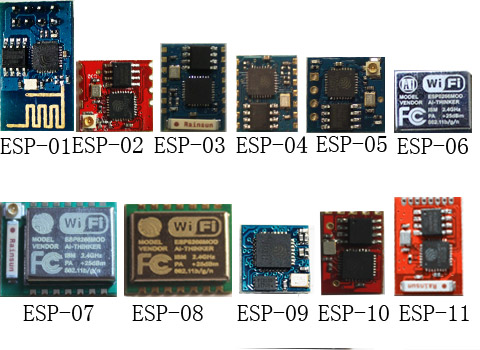
**Sensor Netwerken**



**Naam:** Wesley de Hek

**Klas:** 23TiVtIOT1

**Github:** https://github.com/xrgman/Ti3.3-SensorNetwerken

**Inhoudsopgave**

[Inleiding 3](#_Toc475108907)

[Les Week 1 4](#_Toc475108908)

[Opdracht 1: Seriële communicatie 4](#_Toc475108909)

[Opdracht 2: Analoge sensor waarden uitlezen 5](#_Toc475108910)

[Opdracht 3: HUE lampen aansturen met de ESP8266 7](#_Toc475108911)

[Les week 2 9](#_Toc475108912)

[Opdracht 1: Installatie en werkend maken van Node.js 9](#_Toc475108913)

[Opdracht 2: Eenvoudige database 9](#_Toc475108914)

[Opdracht 3: Maak een RESTful node applicatie 10](#_Toc475108915)

[Lesweek 3 12](#_Toc475108916)

[Opdracht 1a: Communicatie tussen router en coördinator 12](#_Toc475108917)

[Opdracht 1b: Communicatie tussen coördinator en meerdere router nodes 12](#_Toc475108918)

[Opdracht 1C: Communicatie met de Arduino 13](#_Toc475108919)

[Opdracht 2a: Informatie naar het internet 13](#_Toc475108920)

[Opdracht 2b: Temperatuurmetingen naar de database 15](#_Toc475108921)

# **Inleiding**

Voor het vak sensor netwerken, gaan we tijdens de derde periode van het derde leerjaar aan de slag met een ESP8266 microchip. Dit is een zeer krachtige microcontroller, die ook nog een beschikt over een ingebouwde wifiradio. Tijdens de lesweken zullen er elke week een aantal opdrachten gemaakt worden, om op deze manier meer te leren over sensoren en de ESP8266. Deze opdrachten zullen in dit document per week vastgelegd worden.

# **Les Week 1**

## **Opdracht 1: Seriële communicatie**

Bij deze opdracht moest er een seriële afhandeling gemaakt worden op de Esp8266, die zo werkte dat hij de achterliggende code niet zal storen zodra er een commando binnen zou komen. Het uitlezen van een serieel commando werd op de volgende manier afgehandeld:

void loop() {

//Polling the serial commands (Because there is no interrupt :( )

serialEvent();

}

void serialEvent() {

if(Serial.available()) {

char ch = Serial.read();

if(ch == '\n' || ch == '\r') {

cmd[cmdIndex] = '\0';

cmdIndex = 0;

processSerialCommand(cmd);

}

else {

cmd[cmdIndex] = ch;

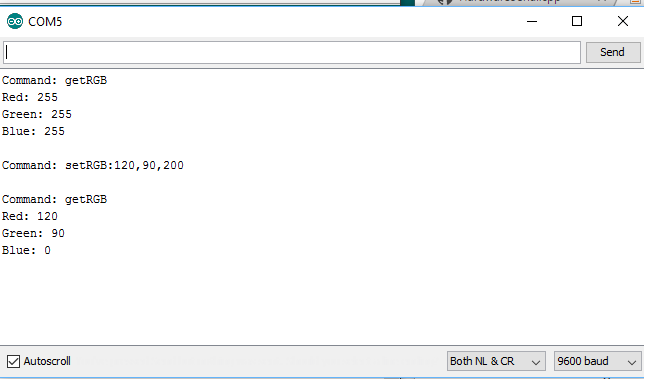
cmdIndex++;

}

}

}

Vervolgens werden er nog een paar commando’s geprogrammeerd om het ook daadwerkelijk te testen:



Afbeelding 1 - Serial Commands

## **Opdracht 2: Analoge sensor waarden uitlezen**

Bij deze opdracht moest er via de esp8266 een aantal analoge waardes van een analoge sensor uitgelezen worden. Deze waarden zouden vervolgens naar de console gelogd worden. Dit is gedaan via het onderstaande stukje code en schakeling:

#define analogSensorPin 0

int value;

void setup() {

pinMode(analogSensorPin,OUTPUT);

value = 0;

Serial.begin(9600);

