens;
  if(digitalRead(irsens)==LOW)  {

      IRsens = "1";
    }
   else {
      IRsens = "0";
    }
     return String(IRsens);

}
```

Door de aard van deze sensor was het niet mogelijk om met de middelen die we ter beschikking hadden te kunnen testen of de sensor werkt of niet.
Hierdoor is dus de errordetectie hard coded.
Het is aan te raden om een sensor te implementeren waarvan de werking gecontroleerd kan worden.

1.1.7 Stofsensor

De gebruikte stofsensor is een "optical dust sensor".
De aansluiting kan je vinden in de onderstaande afbeelding.
De witte kabel gaat naar pin 12 en de paarse naar pin A1. De rode en zwarte kabels gaan respectievelijk naar 5V en GND.



1.1.7.1 Code

```
float stof(){
  noInterrupts();
  digitalWrite(ledPower,LOW); // power on the LED
  delayMicroseconds(samplingTime);

  float voMeasured = analogRead(measurePin); // read the dust value

  delayMicroseconds(deltaTime);
  digitalWrite(ledPower,HIGH); // turn the LED off
  interrupts();
  if (voMeasured != 0){
    statdust = "1";
  }
  else {
    statdust = "0";
    error += 32;
  }
  return voMeasured;
}
```

LET OP: Een optische stofsensor maakt gebruik van een led die op strikte timing intervallen zal zenden en ontvangen. De lijnen "noIntterupts()" en "interrupts()" zijn dus zeer belangrijk om tot een nauwkeurige meting te komen.

1.1.8 Volledige programmacode en implementatietips

De volledige programmacode, de firmware van de wifi module en het volledige bedradingsschema kan je vinden in de bijlagen.

We raden de persoon aan die het POC omzet in een implementeerbaar resultaat om met de volgende dingen rekening te houden.

Het systeem werkt momenteel met een hub waarop alle sensoren aangesloten worden. Elke keer als er nu een sensor wordt toegevoegd moet de hub dus opnieuw geprogrammeerd worden.
Dit is niet mogelijk in een echte opstelling, daarom raden we aan om ofwel een hub per type sensor te gebruiken (bv. Een "1-Wire hub") waar een reeks poorten reeds op voorgeprogrammeerd zijn. Op die manier kan men gewoon een sensor toe voegen.
Ofwel is het ook mogelijk om elke sensor apart draadloos naar de gateway te sturen. Een combinatie van de twee is ook mogelijk.

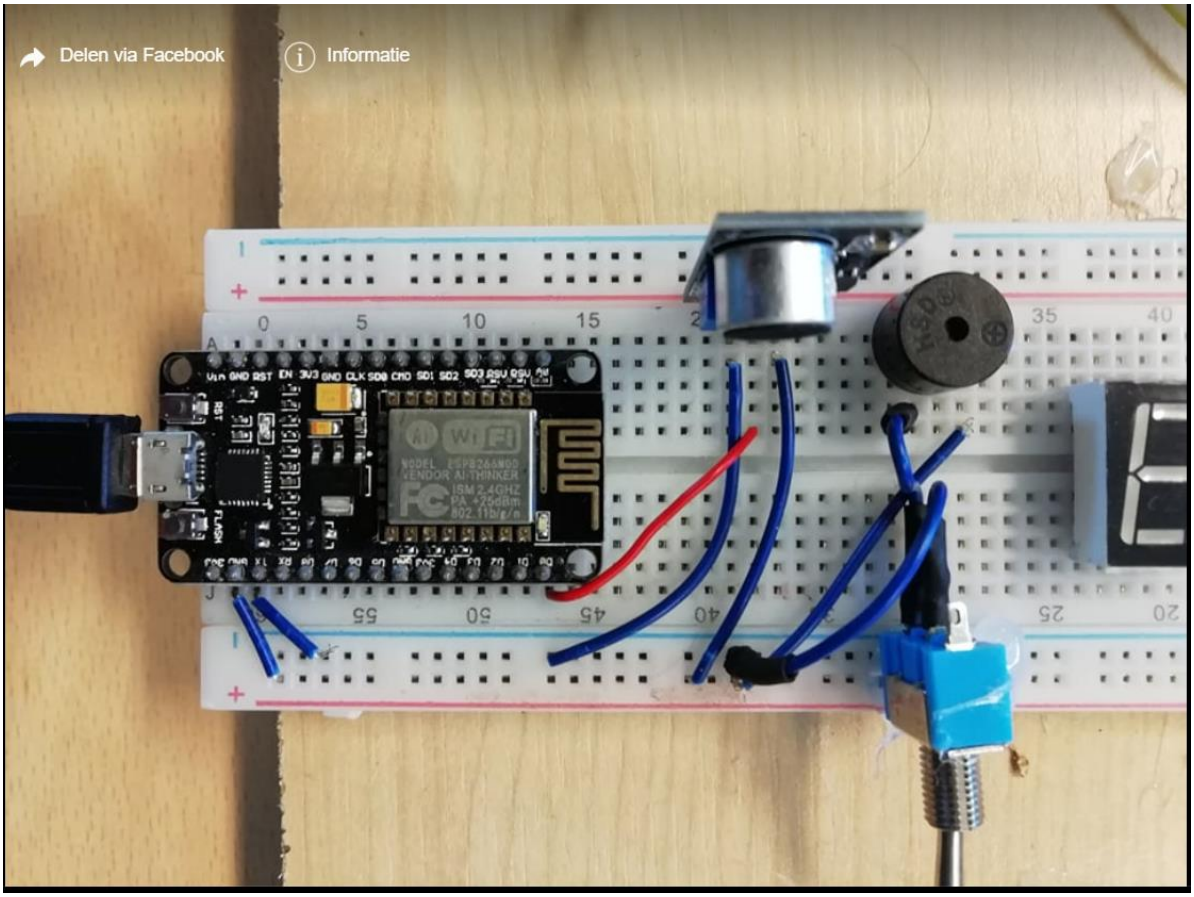
In het volgend deel zal de laatste optie verder beschreven worden.

1.2 Schaalbaarheid van het systeem

Om het systeem implementeerbaar en schaalbaar te maken zijn er twee grote opties.
Ofwel moet er gebruik gemaakt worden van een hub met voorgeprogrammeerde poorten per type sensor.
Ofwel elke sensor apart draadloos zijn data laten doorsturen. Aangezien de eerste methode niet kostenefficiënt is en verder zorgt voor een vermindering in schaalbaarheid, hebben we de tweede methode reeds onderzocht en gebouwd.

We hebben hiervoor een esp module gebruikt met enkele pinnen voor IO en deze geprogrammeerd om data naar een UDP poort door te sturen in een netwerk.
We hebben dit deel niet geïmplementeerd binnen Azure omdat de sensor niet echt nut had binnen deze proefopstelling. Het zou echter wel perfect mogelijk zijn aangezien de gateway de sensordata kan ontvangen.

De code is terug te vinden in de bijlagen. Hieronder vindt u een afbeelding van deze opstelling.

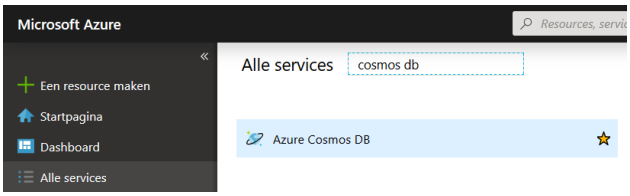


2 Azure omgeving

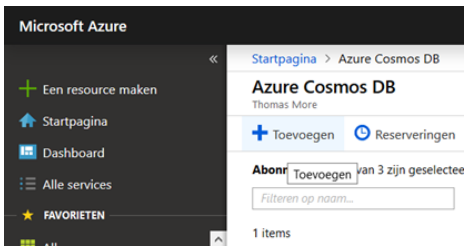
Hieronder wordt beschreven hoe de Azure omgeving omgebouwd moet zijn. We geven er ook uitleg bij waarom we sommige structuren maken.

2.1 Cosmos DB

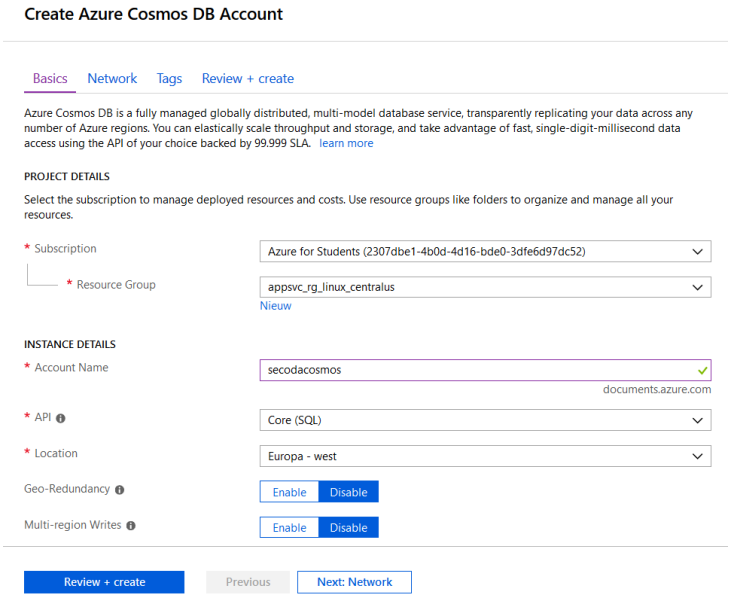
We beginnen ons Azure verhaal met het maken van een database met Cosmos DB. Ga naar "**Alle services**" in het linkse menu en zoek naar Cosmos DB.



Kies dit en klik dan op de knop "**Toevoegen**".

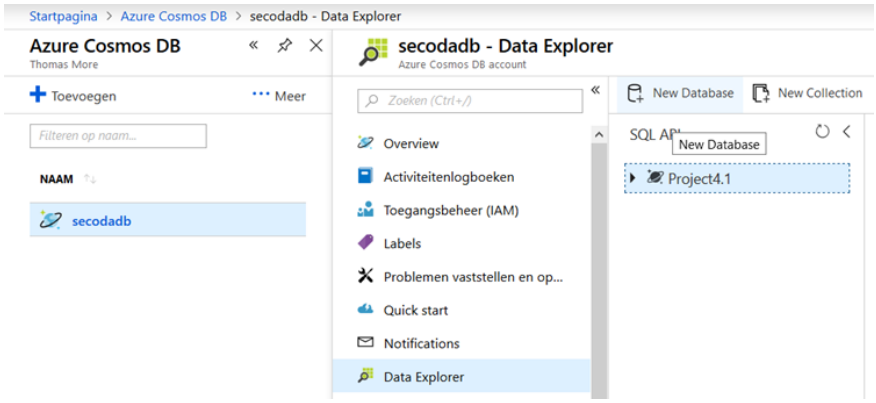


Je komt op volgende pagina terecht. Vul de nodige informatie in en klik op "**Review + create**". Zorg ervoor dat je "**Core (SQL)**" aanduidt.
