

Part 1:
The 35 euro IoT project

Oene Bakker © 2016-2017

Part 1: The 35 euro IoT project

1 Preface

As of the early 1980's, I am a programmer in automation. Started as a COBOL programmer, I think I took a lot of the movements in ICT. And until 2013 it was a pretty static world for me. Of course, the internet came up and connectivity became an increasingly important subject, also for me. How do we connect the client computers to the servers? But for me, this meant, above all, the construction of software. Hardware setup and configuration was done by others.

However, I was fascinated by electronics since my youth. When the first calculators came on the market, I could not wait to buy one (it cost all of my pocket money). And I heard of transistors (in radios), resistors, diodes and all kinds of other electronic components. But it were only theoretical concepts for me. I really did not have any idea how these things work. Let alone I was able to build my own electronic circuits.

Somewhere in the summer vacation of 2013, I first came into contact with the Arduino platform. A new world opened for me. For the first time, I could make electronic circuits and program them myself. How cool is that? The first project was, of course, the blinking of an led using an Arduino.

But yes, they were standalone boards. By default, there was no connection to a local network or internet.

With the arrival of the ESP8266 boards it changed quickly. Although programming (flashing) with the so-called LUA interface left over, it was a step forward. Certainly when the so-called ESP-12E Development Kit came out. A board that has almost as many possibilities as an Arduino but with builtin Wi-Fi capabilities. And that for a very low price!

In this book, the ESP-12E Development Kit is used to measure temperature and humidity. This data is then displayed with a AngularJS application in a browser. And in the last chapter, an example of an Android App is shown.

The software used in the book is free to download. And the software and scripts are also made available for free. The required hardware must be purchased. I estimate that the total hardware costs will be around € 35, if ordered in China.

Lots of reading and DIY fun!

July 11, 2017
Oene Bakker
The Westereen

Part 1: The 35 euro IoT project

Table of Contents

| | | |
|----------|-------------------------------------|-------------------------------------|
| 1 | Voorwoord..... | Fout! Bladwijzer niet gedefinieerd. |
| 2 | Het idee | Fout! Bladwijzer niet gedefinieerd. |
| 2.1 | Verantwoording..... | Fout! Bladwijzer niet gedefinieerd. |
| 2.2 | Voorkennis..... | Fout! Bladwijzer niet gedefinieerd. |
| 2.3 | Leeswijzer..... | Fout! Bladwijzer niet gedefinieerd. |
| 2.4 | Afkortingen..... | 11 |
| 3 | Gebruikte technieken..... | 12 |
| 3.1 | Android..... | 12 |
| 3.1.1 | Ontwikkeling..... | 12 |
| 3.1.2 | Applicaties | 12 |
| 3.1.3 | Versies | 13 |
| 3.2 | AngularJS | 14 |
| 3.2.1 | SPA..... | 14 |
| 3.2.2 | De cliënt en de server..... | 14 |
| 3.3 | MQTT..... | 15 |
| 3.3.1 | Inleiding | 15 |
| 3.3.2 | De werking van MQTT | 16 |
| 3.3.2.1 | Connection | 16 |
| 3.3.2.2 | Authentication..... | 16 |
| 3.3.2.3 | Communication | 16 |
| 3.3.2.4 | QoS | 17 |
| 3.3.2.5 | Unacknowledged Service (QoS0) | 17 |
| 3.3.2.6 | Acknowledged Service (QoS1)..... | 17 |
| 3.3.2.7 | Assured Service (QoS2)..... | 18 |
| 3.3.2.8 | Operaties | 18 |
| 3.3.2.9 | Publish | 18 |
| 3.3.2.10 | Subscribe | 18 |
| 3.3.2.11 | Unsubscribe | 18 |
| 3.3.2.12 | Ping | 18 |
| 3.3.3 | Topic strings..... | 19 |
| 3.3.3.1 | Enkelvoudige wildcard..... | 19 |