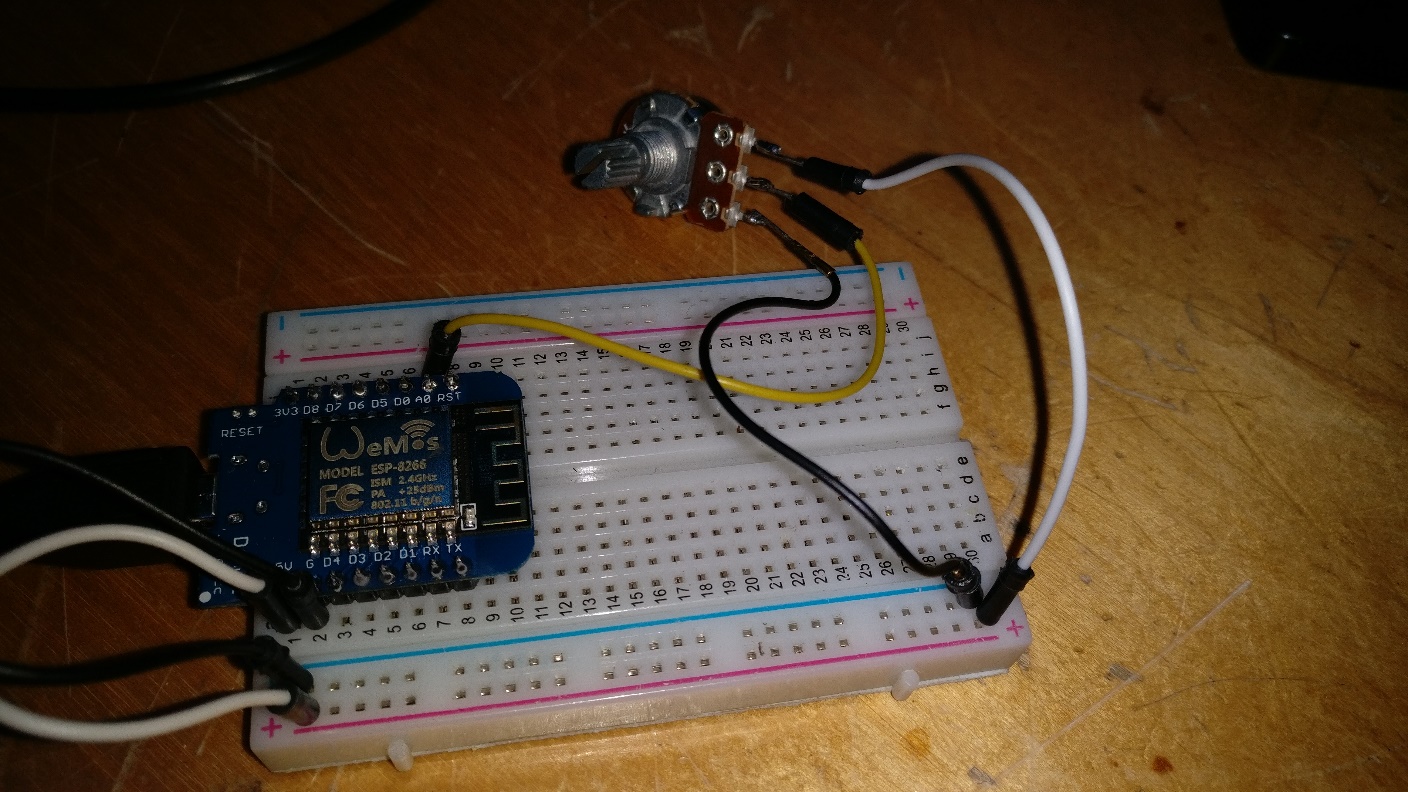
}

void loop() {

value = analogRead(analogSensorPin);

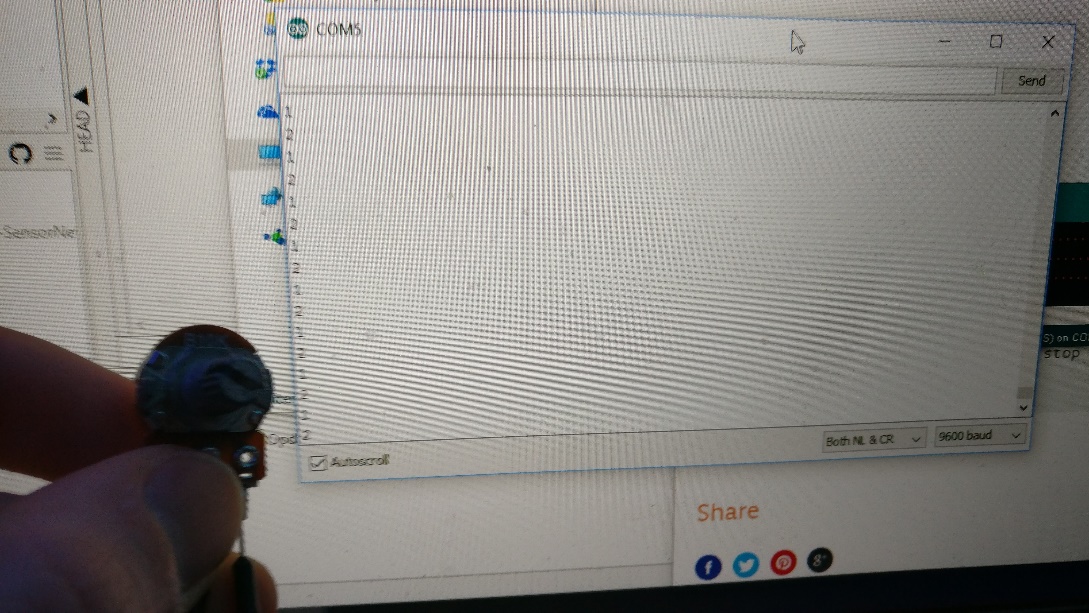
Serial.println(value);

}

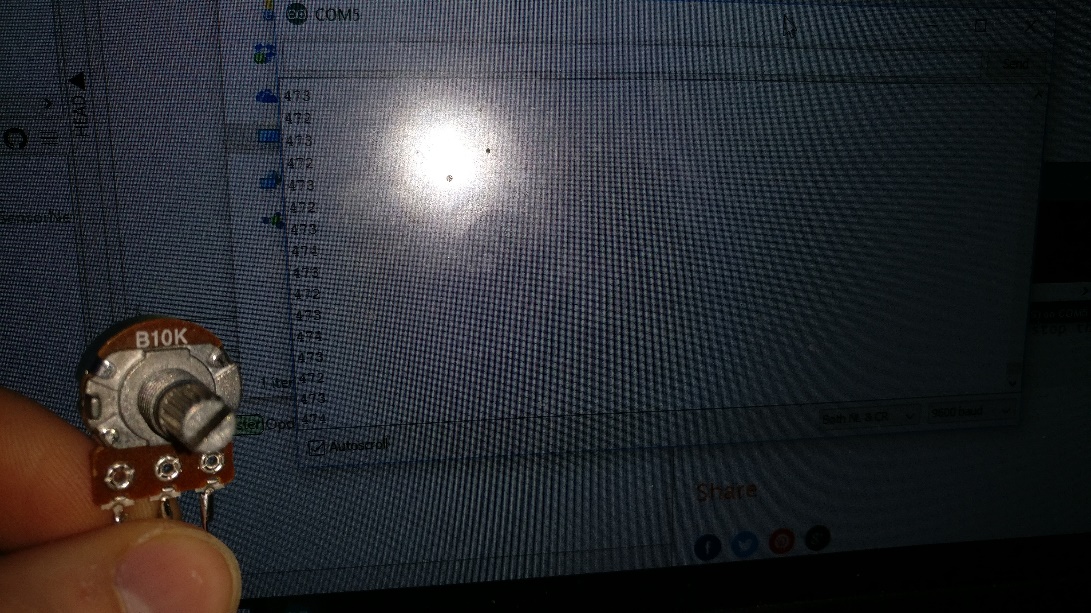


Afbeelding 2 - Uitlezen analoge potmeter

Hier onder nog de resultaten van de waardes van de potmeter die uitgelezen werden:



Afbeelding 3 - Potentiometer op de laagste stand



Afbeelding 4 - Potentiometer op 70%

## **Opdracht 3: HUE lampen aansturen met de ESP8266**

Bij deze opdracht moesten we een HUE lamp aansturen via de ESP8266, hierbij is er een toggle knop toegevoegd waarmee de lamp aan en uitgezet kan worden. Dit is gedaan met behulp van de volgende code:

int newState = digitalRead(switchPin);

if(newState != state) {

if(newState) {

turnLightOn(1);

}

else {

turnLightOff(1);

}

state= newState;

}

Hierbij zagen de methodes turnLightOn en off er zo uit:

void turnLightOn(int lightID) {

HTTPClient http;

String request = "http://192.168.1.132:8000/api/1502a28be4803a275231925ee1b9a4e/lights/";

request += lightID;

request += "/state";

http.begin(request); //HTTP

int httpCode = http.sendRequest("PUT","{ 'on': true }");

if(httpCode == HTTP\_CODE\_OK) {

String payload = http.getString();

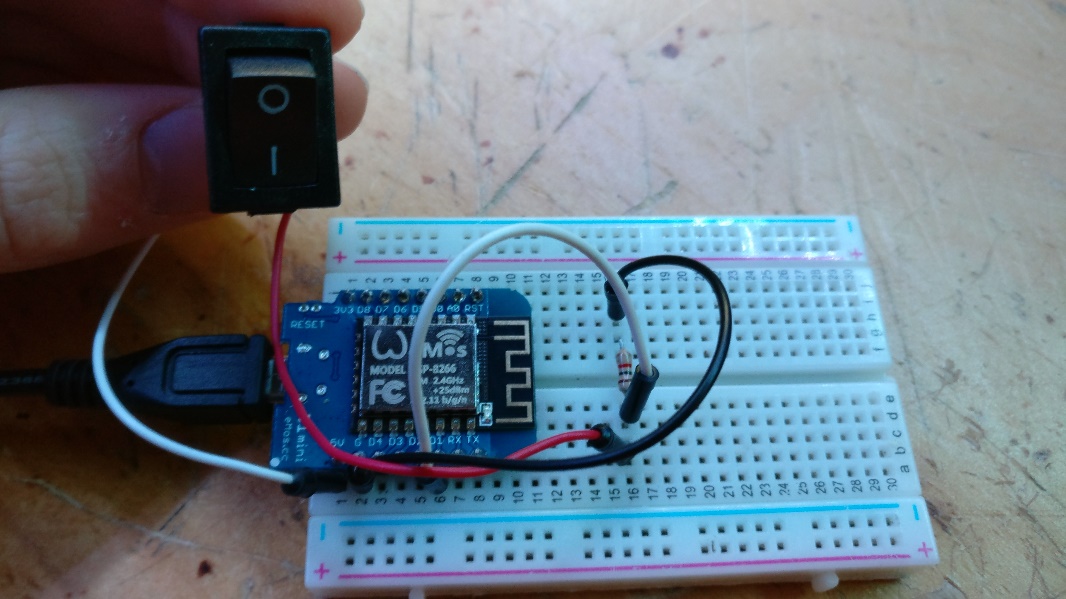
Serial.println(payload);

}

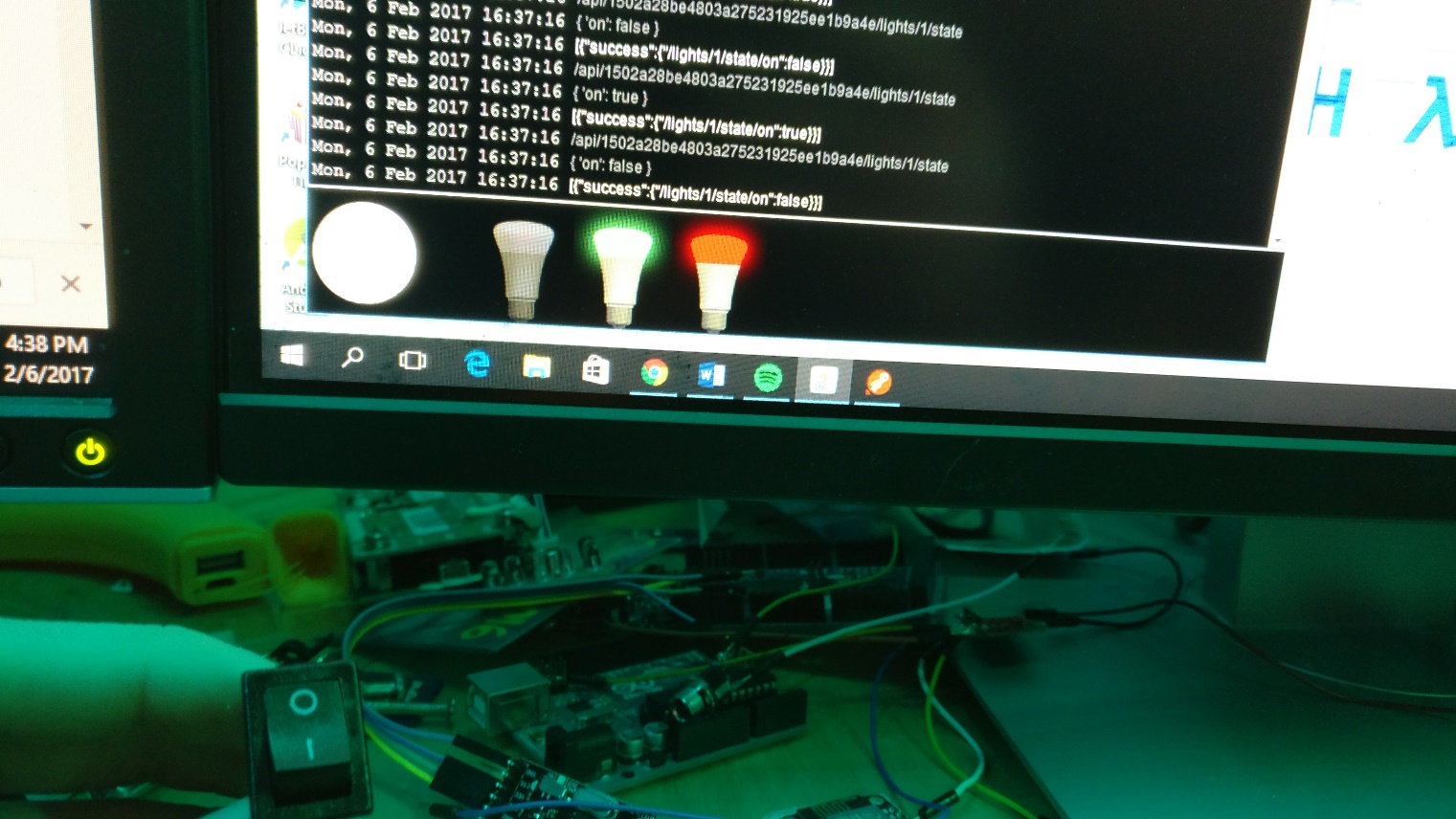
http.end();