Voor de "**Resource Group**" als er nog geen bestaat, kan je een nieuwe maken. Anders probeer je een bestaande groep te kiezen.

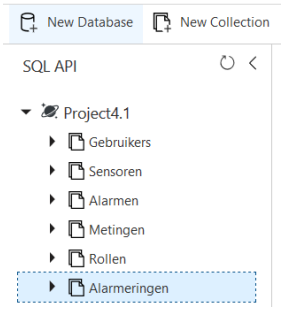


Je hoeft dan gewoon te wachten tot het implementeren voltooid is.

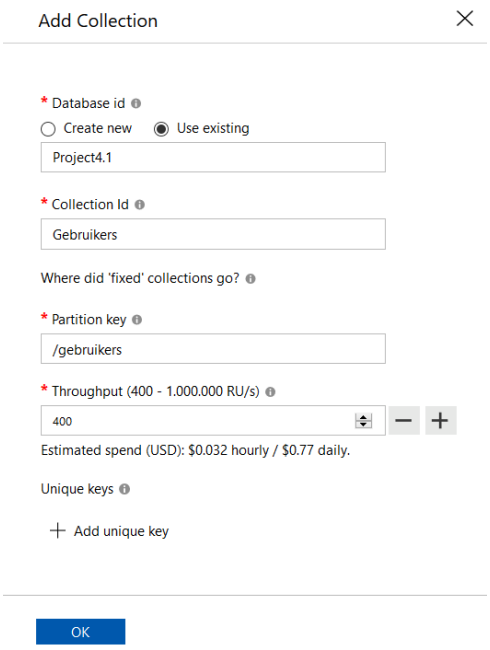
Ga vervolgens naar "**Data Explorer**" in je nieuw gemaakte Cosmos DB en klik op "**New Database**".



Vul de naam: "**Project4.1**" in een klik op "**OK**".
Vervolgens maak je volgende collecties aan. Dit doe je door op "**New Collection**" te klikken.



Er zal een menu rechts open gaan. Hier vul je op basis van de database en collectienaam volgende informatie in. Doe dit voor de andere collecties ook.



Vervolgens dien je even wat manueel werk te verrichten. In de collectie "**Alarmen**" zullen we de voor gedefinieerde beschikbare alarmen zetten. Op de gateway unit (Foto te zien bij: Bedradingen schema) kan je zien hoeveel aansluitingen er zijn. Hier staan nummers bij vermeld.
Dit zijn de id's. Deze dien je met volgende structuur toe te voegen in een document in de collectie "**Alarmen**".

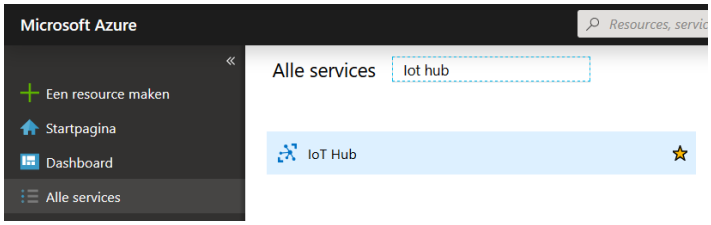
De andere waarden kan je veranderen op basis van waar de component (LED of buzzer) komt te staan. Klik bij elk document op de "**Update**" knop bovenaan.

```
{
"id": "1",
"naam": "Zwaailamp",
"type": "led",
"locatie": "voorraadkamer"
}
```

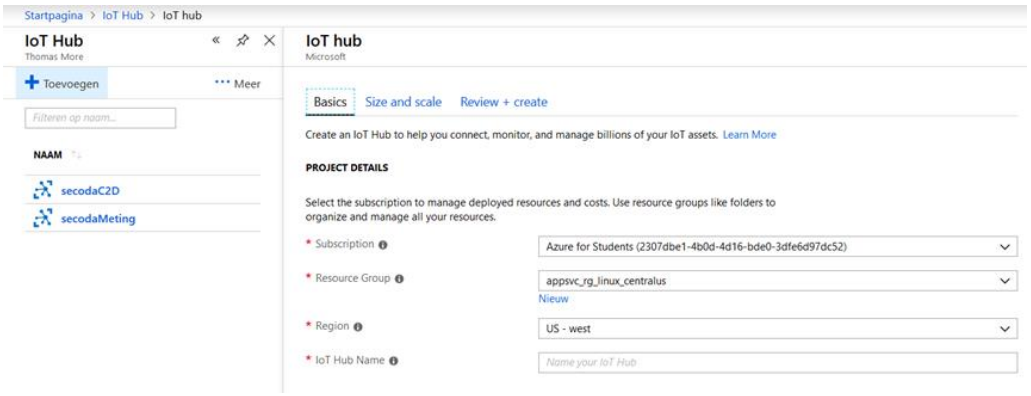
Ook al zou de koppeling leeg blijven, voeg je deze al toe. Dit maakt niet uit voor de werking of de LED verbonden is of niet. Dit is makkelijk na te gaan bij de gateway zelf als er niks aanhangt.

2.2 IoT Hub

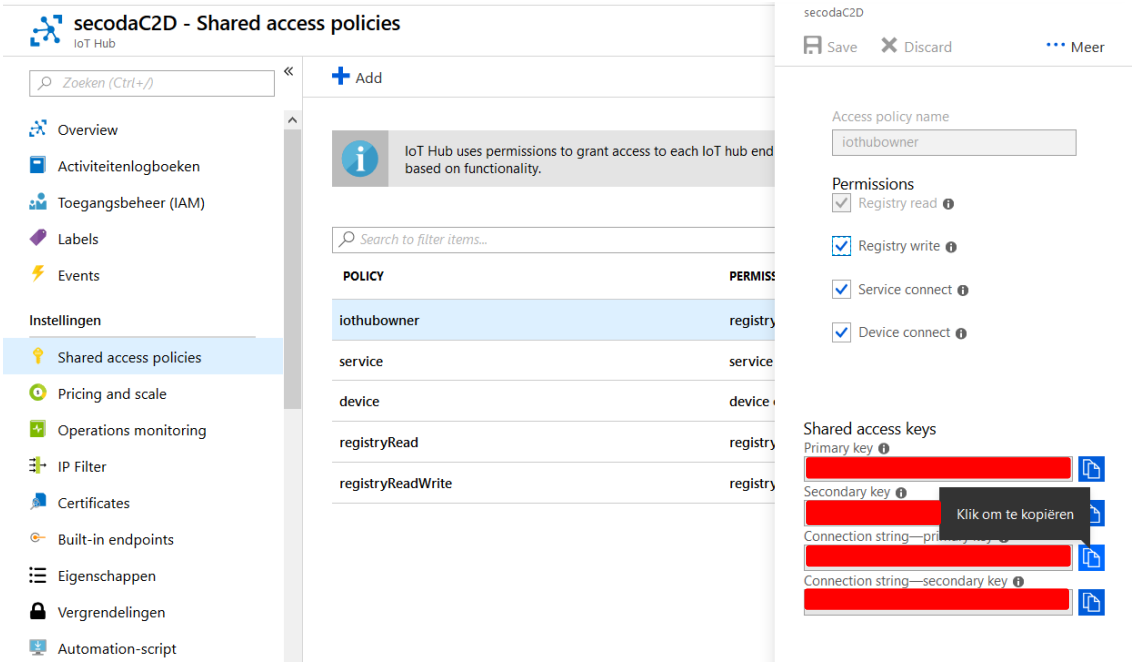
Hieronder maken we de nodige IoT Hub structuur. Zoek in "**Alle services**" naar IoT Hub.



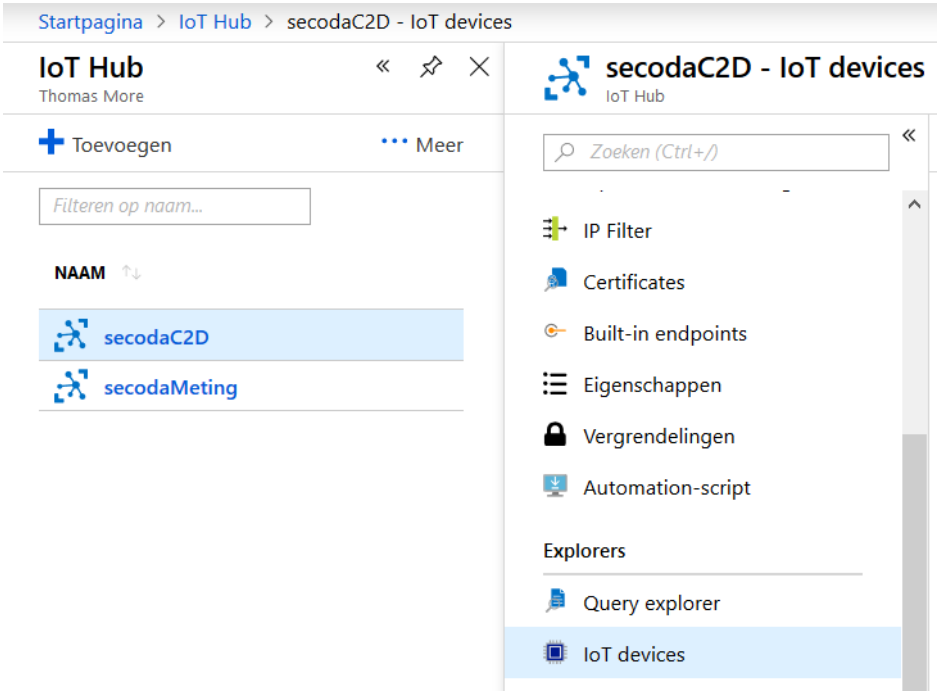
Klik op toevoegen en vul de nodige informatie in. De naam van de hub voor duidelijke redenen noem je iets met "C2D" in. Deze wordt gebruikt voor "Cloud to Device" berichten.



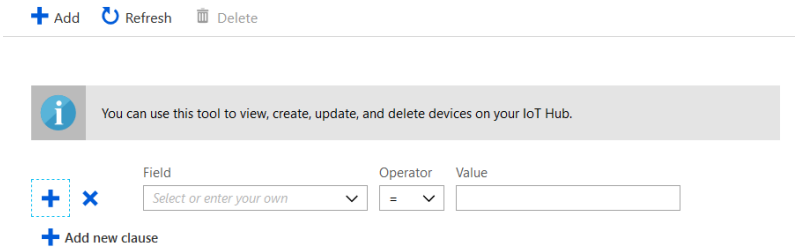
Klik op "**Review + create**".
Alvorens verder te gaan kijk je eerst bij "**Shared access policies**". Hier klik je op de "**iothubowner**". Daar kopieer je de "**Connection string—primary key**". Deze heb je later nodig in de backend.