Part 1: The 35 euro IoT project

| | |
|---|----|
| 3.3.3.2 Meervoudige wildcard..... | 20 |
| 3.3.3.3 Uitsluitingskarakter | 20 |
| 3.3.4 Termination | 20 |
| 3.3.5 MQTT implementaties..... | 21 |
| 3.4 NTP | 22 |
| 3.4.1 Techniek | 22 |
| 3.4.2 Nederland..... | 22 |
| 3.5 The Cloud..... | 23 |
| 3.5.1 Inleiding | 23 |
| 3.5.2 MongoDB..... | 23 |
| 4 Boodschappenlijstje | 24 |
| 4.1 Inleiding | 24 |
| 4.2 Hardware..... | 24 |
| 4.2.1 Kostenplaatje..... | 24 |
| 4.2.2 ESP-12E Development Kit..... | 25 |
| 4.2.3 DHT22 Sensor | 26 |
| 4.2.4 DS1307 RTC | 26 |
| 4.2.5 Level Shifter..... | 27 |
| 4.2.6 Arduino power adapter | 28 |
| 4.2.7 USB micro kabel..... | 28 |
| 4.2.8 Breadboard en overige benodigdheden | 29 |
| 4.2.9 Kopen in China..... | 30 |
| 4.2.10 Kopen in Nederland of de EU | 31 |
| 4.3 Software | 32 |
| 4.3.1 Inleiding | 32 |
| 4.3.2 Arduino IDE..... | 33 |
| 4.3.2.1 Installatie | 33 |
| 4.3.2.2 Arduino Bibliotheken..... | 38 |
| 4.3.2.3 Blink | 39 |
| 4.3.3 NodeJS | 45 |
| 4.3.3.1 NodeJS installatie | 45 |
| 4.3.3.2 Installeren benodigde package | 50 |
| 4.3.3.3 Server script aanmaken..... | 53 |

Part 1: The 35 euro IoT project

| | | |
|----------|-------------------------------|-----|
| 4.3.3.4 | Starten server | 54 |
| 4.3.4 | AngularJS | 54 |
| 4.3.4.1 | AngularJS installatie..... | 54 |
| 4.3.4.2 | HTML5 Canvas Gauge 1..... | 56 |
| 4.3.4.3 | HTML5 Canvas Gauge 2..... | 56 |
| 4.3.4.4 | Een simpel voorbeeld..... | 58 |
| 4.3.5 | Chart.js..... | 60 |
| 4.3.5.1 | Download | 60 |
| 4.3.5.2 | Test | 61 |
| 4.3.6 | Mosquitto | 63 |
| 4.3.6.1 | Inleiding | 63 |
| 4.3.6.2 | Installatie | 63 |
| 4.3.6.3 | Broker | 77 |
| 4.3.6.4 | Subscriber | 78 |
| 4.3.6.5 | Publisher | 78 |
| 4.3.7 | Eclipse Java | 79 |
| 4.3.7.1 | Inleiding | 79 |
| 4.3.7.2 | Installatie | 79 |
| 4.3.7.3 | Java Hello World..... | 83 |
| 4.3.8 | PAHO Java cliënt..... | 87 |
| 4.3.8.1 | Inleiding | 87 |
| 4.3.8.2 | Installatie | 87 |
| 4.3.8.3 | Java MQTT cliënt | 88 |
| 4.3.8.4 | Testen cliënt..... | 94 |
| 4.3.9 | Android Studio 2.x..... | 96 |
| 4.3.9.1 | Inleiding | 96 |
| 4.3.9.2 | Java JDK 1.8 | 96 |
| 4.3.9.3 | Android Studio..... | 102 |
| 4.3.9.4 | Android Hello World App | 108 |
| 4.3.10 | MongoDB installeren..... | 121 |
| 4.3.10.1 | Drivers..... | 121 |
| 4.3.10.2 | Installeren..... | 121 |
| 4.3.10.3 | MongoDB..... | 121 |