}

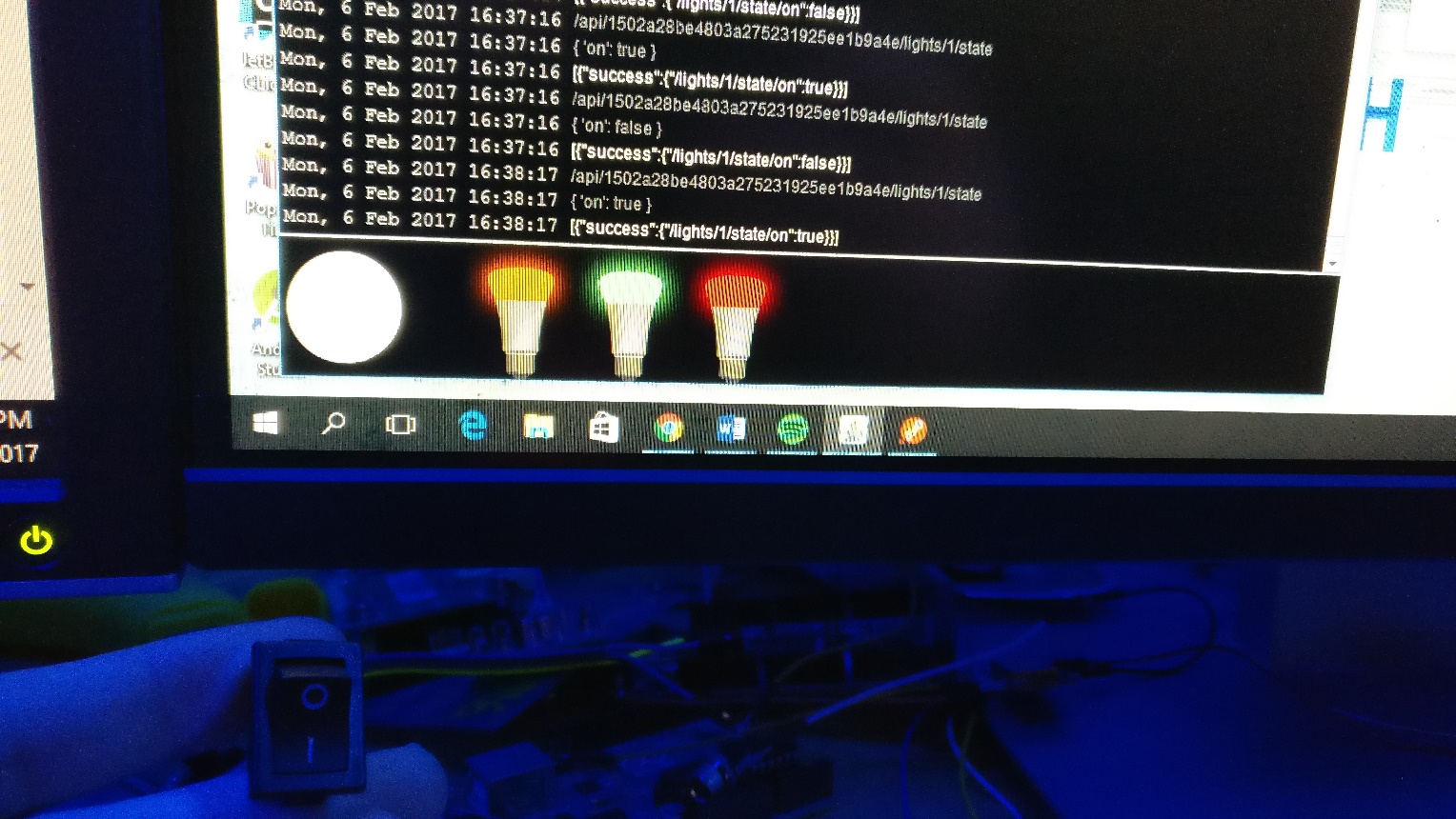
Deze code is vervolgens uitgevoerd met de onderstaande opstelling:



Afbeelding 5 - Toggle switch

Deze opstelling gaf de volgende resultaten, zie ook de github voor een video van het resultaat: 

Afbeelding 6 - Uitstand lichtknop



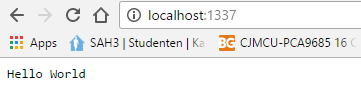
Afbeelding 7 - Aanstaand lichtknop

# **Les week 2**

Deze week stond in het teken van NodeJs en MongoDB.

## **Opdracht 1: Installatie en werkend maken van Node.js**

Bij deze opdracht moest Node.js geïnstalleerd worden en getest worden of het programma goed werkte, de uiteindelijke output zag er zo uit:



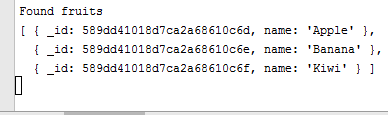
Afbeelding 8 - Simpele NodeJS Server

## **Opdracht 2: Eenvoudige database**

Voor deze opdracht is er gebruik gemaakt van een simpele MongoDB database, waarbij er een aantal fruitsoorten in een collectie werden gestopt, waarna ze vervolgens hier weer uit opgehaald werden. Dit werd gedaan door de volgende twee functies:

**var** *insertFruits* = **function**(db, callback) {  
 **var** fruits = db.**collection**(**'fruits'**);  
  
 fruits.insertMany([ { **name** : **"Apple"**}, { **name**: **"Banana"**}, { **name**: **"Kiwi"**} ], **function** (err, result) {  
 assert.equal(err, **null**);  
 assert.equal(3, result.**result**.**n**);  
 assert.equal(3, result.ops.**length**);  
 **console**.log(**"Inserted Fruits!"**);  
 callback(result);  
 });  
}  
  