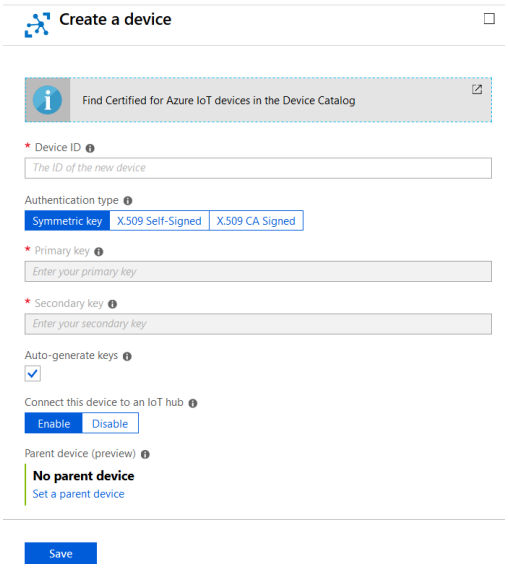
Vervolgens ga je naar "**IoT Devices**".



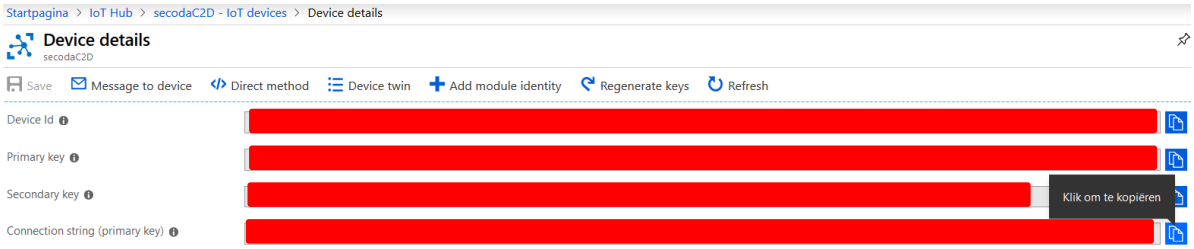
Druk hier op de bovenste "**+ Add**" knop.



Kies een "Device ID" (bijvoorbeeld: SecodaC2D). De rest wordt automatisch aangemaakt.



Hier klik je op het aangemaakte device en ga je de connection string opslaan. Deze is nodig voor de Python scripts op de gateway.



Dit doe je nog eens maar dan geef je de naam: "secodaSensor". Deze zal de status van de sensoren ontvangen. Meer hierover later.

Je maakt daarna een tweede IoT Hub aan. Deze wordt gebruikt voor het ontvangen van de sensor metingen. Hier maak je één device aan. De naam is vrij te kiezen. Probeer het logisch te houden.

We splitsen dit zodat de communicatie van de metingen gescheiden is van de andere. De metingen gaan al genoeg trafiek genereren voor één IoT Hub.

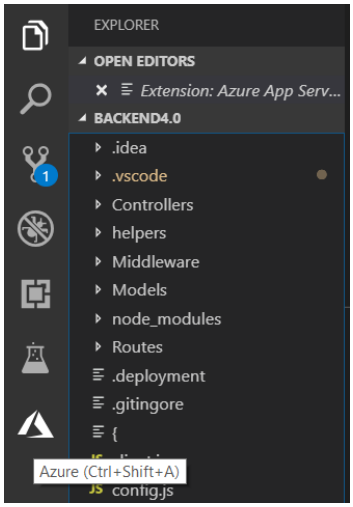
Bewaar zeker alle drie de connection strings!

2.3 App services

Onderstaande onderdelen leggen de manier die wij gebruiken om onze websites te hosten uit.

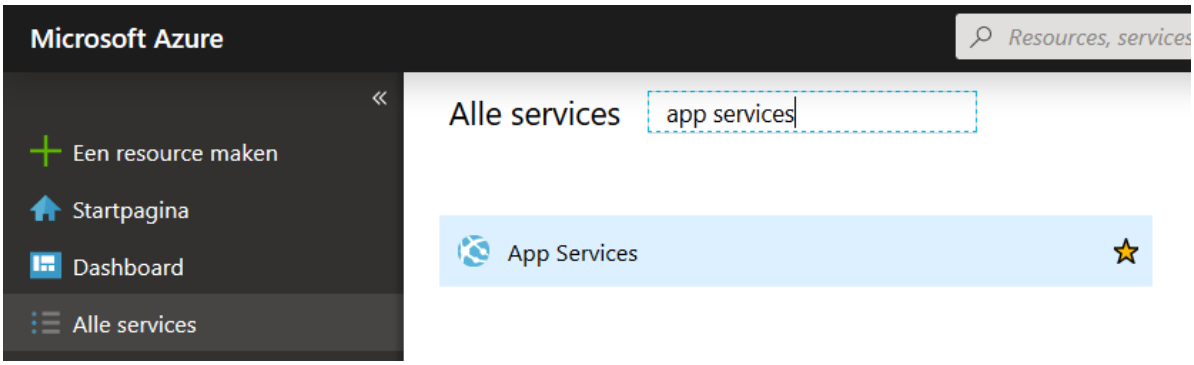
Voor men hieraan begint, is het makkelijker om nu al volgende in VS Code extensie te installeren. We maken gebruik van volgende extensie: "**Azure App Service**".

Na de installatie dien je je aan te melden met je Azure account. Ga naar het Azure symbool links.

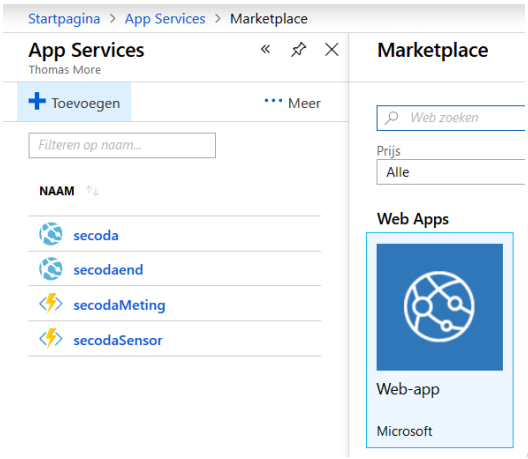


2.3.1 Front-end: Angular 6

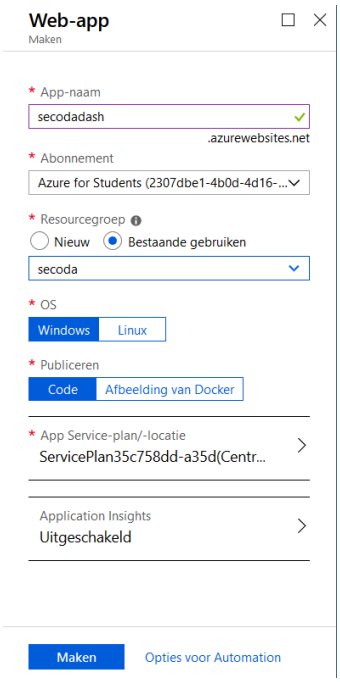
We zullen beginnen via "**Azure Portal**" een "**App Service**" aan te maken. Men begint met het zoeken in "Alle Services" in het linkse menu. Hier zoek je het volgende.



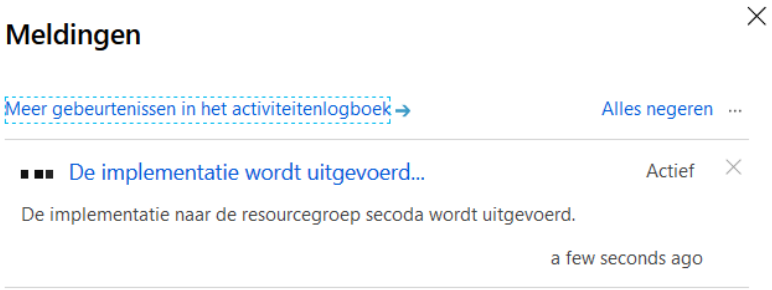
Hier klik je op en daarna kies je voor “Toevoegen” bovenaan en kies je voor “**Web-app**”. Je klikt op “**Maken**” onderaan.



Hier kan je de volgende informatie invullen. De “App-naam” is de naam voor de website waar de gebruikers op het dashboard zullen connecteren. Kies deze dus wijs.



Daarna kan je wachten tot het implementeren klaar is.



Na het ontwikkelen van je Angular 6 applicatie zoals je normaal doet, kan je met volgend commando's de applicatie builden:
"**ng build --prod**"

Je app zou nu onder de "dist folder > JouwAppNaam" moeten zien staan.

Daarna ga je naar de Azure extensie kies je de App naam die je voorheen al maakte in de setup. Dan kies je rechtsboven: "**Deploy Web App**".

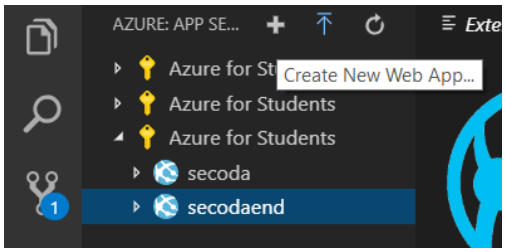
Hier kies je terug de App naam en daarna browse je naar de "dist folder > JouwAppNaam". Klik "yes" op de pop-up en druk op "deploy".

Daarna wacht je tot er een pop-up komt die aanbiedt om de website link te openen.

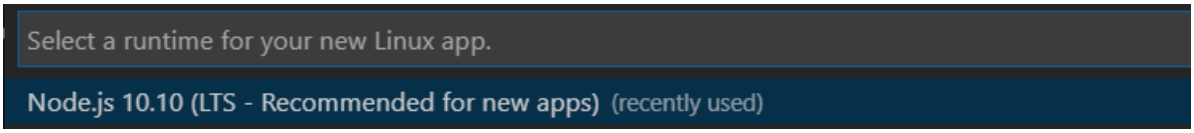
2.3.2 Backend: Nodejs

Je maakt eerst een template Web app voor de Nodejs applicatie. Dit doe je op de volgende manier. Je kiest de correcte Azure subscriptie in de Azure extensie.

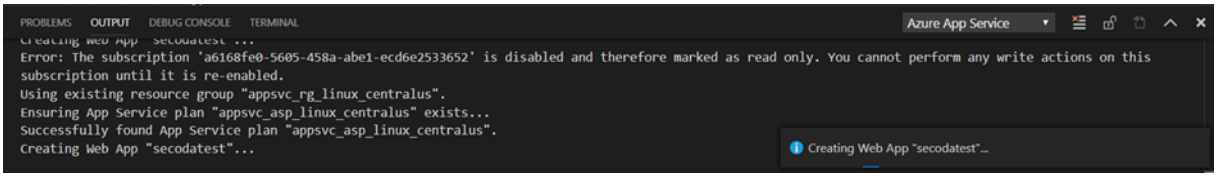
Je klikt op het + symbool om een Web app toe te voegen. Je kiest dan een unieke naam voor deze Web app.