Part 1: The 35 euro IoT project

| | | |
|----------|---|-----|
| 4.3.10.4 | MongoDB NodeJS driver..... | 125 |
| 4.3.10.5 | MongoDB starten | 125 |
| 4.3.10.6 | JSON importeren | 127 |
| 4.3.10.7 | NodeJS script | 128 |
| 4.3.11 | Virtual Router | 131 |
| 4.3.11.1 | Geen Wi-Fi en nu? | 131 |
| 4.3.11.2 | Installatie | 131 |
| 4.3.11.3 | Starten Virtual Router | 136 |
| 5 | Bouwen..... | 138 |
| 5.1 | Hardware | 138 |
| 5.1.1 | Breadboard en voeding | 138 |
| 5.1.2 | ESP-12E DEVKIT | 139 |
| 5.1.3 | DHT22 | 140 |
| 5.1.4 | DS1307 RTC | 141 |
| 5.1.5 | Hardware schema..... | 142 |
| 5.2 | Software | 144 |
| 5.2.1 | NodeJS, Mosca en MongoDB | 144 |
| 5.2.1.1 | Inleiding | 144 |
| 5.2.1.2 | Javascript source | 144 |
| 5.2.2 | AngularJS dashboard | 147 |
| 5.2.2.1 | Inleiding | 147 |
| 5.2.2.2 | AngularJS HTML en JavaScript sources..... | 147 |
| 5.2.3 | ESP-12E MQTT programma | 151 |
| 5.2.3.1 | Inleiding | 151 |
| 5.2.3.2 | Source..... | 151 |
| 5.2.3.3 | Code uploaden | 161 |
| 6 | Testen | 164 |
| 6.1 | ESP-12E en WiFi..... | 164 |
| 6.1.1 | WiFi thuis..... | 164 |
| 6.1.2 | Mobiele Hotspot..... | 164 |
| 6.1.3 | Virtual Router | 165 |
| 6.2 | ESP-12E, NodeJS en Mosca..... | 165 |
| 6.2.1 | Starten NodeJS Express server en Mosca broker | 165 |

Part 1: The 35 euro IoT project

| | | |
|---------|--------------------------------------|-----|
| 6.2.2 | Starten ESP-12E | 166 |
| 6.2.3 | Starten Chrome browser | 167 |
| 7 | Android | 168 |
| 7.1 | App maken..... | 168 |
| 7.1.1 | AndroidManifest.xml..... | 169 |
| 7.1.2 | activity_main.xml | 170 |
| 7.1.3 | config.properties | 170 |
| 7.1.4 | Android Java code | 171 |
| 7.1.4.1 | Inleiding | 171 |
| 7.1.4.2 | PropertiesReader.java | 171 |
| 7.1.4.3 | MainActivity.java | 173 |
| 7.2 | App testen | 177 |
| 7.2.1 | Voorbereiding..... | 177 |
| 7.2.2 | Android App..... | 178 |
| 7.2.3 | Smartphone test..... | 181 |
| 7.2.3.1 | Samsung USB-driver | 181 |
| 7.2.3.2 | Ontwikkelaarsopties inschakelen..... | 184 |
| 7.2.3.3 | MQTest App testen | 185 |
| 8 | The Cloud..... | 188 |
| 8.1 | Inleiding | 188 |
| 8.2 | Grafieken | 188 |
| 8.2.1 | NodeJS Express server | 188 |
| 8.2.2 | Index.html..... | 191 |
| 8.2.3 | Test | 194 |
| 9 | Index | 196 |

Part 1: The 35 euro IoT project

2 The Concept

The idea of showing the temperature and humidity in a DIY project is not new. There are numerous examples on the internet using an Arduino or Raspberry PI to accomplish this.



By the end of 2015, I came into contact with ESP8266. A small stand-alone Wi-Fi module for a few euros (ESP-01). Programming via the so-called LUA loader was a difficult for me. Sometimes it worked, but very often not. The integration with the Arduino IDE was therefore an outcome for me. With the arrival of the ESP-12E Development Kit, USB programming was also possible. Although not new, I also came up with the idea of combining it with the world of IOT.