**var** *findFruits* = **function**(db, callback) {  
 **var** fruits = db.**collection**(**'fruits'**);  
  
 fruits.find({}).toArray(**function** (err, docs) {  
 assert.equal(err, **null**);  
 **console**.log(**"Found fruits"**);  
 **console**.log(docs);  
 callback(docs);  
 });  
}

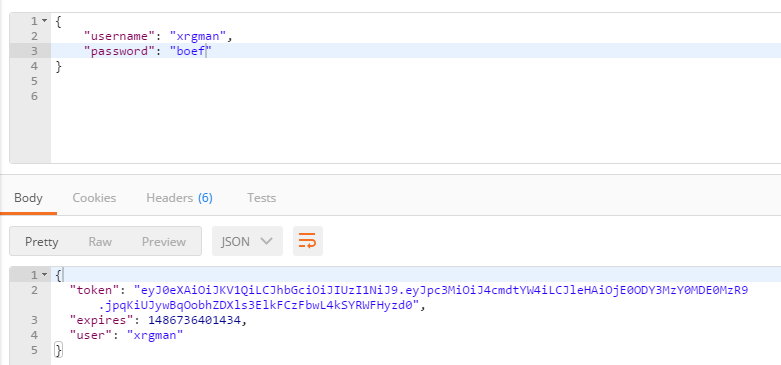
Deze code gaf de volgende output in de terminal:



Afbeelding 9 - Simpele MongoDB database connectie met NodeJS

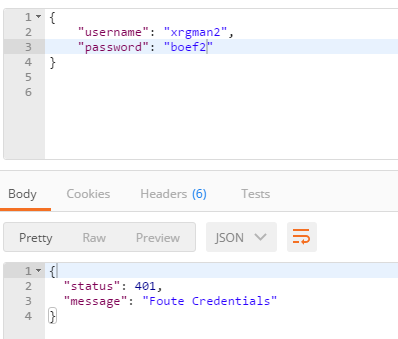
## **Opdracht 3: Maak een RESTful node applicatie**

Bij deze opdracht moest er een RESTful Node applicatie ontwikkeld worden, de eisen van deze applicatie waren dat je kan inloggen via een token. Die door middel van een eindpunt aangevraagd kan worden. Zodra de gebruiker naar het eindpunt /apiv1/login gaat en in de body de juiste username en password invoert krijgt hij een token terug:



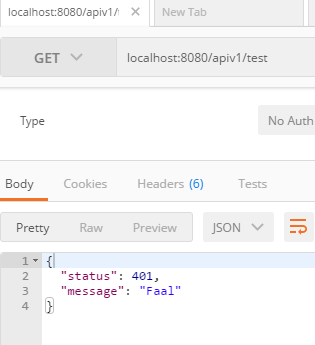
Afbeelding 10 - Succesvolle login REST api

Bij een foute inloggegevens zal de gebruiker geen token ontvangen en een 401 status code terugkrijgen:

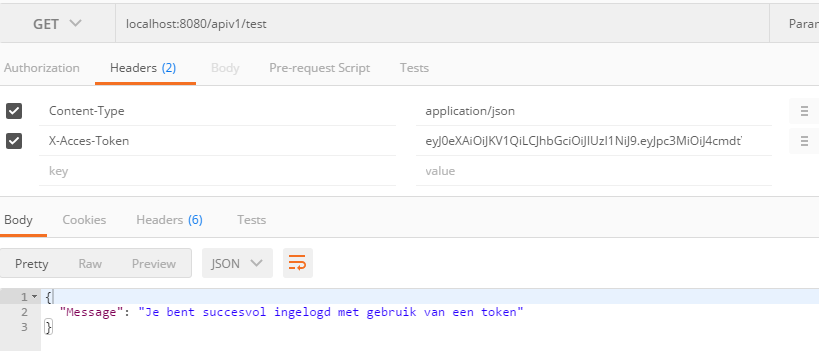


Afbeelding 11 - Onsuccesvolle login REST api

Tot slot is er nog een test eindpunt gemaakt (/apiv1/test) die alleen aangeroepen kan worden door gebruik van de token in de header, hieronder nog twee afbeeldingen van een geslaagde en niet geslaagde benadering van dit eindpunt.



Afbeelding 12 – Aanroepen endpoint zonder token.



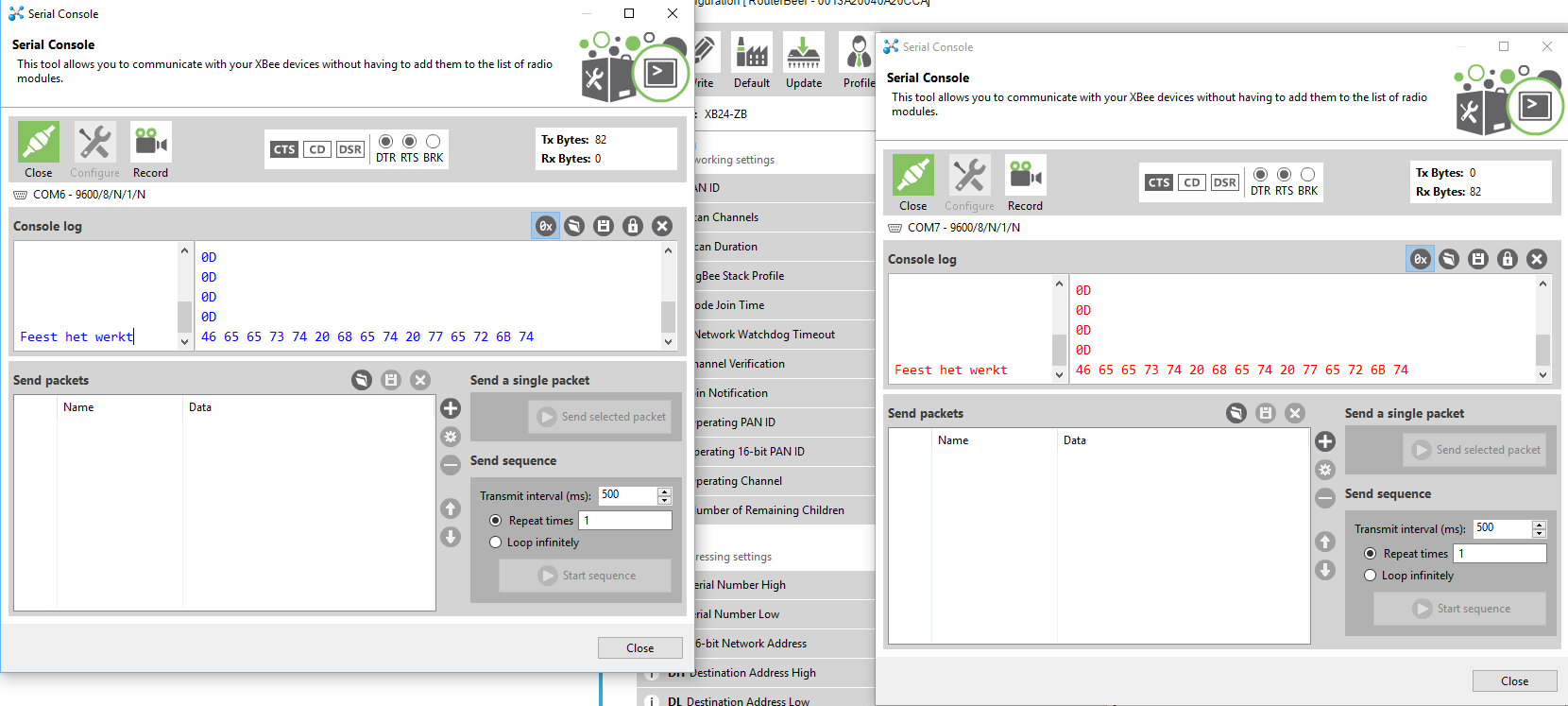
Afbeelding 13 - Aanroepen endpoint met token