Vervolgens kies je onderstaande Nodejs versie.



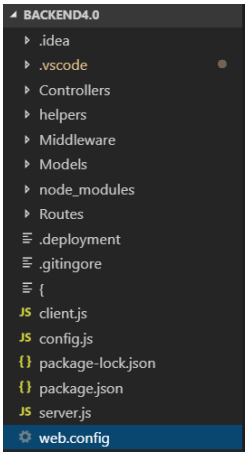
Nu kan je wachten tot de Web app gemaakt is. Je kan de voltooiing volgen in het output venster onderaan VS Code.



De werkelijke hosting wordt hieronder verder uitgelegd.

Alvorens je de Web app gaat hosten met Azure, zal je een tweetal dingen moeten toevoegen/veranderen aan je Nodejs applicatie.

Als eerste dien je een bestand, genaamd web.config toe te voegen aan je applicatie. Dit komt in de root directory te staan.



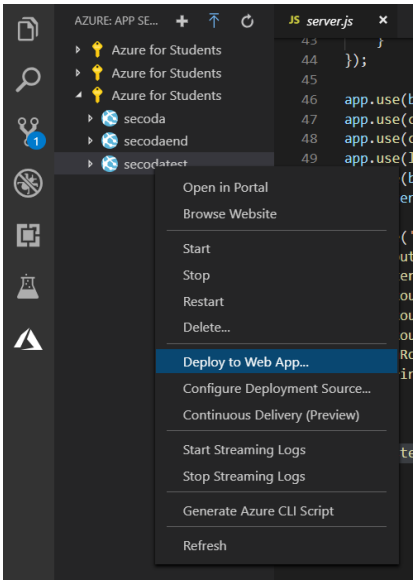
De inhoud van deze file kan je terugvinden in onze Nodejs applicatie.

Het tweede dat je moet doen is het aanpassen van de poort variabele waarop Nodejs draait. Dit is te vinden in de volgende locatie: "**server.js**".

Je dient het volgende aan te passen:

```
app.listen(process.env.PORT || 3000, () => console.log('listening on port 3000!'))
```

Daarna dien je eerst lokaal te kijken of je applicatie goed draait natuurlijk.
Dan kan je beginnen met deployen. Je hoeft gewoon door te klikken en kan dan wachten tot het klaar is.

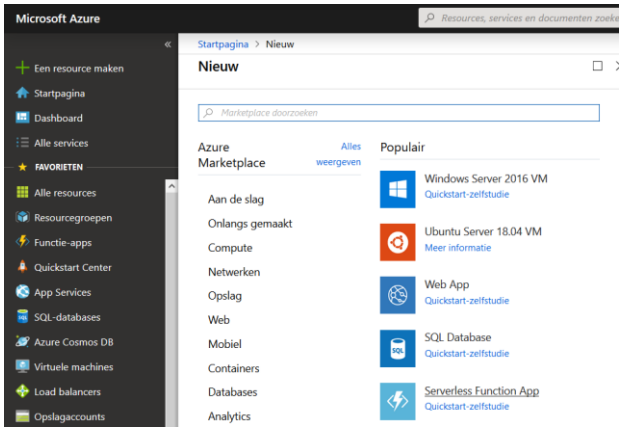


2.4 Functions

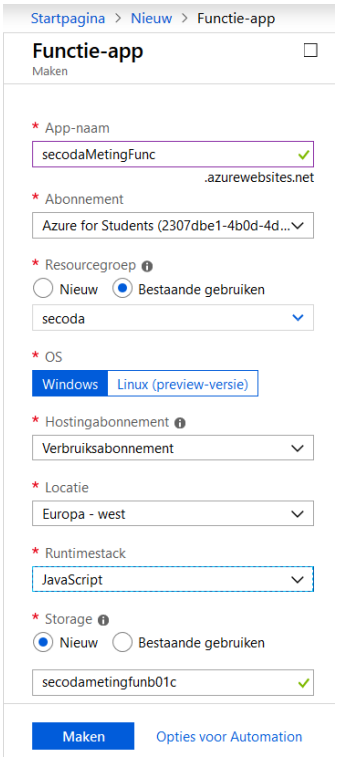
Vervolgens maken we onze functies aan die onze ontvangen metingen zullen verwerken en dan opslagen in Cosmos DB.

Je begint met het aanmaken van twee functies. Eentje voor het verwerken van de metingen. De andere voor het verwerken van de status van de sensoren. We maken er twee aparte voor de overzichtelijk van de structuur. Dit naar de toekomst meer schaalbaar.

Je klikt op "**Een resource maken**". Daarna kies je voor "**Serverless Function App**".



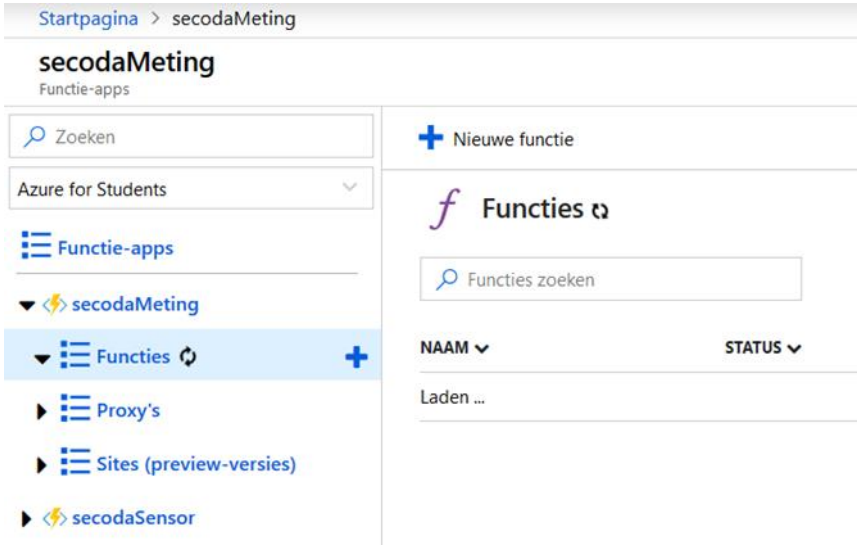
Je kan dan volgende informatie invullen. Kies zeker voor "**JavaScript**" als "**Runtimestack**"!



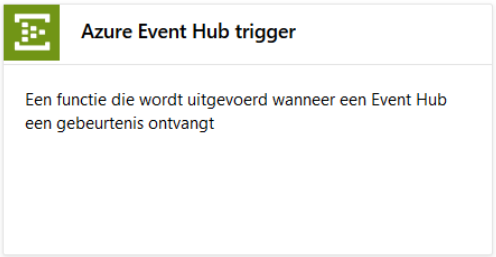
Doe dit voor de tweede functie ook, maar neem dan de naam: "secodaSensorFunc" bijvoorbeeld.

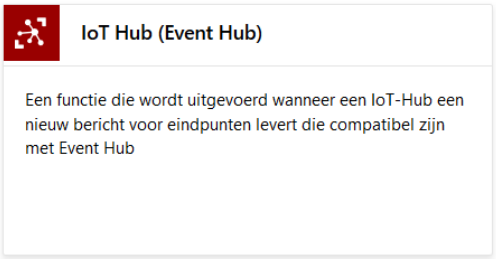
Vervolgens zoek je in "**Alle services**" naar functie-apps. Je zou bij je abonnement de twee gemaakte functies moeten zien.

Klik de eerste open en klik op "**Nieuwe functie**".

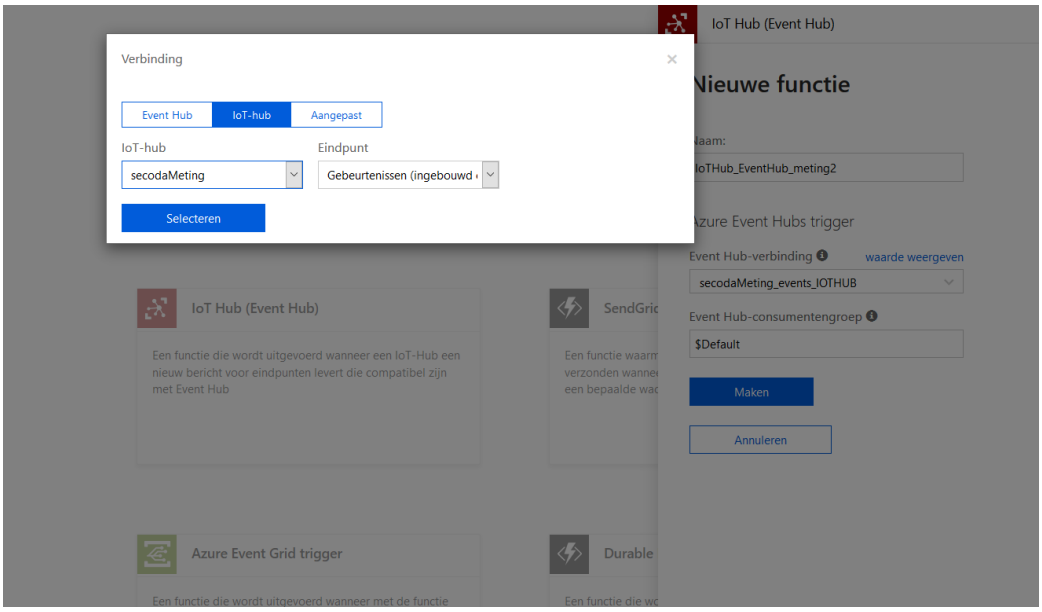


Kies voor een "**IoT Hub (Event Hub)**" trigger.



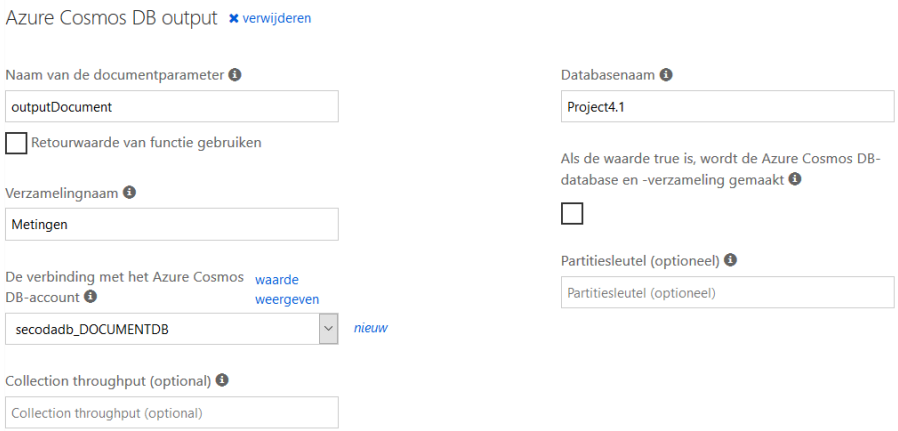


Bij eventuele installatie pop-ups klik je op installeren.