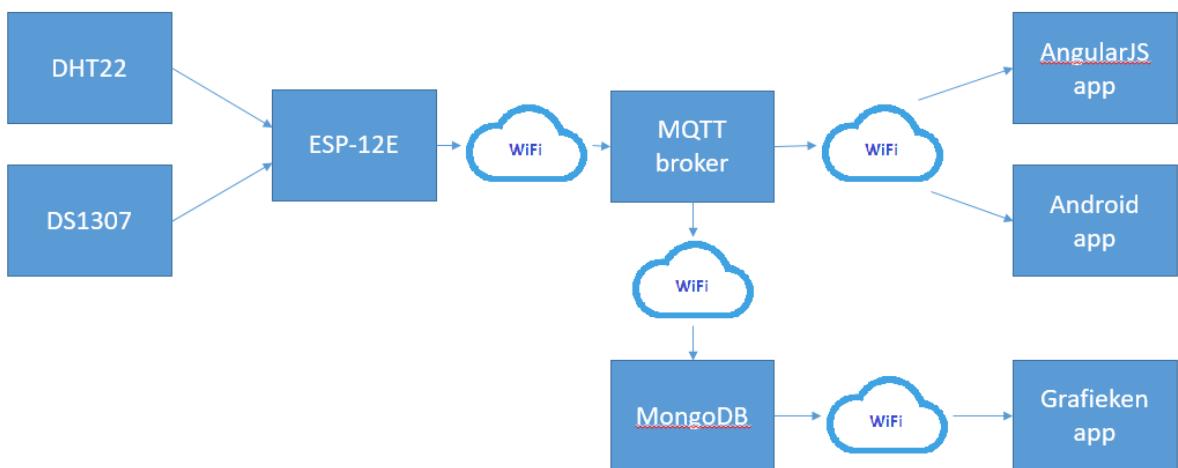
The concept:

Koppel een DHT22 temperatuur en luchtvochtigheidsensor en een DS1307 RTC-klok aan een ESP-12E Development Kit. De ESP-12E publiceert de data via MQTT* naar een MQTT-broker. Een AngularJS applicatie en een Android App abonneren zich op deze data. De AngularJS applicatie toont de data in een temperatuur- en vochtigheidsmeter. De Android App toont de gepubliceerde data op het scherm van smartphone. De data wordt opgeslagen in een MongoDB database en deze data wordt later gebruikt voor het tonen van temperatuur en vochtigheidsgrafieken.

Connect a DHT22 temperature and humidity sensor and a DS1307 RTC clock to an ESP-12E Development Kit. The ESP-12E publishes the data via MQTT * to a MQTT broker. An AngularJS application and an Android App subscribe to this data. The AngularJS application shows the data in a temperature and humidity gauge. The Android App shows the published data on the smartphone screen. The data is stored in a MongoDB database and this data is used later to show temperature and humidity graphs.

*) MQTT Is discussed extensively in this book.

Schema:



Part 1: The 35 euro IoT project

For this solution, I use the following operating systems:

- ESP8266 NodeMCU v1
 - ESP-12E Development Kit
- Windows 10 *)
 - Pc, laptop or tablet
- Android 5.0.1
 - Smartphone (in this case an oldie Samsung Galaxy S4).

The key goals I have set for this project are:

- It must be cheap, I have set a maximum of 50 euros for myself.
- It must be accessible (the hardware and software must be (freely) available).
- It must be understood by all.
- It needs to teach me something about IOT.

*) In principle, it is also possible to use Windows 7, 8, or 8.1. Certain images shown may than differ.

2.1 Accountability

The texts quoted and recorded in this book are as far as I know as a writer from the so-called free domain (free pics, public text). If this is not the case, I would like to express my apologies.