# **Lesweek 3**

Deze week stond in het teken van de Zigbee

## **Opdracht 1a: Communicatie tussen router en coördinator**

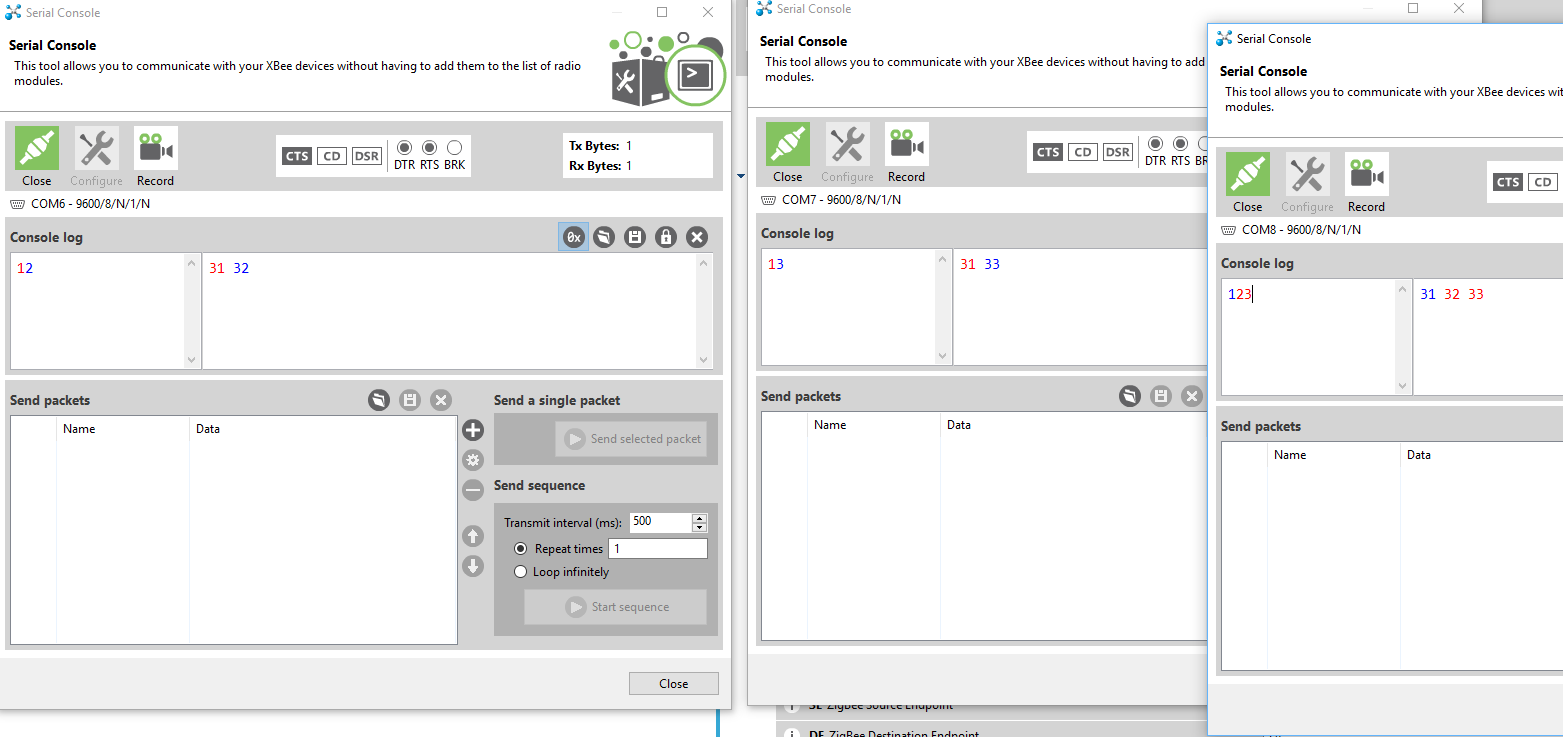
Bij deze opdracht moest er een basis communicatie tussen een coördinator en een router node opgezet worden. Deze communicatie is opgezet en werd getest door middel van twee console terminals. Bij het typen in de routers terminal, zien we de tekst direct bij de coördinators terminal verschijnen.