Je geeft het een naam in functie van wat het moet verwerken. Bij "**Event-Hub verbinding**" kies je "**nieuw**" en neem je de corresponderende IoT Hub dat het moet verwerken.



Daarna klik je op "**Maken**". Zodra de functie gemaakt is, klik je op "**Integreren**". Daar klik je rechtsboven op "Uitvoer". Kies dan voor "**Azure Cosmos DB**".

Installeer de bijkomende extensie en vul dan volgende informatie in.



De tweede functie doet hetzelfde, maar ontvangt berichten van de andere IoT Hub. Bij de tweede functie dien je hetzelfde te integreren, maar de data worden opgeslagen in de collectie "**Sensoren**".

Hier moet je wel nog eerst een input bij integreren. Kies daar ook "**Azure Cosmos DB**" en ook de collectie "**Sensoren**". Dit doen we zodat de vorige, eerder aangemaakte informatie zoals namen, niet zal worden overschreven.

De code voor beide functies is in de bijlage te vinden.

3 Applicatie

Onze webapplicatie bestaat uit 2 delen, een front-end en een backend. Hieronder wordt er beschreven wat je moet doen in de backend om deze te laten functioneren.

3.1 Node JS backend

De front end verstuurd API request naar onze backend om data te verkrijgen. Deze data haalt de node JS applicatie uit de databank. De connectie tussen onze backend en de databank van Azure gebeurt via een unieke connectiestring en een URI. Deze 2 zaken dienen in de backend geplaatst te worden in het client.js bestand. Dit bestand kan je terug vinden op het hoogste niveau van de mappenstructuur.

Op lijn 3 van het bestand dien je de URI string in te geven, vervolgens geef je op lijn 4 de connectie string in die je ontvangen hebt na het aanmaken van de databank op Azure. Nu is de connectie tussen de databank en de backend compleet.

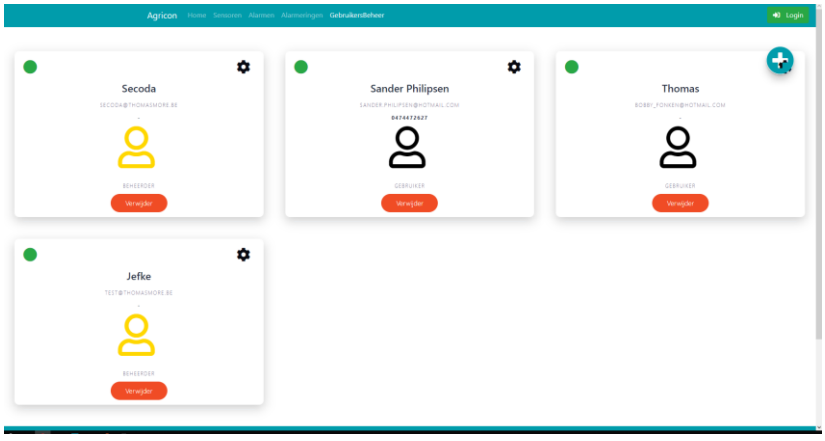
Client.js:

```
1  const CosmosClient = require("@azure/cosmos").CosmosClient;
2
3
4  const URI = "https://secodadb.documents.azure.com:443/";
5  const KEY = "LrVIen8qOYB6n31m3s2dR8BjgdsPo6RRLbKIjZWVDfgC7gyzQXEzSGBWDjBmAzRILXE2z1v7HltUUZg3CGm2pA==";
6
7  const dbClient = new CosmosClient({
8      endpoint: URI,
9      auth: {
10         masterKey: KEY
11     },
12     consistencyLevel: 'Session'
13 });
14
15 module.exports = dbClient;
16
```

3.2 Gebruikers

In onze applicatie zijn er 2 soorten gebruikers, administrators en gewone gebruikers. Gewone gebruikers kunnen enkel het dashboard raadplegen. Administrators hebben de mogelijkheid om sensoren, alarmen en gebruikers te beheren.

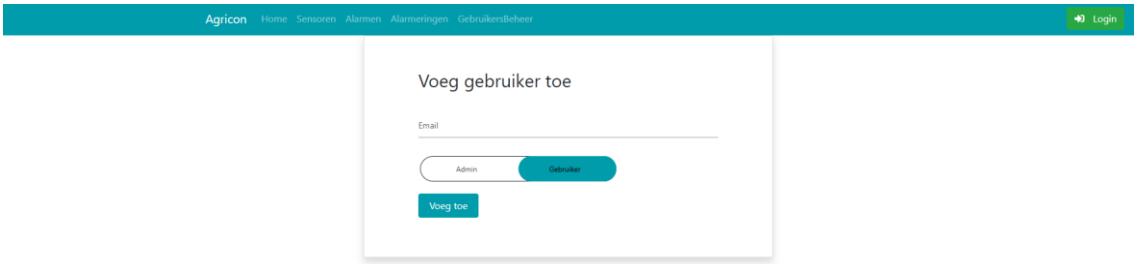
De pagina om gebruikers te beheren.



3.2.1 Nieuwe gebruiker

Een administrator heeft de mogelijkheid om een nieuwe gebruiker toe te voegen. Dit doet hij door bovenaan rechts op het plus teken te klikken.

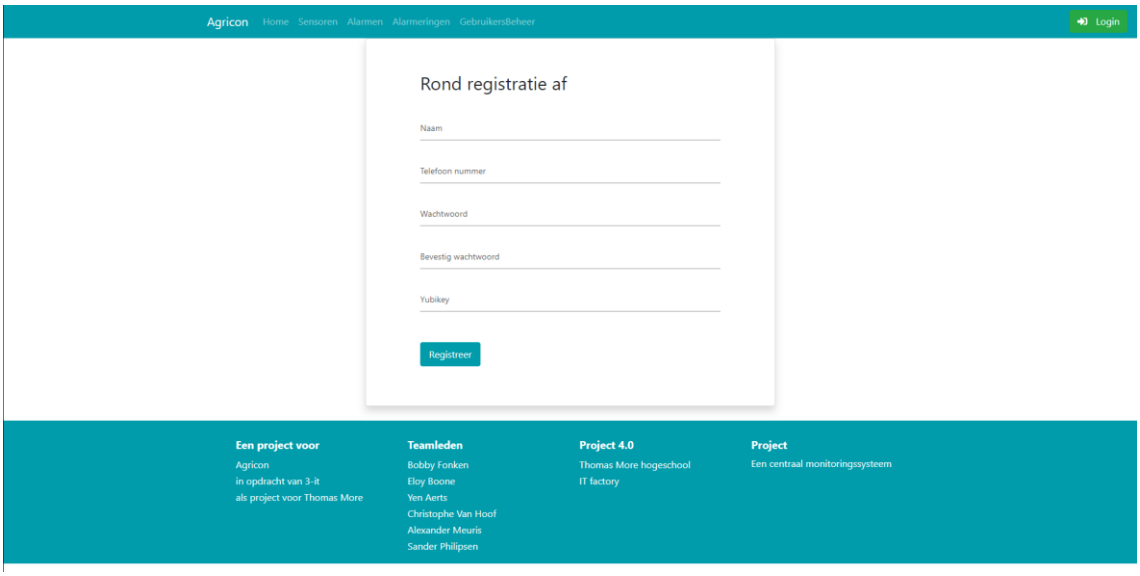
Vervolgens komt hij op een nieuwe pagina waar hij een email adres kan ingeven. Dan heeft hij ook nog de mogelijkheid om de nieuwe gebruiker admin rechten te geven.



Er wordt een email gestuurd naar dit email adres met een link in naar een pagina om zijn verdere gegevens door te geven.

Op deze pagina zal de gebruiker terecht komen indien hij op de ontvangen link klikt.

Als de gebruiker deze pagina correct heeft ingevuld is de gebruiker geactiveerd. Vanaf nu kan deze gebruiker inloggen en de applicatie gebruiker naar gelang de rol die hij heeft.



3.2.2 Gebruikers beheren

De administrator heeft altijd de mogelijkheid om de gegevens van een bepaalde gebruiker te wijzigen of een gebruiker te verwijderen.

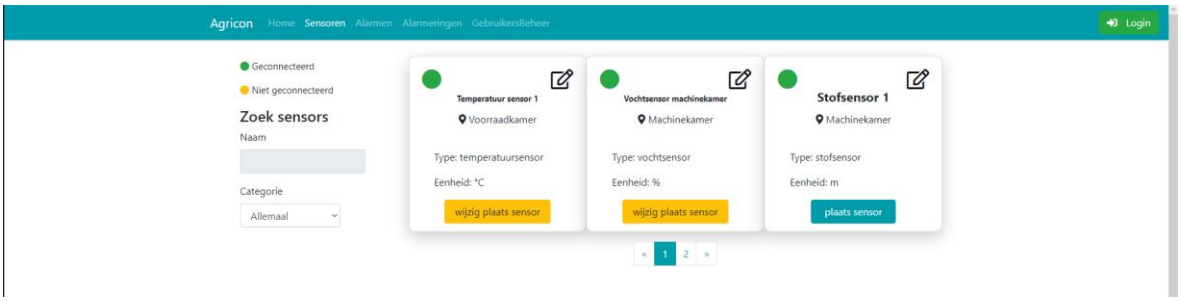
3.3 Sensoren

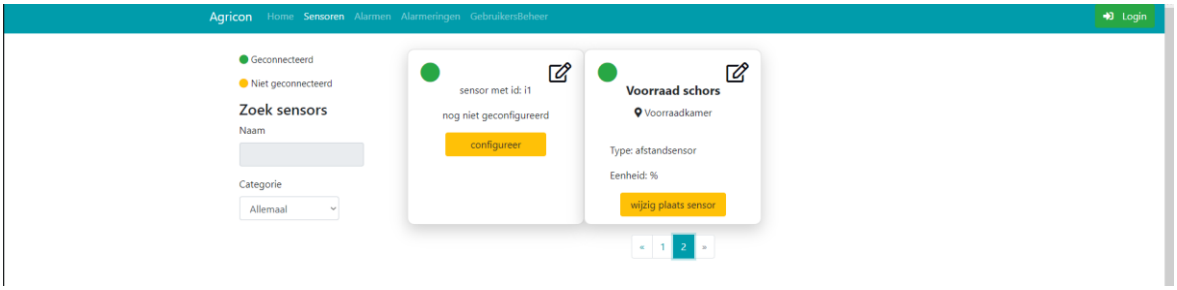
De administrator heeft de mogelijkheid om de sensoren te configureren via de webapplicatie. De sensoren krijgen in de databank een status, ontvangen van de sensoren in de opstelling. Actief of niet actief. Bij elke sensor zie je links bovenaan een oranje of groene bol. Als de sensor geactiveerd en geconnecteerd is zal de bol groen zijn. In elk ander geval oranje. Dan weet je dat er iets mis is met deze sensor. Bij elke sensor zie je rechts bovenaan een potlood. Hiermee kan je de gegevens van de sensor wijzigen. Onderaan kunnen er 3 knoppen staan.