This book has been compiled with the utmost care and the solutions shown have been extensively tested. However, if errors occur this has been done without any intention. Despite all the care taken in the composition of this book, the author can not be held liable for any damage resulting from any error in this publication.

The book shows a hardware solution with estimated costs of approximately € 35, -. This price was at the time of writing and is subject to exchange rate fluctuations of, among other things, the dollar rate. The author can not be held liable for this and for the availability of the hardware used.

The assumption is that the reader is in possession of a PC, laptop or tablet with preferably Windows 10. In addition, the ownership of an Android smartphone with Android version 5.0.1 or above is recommended but not required (the solution also works with an emulator).

Working with the required hardware and power adapters can be a risk. The solution shown works with low voltages (max. 5 volt) which, of course, limits the risks. Incorrectly connecting may damage your hardware irreparably! The author can not be held liable for any damage resulting from this. It's all at your own risk.

The contents of this book may not be commercially used. The reader is free to use the contents of this book for private and hobby purposes. This also applies to use this books and its content in education. The sources associated with this book may be copied, used and / or modified without any limitation.

Part 1: The 35 euro IoT project

2.2 Prescience

Knowledge of programming is a pre, but with some perseverance it must be possible for every hobbyist to realize the solutions shown. Also, no knowledge of soldering is needed because we a so-called breadboard solution. The layout of this book is low-threshold and works step by step to the final effect.

And as often, Google is our best friend (or any other search engine, by the way).

For questions about the contents of this book, the following email address is available:
Info@diyiot.nl

The author will do his best to answer all the questions.

2.3 Reading guide

This book is divided into an introductory part (chapters 1 and 2), a theoretical part (chapter 3) and a practical part (chapter 4 and beyond).

The sources can be downloaded from www.diyiot.nl/download

Extract the downloaded zip file. The book refers to the sources by:
See: Sources → ... → ...

2.4 Version management

| Version | Date | Remarkt |
|---------|------------|--|
| 1.0 | 30-06-2016 | New (Dutch version) |
| 1.1 | 05-07-2017 | Reader remarks applied |
| 2.0 | 31-07-2017 | Book translated to the English version |

Part 1: The 35 euro IoT project

2.5 Afkortingen

| Afkorting | Betekenis |
|-----------|--|
| API | Application Programming Interface |
| BSON | Binair JSON |
| DHT | Digital Humidity Temperature (sensor) |
| DIY | Do It Yourself |
| ESP | ESPressif (Chinese fabrikant) |
| EU | Europese Unie |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| IDE | Integrated Development Environment |
| IoT | Internet of Things |
| JDK | Java Development Kit |
| JS | Java Script |
| JSON | Java Script Object Notation |
| LIFO | Last In – First Out |
| LUA | Loe-ah = maan in het Portugees, dus geen afkorting |
| MQTT | Message Queue Telemetry Transport |
| M2M | Machine to Machine |
| NTP | Network Time Protocol |
| ODM | Object Data Modelling |
| OHA | Open Handset Alliance |
| QoS | Quality of Service |
| RDBMS | Relational Database-Management System |
| RTC | Real Time Clock |
| SDK | Software Development Kit |
| SPA | Single Page Application |
| SSL/TLS | Secure Sockets Layer/ Transport Layer Security |
| TCP/IP | Transmission Control Protocol/ Internet Protocol |
| USB | Universal Serial Bus |

3 Gebruikte technieken

3.1 Android



Android is een opensourceplatform en besturingssysteem voor mobiele telefoons, tablet-pc's, camera's en meer, gebaseerd op de Linuxkernel en het Java-programmeerplatform. Android is het meest verkochte besturingssysteem op mobiele telefoons. In totaal zijn er in meer dan 1,4 miljard Android-apparaten geactiveerd (september 2015).