Afbeelding 14 - Communicatie tussen een coördinator en router node

## **Opdracht 1b: Communicatie tussen coördinator en meerdere router nodes**

Bij deze opdracht moesten er 3 modules aangesloten worden, een coördinator en 2 routers. Hierbij moest de coördinator de berichten van alle routers ontvangen en zijn berichten ook bij alle routers afleveren. De router mochten echter alleen maar met de coördinator communiceren. Dit is te zien in de onderstaande afbeelding, COM6 & COM7 zijn de routers en COM8 is de coördinator:



Afbeelding 15 - Communicatie tussen Coördinator en meerdere router nodes

## **Opdracht 1C: Communicatie met de Arduino**

Bij deze opdracht moesten de routers verbonden worden met een Arduino die om de zoveel tijd een karakter verstuurd naar de coördinator. Dit is gerealiseerd met het volgende stuk code:

void setup() {

Serial.begin(9600);

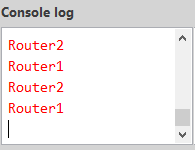
}

void loop() {

Serial.println("Router1");

delay(1000);

}

De tweede router print router2 naar de module, de coördinator ontvangt deze teksten zoals hier te zien is:

Afbeelding 16 - Communicatie tussen Arduino en coördinator

## **Opdracht 2a: Informatie naar het internet**

Bij deze opdracht moest de coördinator module verbonden worden aan het internet door middel van het gebruik van een Arduino. Hierbij stuurt de router node een random getal naar de coördinator node, welke vervolgens door middel van een api call deze waarde in een database zet. De api call ziet er als volgt uit:

void sendRandomNumber(int number) {

HTTPClient http;

String request = "http://192.168.1.132:8080/apiv2/random";

String body = "{ \"Number\": ";

body += number;

body += " }";

http.begin(request); //HTTP

http.addHeader("Content-Type", "application/json;charset=utf-8");

http.addHeader("cache-control", "no-cache");

int httpCode = http.sendRequest("PUT", body);

Serial.printf("[HTTP] GET... code: %d\n", httpCode);

if(httpCode == HTTP\_CODE\_OK) {

String payload = http.getString();

Serial.println(payload);

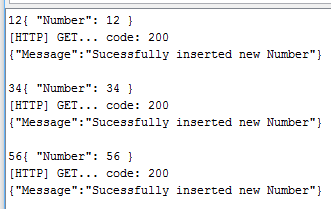
}

http.end();

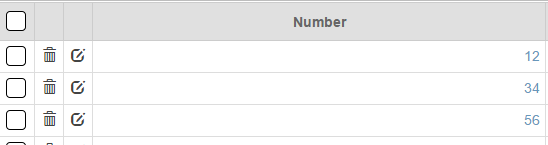
}

Het eindpunt in de REST api zet de waarde vervolgens het nummer in de database met de volgende code:

router.put(**'/random'**, **function** (req, res) {  
  
 **var** number = req.**body**.Number || **''**;  
   
 **if**(number == **''**) {  
 res.**status**(401);  
 res.json({  
 **"status"**: 401, **"message"**: **"Invalid body"** });  
 }  
  
 MongoClient.connect(dbUrl, **function** (err, db) {  
 assert.equal(**null**, err);  
 **console**.log(**"Connected to mongoDB"**);  
 *writeNumberToDatabase*(db, number, **function** () {  
 db.close();  
 });  
 })  
  
 res.**status**(200);  
 res.json({  
 **"Message"**: **"Sucessfully inserted new Number"** });  
  
});  
  
**var** *writeTempValueToDatavbase* = **function**(db, temps, callback) {  
 **var** temperatures = db.collection(**'TempVals'**);  
  
 temperatures.insertMany(temps, **function** (err, result) {  
 assert.equal(err, **null**);  
 **console**.log(**"Inserted Temperatrues!"**);  
 callback(result);  
 });  
}



Afbeelding 17 - Ontvangen van het random getal van een router en het sturen naar de api



Afbeelding 18 - Resultaat in de database

## Opdracht 2b: Temperatuurmetingen naar de database

Bij deze opdracht moest er een temperatuur meter aangesloten worden op een router Arduino node. Deze node zendt elke 10 seconden zijn temperatuur naar de coördinator, welke vervolgens een api call maakt om de temperatuur in de database te zetten. De temperatuur zending werd gerealiseerd met de volgende code:

#include <DHT.h>

#include <DHT\_U.h>

DHT dht(3,DHT11);

void setup() {

Serial.begin(9600);

dht.begin();

}

void loop() {

float temperature = dht.readTemperature();

Serial.println(temperature);

delay(2000);

}

Waarna de coördinator node de temperatuur op de volgende manier ontvangt:

if(zigbee.available()) {

char letter = zigbee.read();

handleSerialCommand(letter);

}

Void handleSerialCommand(char letter) {

if(letter != '\n' && letter != '\r') {

serialCmd += letter;

}

else if(serialCmd.length() > 0) {

int number = serialCmd.toInt();

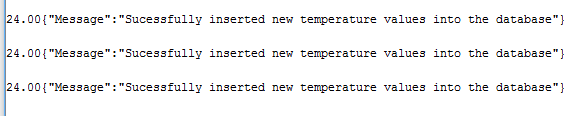
sendTemperature(number);

serialCmd = "";

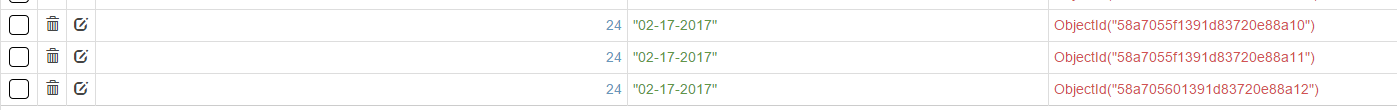
}

}

Het aanroepen van de REST api en het wegschrijven in de database gaat op dezelfde manier als bij het random nummer, van de opdracht hierboven.