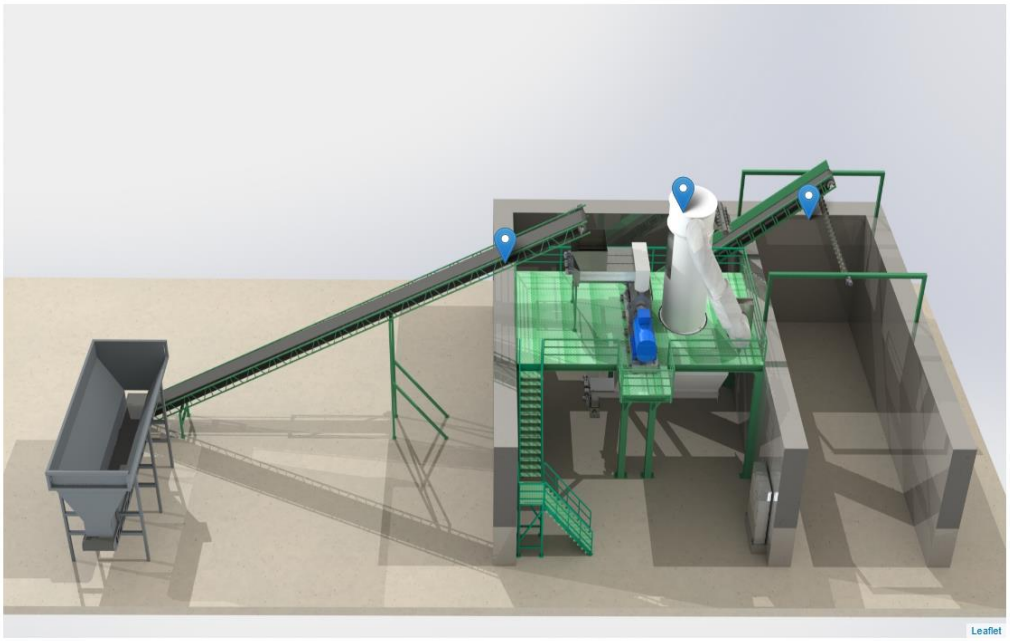
1) configureer: Dit wil zeggen dat de sensor nog niet geconfigureerd is. De sensor heeft nog geen naam, type, eenheid en plaats. Indien een sensor nog niet geconfigureerd is. Kan er geen data zichtbaar zijn voor die sensor in het dashboard.

2) plaats sensor: Dit wil zeggen dat de sensor geconfigureerd is maar nog geen plaats op de 3d tekening van de applicatie heeft. Als je op deze knop klikt heb je de mogelijkheid om een plaats aan te duiden op de 3d foto onderaan van de pagina. Het is de bedoeling dat de administrator aanduidt waar de sensor geplaatst is op of rond de machine.

3) wijzig plaats sensor: Als je deze knop ziet is eigenlijk alles geconfigureerd. Je hebt dan nog de mogelijkheid om de plaats van de sensor op de foto te wijzigen.



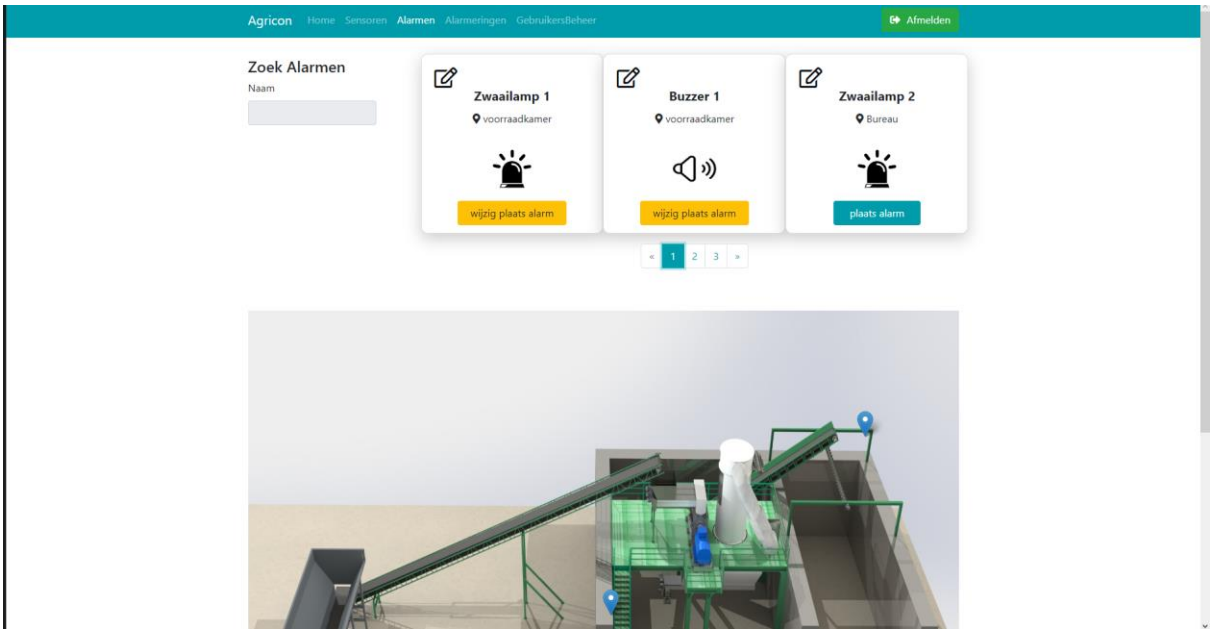


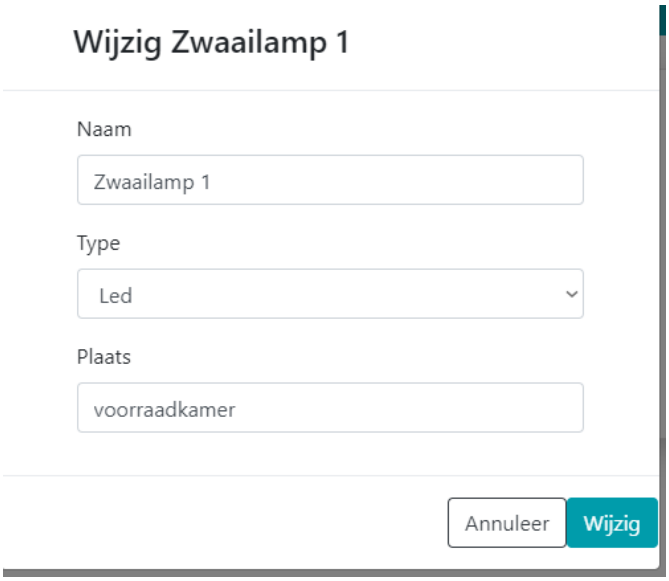


3.4 Alarmen

De administrator heeft de mogelijkheid om de alarmen te configureren. Je ziet een overzicht van de alarmen met enkele filters. Rechts bovenaan heeft de administrator de mogelijkheid om de gegevens van het alarm te wijzigen of door te geven. De administrator kan de naam, het type en de plaats doorgeven.

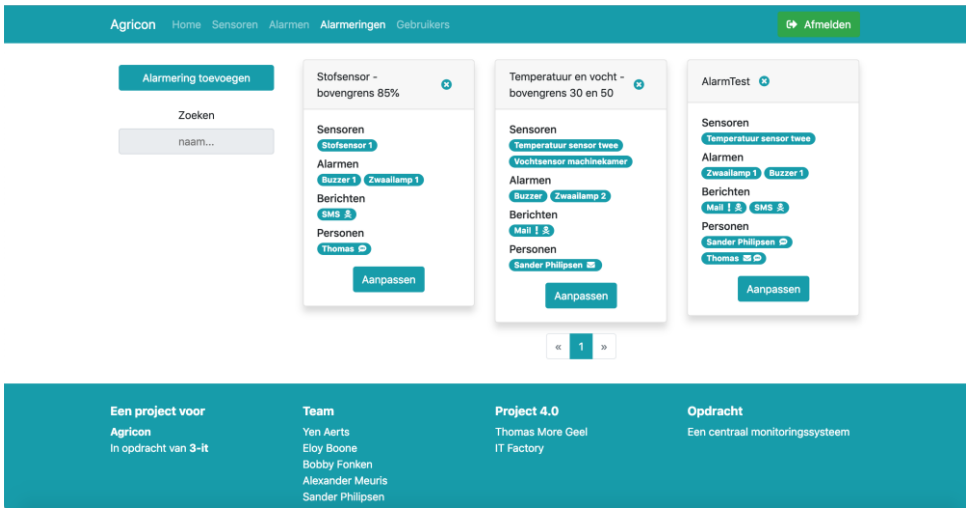
Onderaan van het kaartje zie je een knop. Dit werkt hetzelfde als bij de sensoren. Men kan op de kaart onderaan de pagina via deze knop het alarm plaatsen op de map.





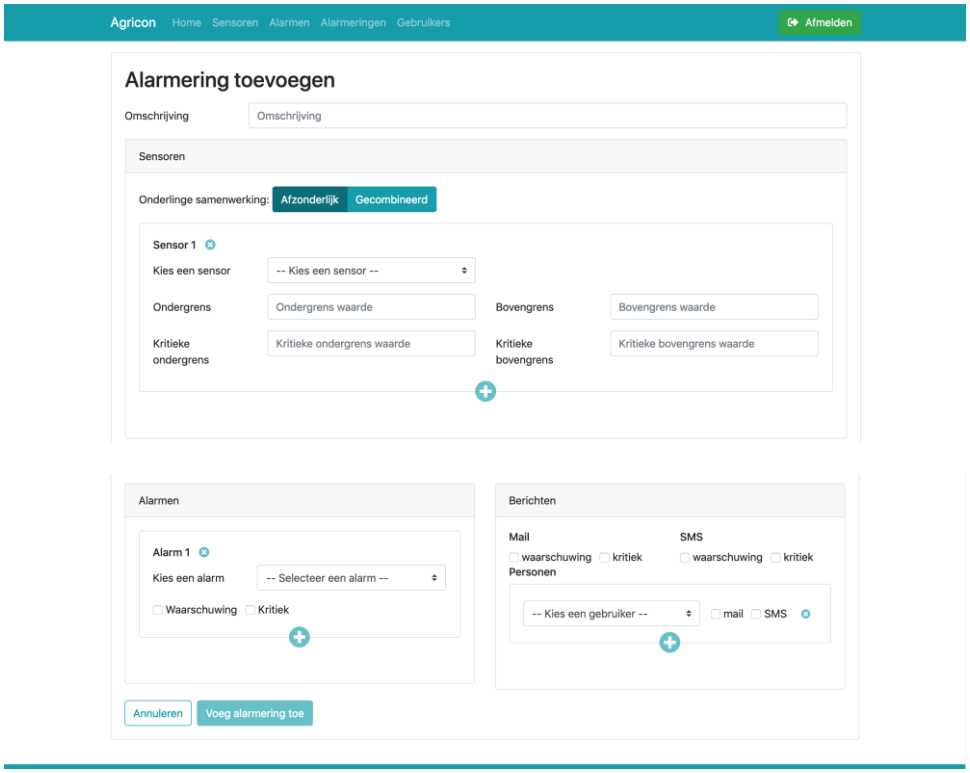
3.5 Alarmeringen

Als laatste heeft de administrator ook de mogelijkheid om alarmeringen te configureren. Net zoals bij de sensoren en alarmen zie je rechts een overzicht van de reeds geconfigureerde alarmeringen.