3.1.1 Ontwikkeling

Android is ontwikkeld door Android Inc., een bedrijf dat in 2005 werd overgenomen door Google. Later werd het bedrijf door Google ondergebracht in de Open Handset Alliance (OHA). Het originele idee achter Android was het ontwikkelen van een besturingssysteem voor camera's met een ingebouwde ondersteuning voor Cloud-mogelijkheden. Android wordt nu voornamelijk gebruikt voor smartphones en tablets. Met de vrijgave van het Android-platform op 5 november 2007 werd ook de oprichting van de OHA aangekondigd. Dit samenwerkingsverband bestond bij de oprichting uit 34 hardware-, software- en telecommunicatiebedrijven die zich erop richten open standaarden voor mobiele apparaten te bevorderen. Het opensourceplatform is beschikbaar onder de Apache licentie. De eerste fabrikanten van mobiele telefoons die gebaseerd waren op Android waren Motorola, HTC, Samsung, LG, Sony-Ericsson, en Acer. Android stond tot aan de lancering ervan bekend onder de naam gPhone.

3.1.2 Applicaties

Om Android als platform te presenteren moesten er apps worden ontwikkeld. Om dat proces te ondersteunen is een Software Development Kit (SDK) vrijgegeven. Hiermee wordt het ook voor anderen dan Google mogelijk hun bijdrage te leveren. Applicaties kunnen verspreid worden via de Google Play Store (vroeger de Android Market). Via de Play Store kunnen eindgebruikers games, applicaties, films en boeken downloaden of kopen. De mogelijkheden verschillen per land. Op 26 juni 2013 kreeg België toegang tot Google Play Books, Nederland volgde op 18 juli in hetzelfde jaar. Ontwikkelaars kunnen de Play Store gebruiken om hun software te verspreiden. Via de Play Store kunnen zowel gratis als betaalde applicaties worden verspreid.

Part 1: The 35 euro IoT project

3.1.3 Versies

Inmiddels is er een aantal versies van Android verschenen. Deze zijn bij de meeste mensen bekend met de namen die Google aan zijn releases meegeeft. De namen zijn zoetwaren en zijn alfabetisch gerangschikt.

| Codenaam | Versie | Uitgavedatum | API-versie |
|--------------------|---------------|---------------------|------------|
| Marshmallow | Android 6.0 | 5 oktober 2015 | 23 |
| | Android 5.1 | 9 maart 2015 | 22 |
| | Android 5.0 | 17 oktober 2014 | 21 |
| KitKat | Android 4.4 | 25 juni 2014 | 20 |
| | | 31 oktober 2013 | 19 |
| Jelly Bean[5] | Android 4.3 | 24 juli 2013 | 18 |
| | Android 4.2 | 13 november 2012[6] | 17 |
| | Android 4.1 | 9 juli 2012 | 16 |
| Ice Cream Sandwich | Android 4.0.3 | 16 december 2011 | 15 |
| | Android 4.0 | 19 oktober 2011 | 14 |
| Honeycomb | Android 3.2 | 15 juli 2011[7] | 13 |
| | Android 3.1 | 10 mei 2011 | 12 |
| | Android 3.0 | 22 februari 2011 | 11 |
| Gingerbread | Android 2.3.3 | 9 februari 2011 | 10 |
| | Android 2.3 | 6 december 2010 | 9 |
| Froyo | Android 2.2 | 20 mei 2010 | 8 |
| Eclair | Android 2.1 | 12 januari 2010 | 7 |
| | Android 2.0.1 | 3 december 2009 | 6 |
| | Android 2.0 | 26 oktober 2009 | 5 |
| Donut | Android 1.6 | 15 september 2009 | 4 |
| Cupcake | Android 1.5 | 30 april 2009 | 3 |
| Battenberg | Android 1.1 | 9 februari 2009 | 2 |
| Angel Cake | Android 1.0 | 23 september 2008 | 1 |

Bron

Vrij vertaald naar: [https://nl.wikipedia.org/wiki/Android_\(besturingssysteem\)](https://nl.wikipedia.org/wiki/Android_(besturingssysteem))

Part 1: The 35 euro IoT project

3.2 AngularJS



AngularJS is een JavaScript-framework dat geheel ontwikkeld is om tegemoet te komen aan de eisen van grote en complexe webapplicaties. AngularJS is een open-source initiatief in nauwe samenwerking met Google.