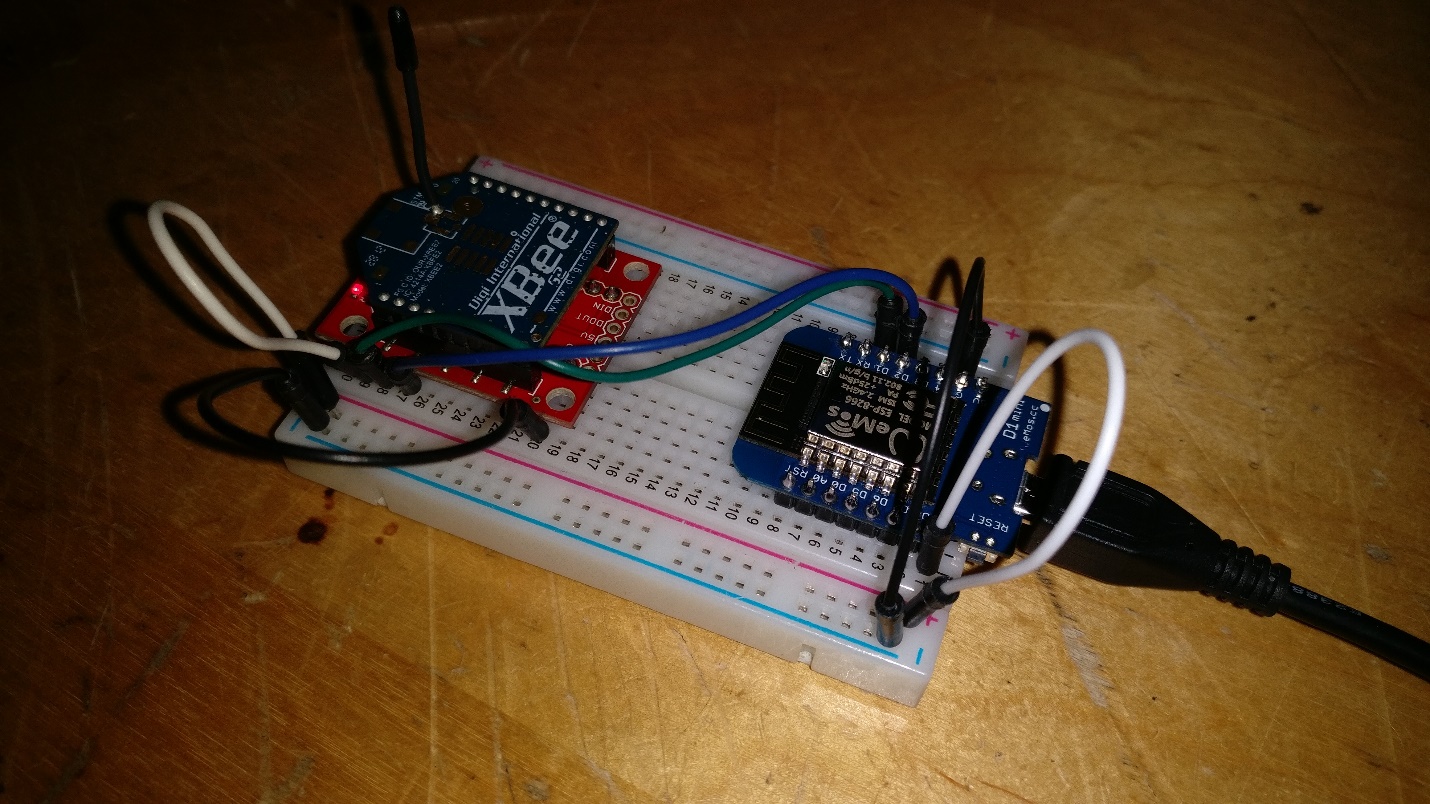


Afbeelding 19 - Ontvangen temperatuur waarden bij de coördinator node

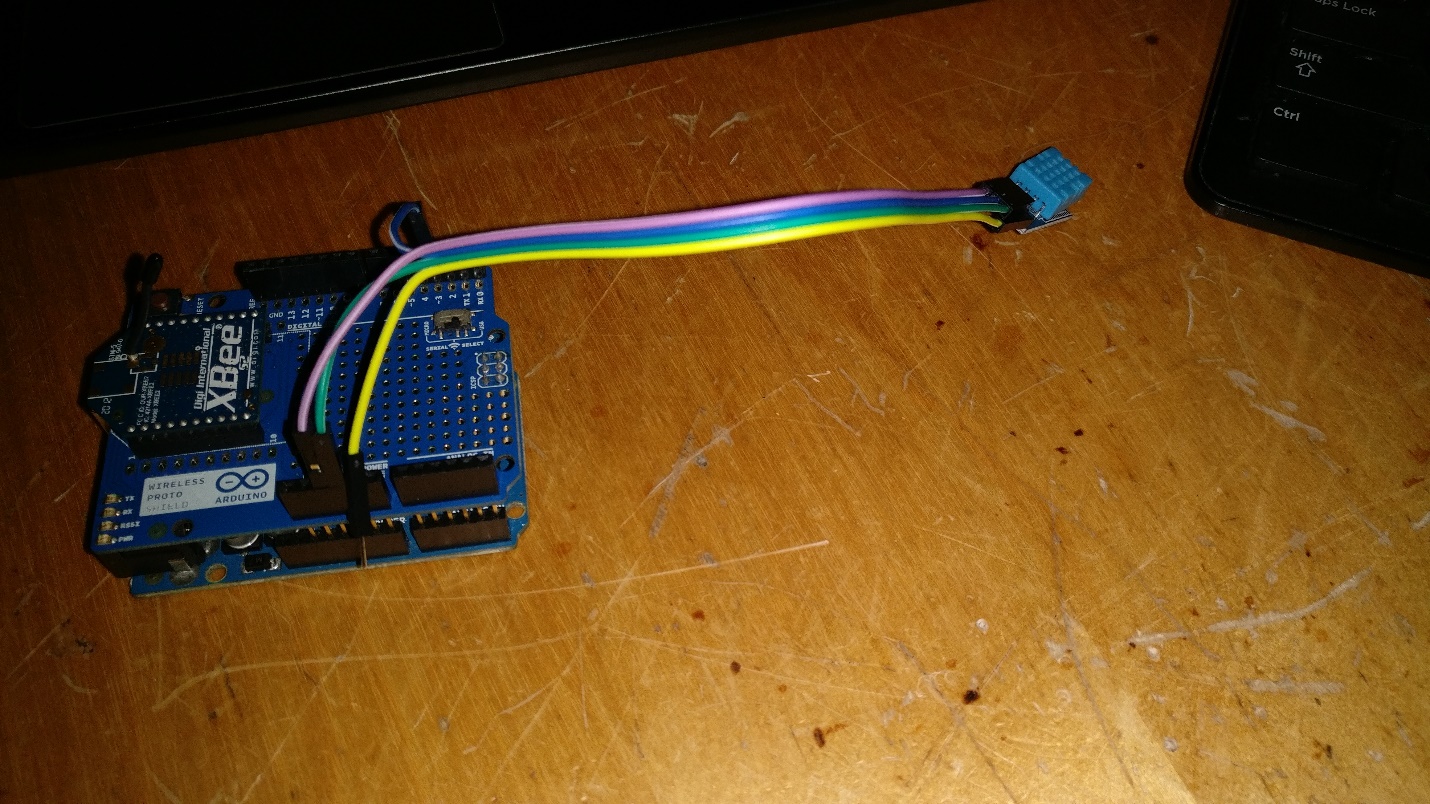


Afbeelding 20 - Temperatuur waardes in de database

De bovenstaande twee opdrachten zijn gerealiseerd met de onderstaande hardware opstelling:



Afbeelding 21 - Coördinator node



Afbeelding 22 - Router node

hoi