Per alarmering kan je zien welke sensoren en alarmen geconfigureerd zijn, welke berichten (mail of SMS) verstuurd zullen worden en naar wie. Links vind je een knop om een extra alarmering toe te voegen, met daaronder een zoekveld om te zoeken op naam in de alarmeringen.



1) Alarmering toevoegen

Een alarmering heeft 3 grote onderdelen: sensoren, alarmen en berichten. Er kunnen één of meerdere sensoren toegevoegd worden. Je kiest een sensor en vult daarna de grenswaarden in.
Vervolgens kan je indien gewenst een extra sensor toevoegen. Een alarm toevoegen verloopt hetzelfde, alleen hoef je hier geen grenswaardes in te vullen. Je kiest of je het alarm enkel bij een waarschuwing, een kritieke alarmering of beide wilt gebruiken.
Als laatste kan je nog berichten versturen. Net zoals bij een fysiek alarm kies je een waarschuwing, kritieke waarde of beide. Daarna kies je één of meerdere gebruikers en geeft aan of je hen een mail, sms of beide wil sturen.



2) Alarmering aanpassen

Een alarmering aanpassen werkt volledig analoog als een alarmering toevoegen, met enige verschil dat er natuurlijk al waardes ingevuld zijn.

3.6 Installatie

Je kan de beide applicaties in de bijlage vinden. Of je kan ze via GitHub binnenhalen via volgende linken.

Frontend: https://github.com/EloyB/Frontend4.0.git
Backend: https://github.com/EloyB/Backend4.0.git

Om al de nodige modules te installeren voer je het commando "npm install --save" uit ter hoogte van de projectmap. Dit is zowel voor de front-end en de backend noodzakelijk.

4 Gateway

De gateway is bij ons gesimuleerd door een Raspberry Pi 3 model b.

4.1 Raspbian installeren

Begin eerst met het correct formatteren van een SD kaart. Wij gebruikten hiervoor "SD card formatter".
Dit zal op een veilige en betere manier de SD kaart formateren.

Vervolgens dien je naar https://www.raspberrypi.org/downloads/raspbian/ te gaan. Hier kies je voor "**Raspbian Stretch with desktop and recommended software**".

Deze dien je vervolgens met Rufus op de SD kaart te plaatsen.

Nadat dit gebeurd is heb je een keuze.
Of terwijl ga je met een monitor, muis en toetsenbord de Pi instellen of (dit doen wij) stel je de Pi via SSH in.

Om bij eerste boot meteen met ssh te kunnen inloggen, moeten we eerst in de boot directory een bestand aanmaken. Dit bestand moet simpelweg "ssh" noemen. Dit mag leeg zijn en zonder extensie zijn.

Daarna kan je de SD kaart in de Raspberry Pi 3 steken en opstarten. Met Wireshark kan je luisteren naar de interface op je laptop en kan je het IP-adres achterhalen.

4.2 Voorbereiden met nodige software

4.2.1 Users en updates

De default login voor de Pi is:
Username: pi
Password: raspberry

Het eerste dat we doen bij aanmelden is het huidige wachtwoord veranderen. Doe dit door gebruik te maken van het commando "**passwd**".

Daarna gaan onze eigen gebruiker aanmaken. Doe dit als volgt:
sudo adduser username
sudo usermod -aG sudo username

Nu meld je je aan met de nieuwe sudo user en voeren we eerst een update en upgrade uit.
sudo apt-get update
sudo apt-get upgrade

Hierna herstarten we de Pi om de veranderingen door te zetten.
sudo shutdown -r now

Daarna kan je nazien of de tijd goed is ingesteld.
"**date**" laat de huidige klok zien. Als deze verkeerd staat kan je met het volgende commando deze aanpassen als je optie 4 kiest.

sudo raspi-config

4.2.2 Nodige software

Vervolgens gaan we al de nodige software erop installeren dat we Python project gebruikt. Het merendeel zit al standaard in Python.

Onderdelen:

- Wiring Pi
- Wi-Fi hotspot: RaspAP
- Azure Python SDK
- Python Twilio
- Python gnupg

Wiring Pi:
sudo apt-get install python-dev
sudo apt-get install python-rpi.gpio

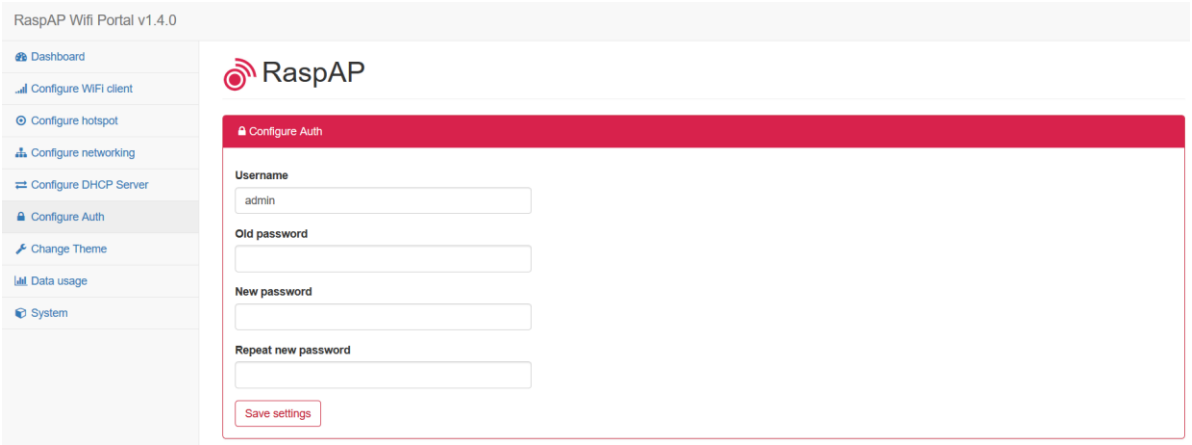
RaspAP:
sudo wget -q https://git.io/voEUQ -O /tmp/raspap && bash /tmp/raspap

Antwoord op alle vragen "y".

Nadat de installatie afgerond is, kan je surfen naar het IP address van de Pi en kan je inloggen met onderstaande gegevens.

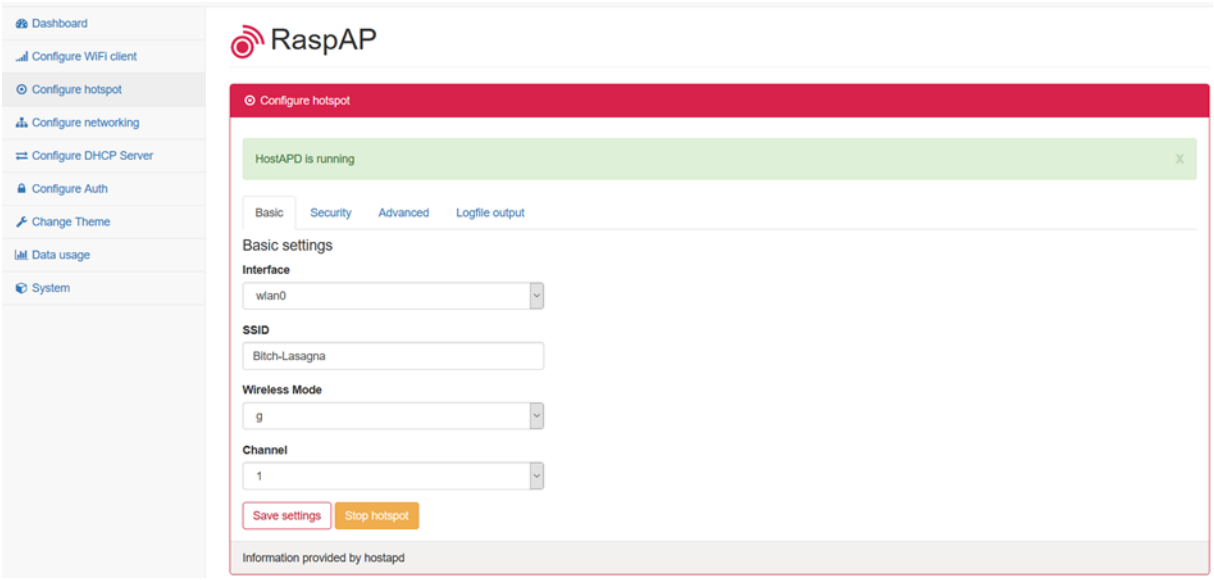
Default login admin webpage:
IP address: 10.3.141.1
Username: admin
Password: secret

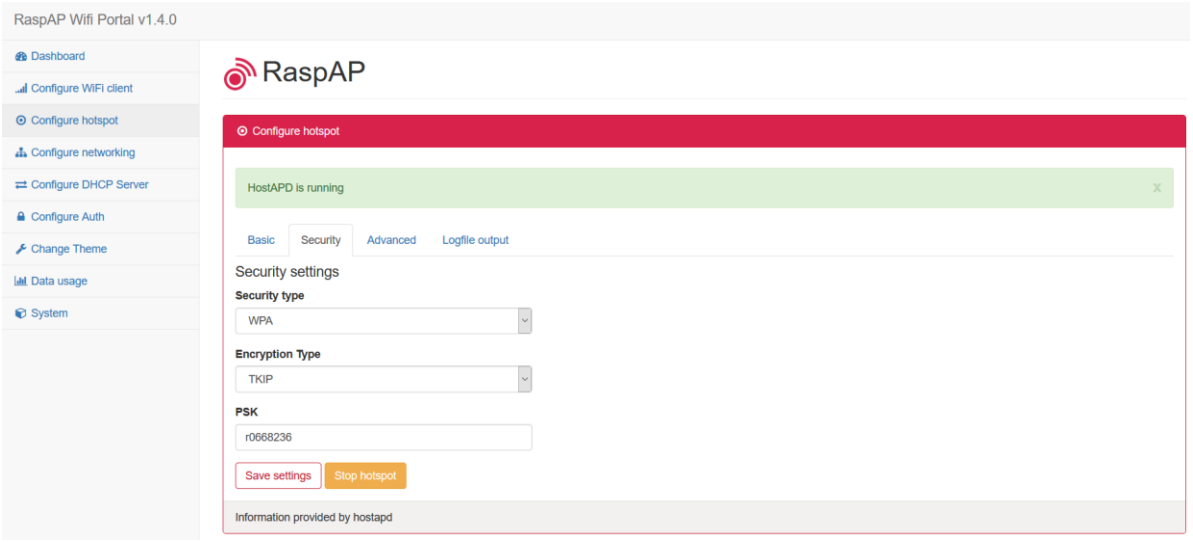
Eenmaal aangemeld, kan je de default admin login veranderen.



Onderstaande login is om op het netwerk van de hotspot te geraken.
Default login network:
SSID: raspi-webgui
Password: ChangeMe

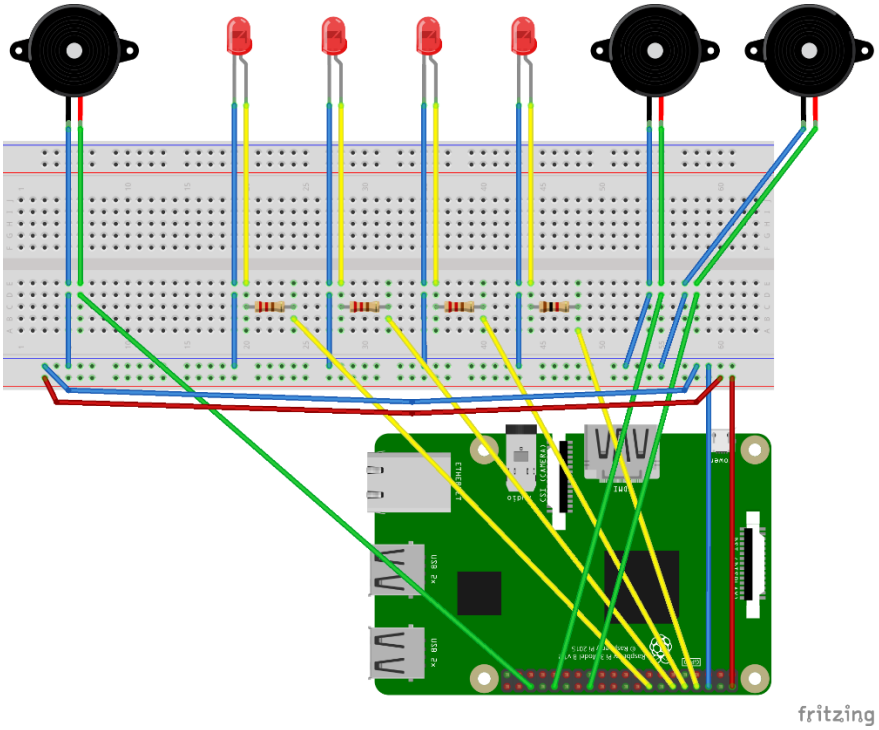
Deze moet je natuurlijk veranderen en kan je op de volgende pagina aanpassen.



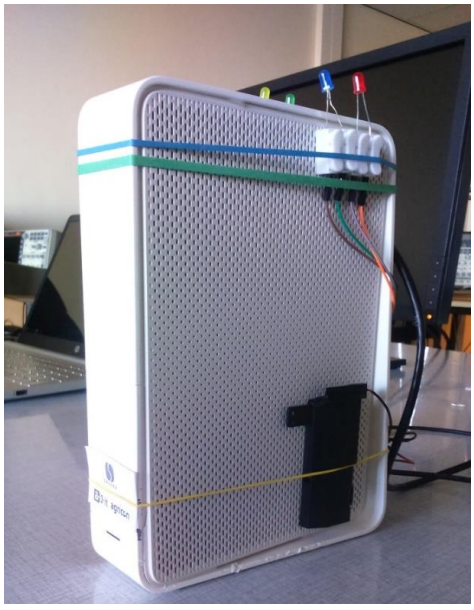


4.2.3 Bedradingen schema

Onderstaande schema is de bedrading van de Raspberry Pi. Hier hangen de ledjes en buzzers aan die gebruikt worden voor de alarmering.
De weerstanden zijn van (1kOhm).



In onze opstelling zitten er nog tussen de componenten en de behuizing verlengstukken zodat je in principe de component daar aanhangt. Een beetje zoals een stopcontact.
Onderstaande foto illustreert dit, het zijn de witte stukken waar de ledjes aanhangen.



Op de gateway staat bij elk contactpunt een id. Deze heb je voordien gebruikt in de database.

4.2.4 Python scripts

Voor de nodige scripts te krijgen kan je met het volgende commando deze binnenhalen van GitHub of kijken in de bijlage.
https://github.com/bobbyfonken/Python-Azure-sensor-communication.git

Daarna moet je nog twee mappen bij aanmaken.
sudo mkdir temp
sudo mkdir azure

Hierna dien je volgende structuur te gebruiken in een nieuw aangemaakt "config.json" bestand waar alle wachtwoorden etc. in voorkomen.

```
[{
"localIP": "RaspAP IP gateway",
"Port1": Poort nummer waar sensor hub naar toe zend,
"CONNECTION_Meting": "Connection string voor metingen te sturen",
"CONNECTION_Sensor": "Connection string voor sensor status te sturen",
"CONNECTION_C2D": "Connection string voor berichten te ontvangen",
"EMAIL_ADDRESS": "email",
"PASSWORD": "email wachtwoord",
"account_sid": "Twilio account_sid",
"auth_token": "Twilio auth_token",
"TwilioNumber": "Twilio nummer",
"alertsJson": "Wachtwoord gebruikt voor het de/encrypteren: azure/alerts.json.gpg",
"usersJson": "Wachtwoord gebruikt voor het de/encrypteren: azure/users.json.gpg "
}]
```

Na deze waarden te hebben aangepast naar uw waarden, verplaats je je naar de "**config/**" directory.

Hier zie je een script "**encrypt.py**". Dit zal op basis van een gegeven wachtwoord zin het "**config.json**" bestand encrypteren. Zo zal alles veilig gebeuren en zijn er geen plain tekst wachtwoorden te vinden.

Vergeet zeker niet het originele "**config.json**" bestand te **verwijderen**!

Als dit allemaal gebeurd is, kan je met volgende commando's elk in een aparte terminal de volgende twee bestanden uitvoeren.
sudo python meting.py
sudo python receive.py

Voor beide scripts geef je de wachtwoord zin die je hebt gebruikt om het te encrypteren met "**encrypt.py**".

Het eerste script zal metingen ontvangen van de sensor hub en doorsturen naar Azure IoT Hub om opgeslagen te worden in Cosmos DB.
Het tweede zal data ontvangen van Azure IoT Hub als er in de collectie "Gebruikers" of "Alarmeringen" wijzigingen voorkomen.
Deze worden dan opgeslagen in de "azure/" directory en zullen dan worden gebruikt door "meting.py"
NOTE: Lees het scriptcommentaar goed! Begin bij "**meting.py**".

5 Versies

Hieronder kan men de versies van onze gebruikte software terugvinden.

5.1 Sensor hub

5.1.1 Technologie

Arduino IDE
Versie: 1.8.8

Flash download tool
Versie: 3.6.5

5.1.2 Software

AT firmware Espressif
Versie: 1.6.2

5.2 Gateway

5.2.1 Technologie

SD card formatter
Versie: 5.0.1

Linux raspberrypi 4.14.98-v7+ #1200 SMP Tue Feb 12 20:27:48 GMT 2019 armv7l GNU/Linux
Raspbian Stretch with desktop and recommended software

```
PRETTY_NAME="Raspbian GNU/Linux 9 (stretch)"
NAME="Raspbian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
```

Rufus
Versie: 3.3

5.2.2 Software

Python
Versie 2: 2.7.13
Versie 3: 3.5.3

Wiring Pi
Versie python-dev: 2.7.13-2
Versie python-rpi.gpio: 0.6.5~stretch-1

RaspAP
Versie:/

Azure Python SDK
Versie: azure-iothub-device-client-1.4.6

Python Twilio
Versie: Twilio-6.24.1 pytz-2018.9

Python gnupg
Versie: python-gnupg-0.4.4

5.3 VS Code

Azure App Service
Versie: 0.12.0

ChartJS
Versie: 2.7.3

5.4 Frontend

Bootstrap
Versie: 4.2.1

chart.js
Versie: 2.7.3

Angular
Versie: 7.1.0

Fontawesome
Versie: 5.6.3

Leaflet
Versie: 0.0.16

5.5 Backend

yubikey
Versie: 0.0.2

azure-iothub
Versie: 1.9.3

twilio
Versie: 3.27.1

express
Versie: 4.16.4

jsonwebtoken
Versie: 8.4.0

nodemailer
Versie: 5.1.1

6 Referenties

Hieronder kan men de referenties naar onze uitgewerkte onderdelen terugvinden.

6.1 Sensor hub

6.1.1 Technologie

Infrarood sensor
https://www.kiwi-electronics.nl/grove-ir-distance-interrupter?gclid=Cj0KCQiA5NPjBRDDARIsAM9X1GK9umvFl80dbf2qCvakEgPqII4NHqXOJiBi52SDwP_1ruVkrJchbokaAvYsEALw_wcB

Infrarood sensor
https://www.antratek.be/optical-dust-sensor-gp2y1010au0f

Arduino ide
https://www.arduino.cc/en/main/software

Flash downloader tool
https://www.espressif.com/en/support/download/other-tools

6.1.2 Software

DHT sensor library
https://github.com/adafruit/DHT-sensor-library

Infrarood sensor
https://www.antratek.be/optical-dust-sensor-gp2y1010au0f

AT firmware
https://www.espressif.com/en/support/download/at

6.2 Gateway

6.2.1 Technologie

SD card formatter
https://www.sdcard.org/downloads/formatter_4/

Raspbian Stretch with desktop and recommended software
https://www.raspberrypi.org/downloads/raspbian/

Rufus
https://rufus.ie/

RaspAP
https://howtoraspberrypi.com/create-a-wi-fi-hotspot-in-less-than-10-minutes-with-pi-raspberry/
https://github.com/billz/raspap-webgui

gnupg
https://pythonhosted.org/gnupg/gnupg.html

Python send data to Azure
https://docs.microsoft.com/en-us/azure/iot-hub/quickstart-send-telemetry-python

Azure IoT Hub Python SDK
https://github.com/Azure/azure-iot-sdk-python

6.3 Backend

server.js/web.config
https://developers.de/2018/05/26/deploying-a-node-js-express-service-app-to-azure-app-service-using-visual-studio-code/

6.4 App services

Deploy Angular/nodejs
https://www.webcodegeeks.com/javascript/angular-js/deploy-angular-app-visual-studio-azure/