3.2.1 SPA

AngularJS is vooral bedoeld voor het bouwen van zogenaamde Single Page Applications (SPA). In AngularJS ontwikkel je echter niet één grote applicatie waarin alles met alles verbonden is. In plaats daarvan maak je kleinere, gespecialiseerde modules. Deze zijn los van elkaar te ontwikkelen en te testen. Op de plekken waar ze nodig zijn in de applicatie, worden ze dynamisch ingevoegd via het principe van Dependency Injection. Een applicatie wordt zo bijvoorbeeld verdeeld in modules voor bijvoorbeeld:

- Het tonen van gegevens in een HTML-pagina,
- Het ophalen van gegevens van de server,
- Voor routing,
- Constanten en
- Voor in- en uitloggen.

3.2.2 De cliënt en de server

Kenmerk van AngularJS-applicaties is dat de applicatie in principe volledig in de browser draait. AngularJS is een client-sided framework. AngularJS-toepassingen zijn in principe volledig onafhankelijk van een server. De server wordt alleen gebruikt voor het ophalen en updaten van data.

Bron

Vrij vertaald naar: <http://www.kassenaar.com/blog/post/2014/04/14/Wat-is-AngularJS-Een-introductie.aspx>

Part 1: The 35 euro IoT project

3.3 MQTT

3.3.1 Inleiding

MQTT (Message Queuing Telemetry Transport) is een lichtgewicht messaging protocol speciaal bedoeld voor netwerk cliënt-computers met beperkte mogelijkheden. Het is speciaal bedoeld voor het distribueren van telemetrische informatie (bijv. temperatuur, luchtvochtigheid, snelheid, toerental van een motor, e.d.). Het is gebaseerd op het zogenaamde publish/subscribe communicatiepatroon en is speciaal bedoeld voor M2M (machine-to-machine) communicatie. Het speelt immiddels een zeer belangrijke rol in de Internet of Things (IoT).

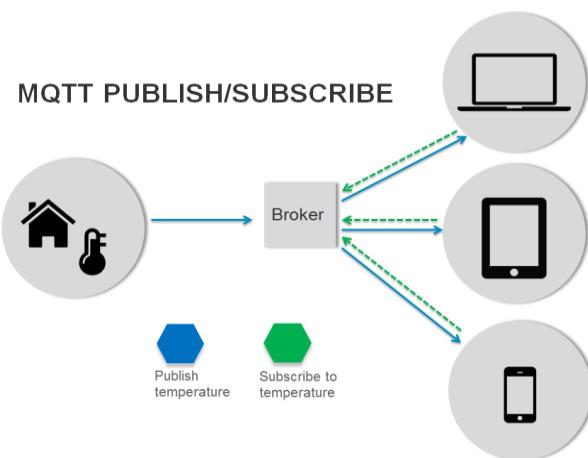
MQTT maakt het mogelijk dat apparaten (devices) informatie over een bepaald onderwerp (topic) versturen naar een zogenaamde MQTT message broker. Deze broker stuurt deze informatie naar alle cliënt devices die een abonnement (subscription) hebben op deze informatie. Het is mogelijk om met een zogenaamde wildcard een abonnement te nemen op meerdere topics.

Een paar basisbegrippen:

- Een broker is een server die de berichtafhandeling tussen de diverse cliënten regelt. De broker ontvangt berichten van de publisher en stuurt deze naar één of meer subscribers.
- Een publisher is een cliënt die een bericht met telemetrische gegevens naar de broker stuurt.
- Een subscriber is een cliënt die een bericht met telemetrische gegevens van de broker ontvangt.

Dit is een beetje te vergelijken met een uitgeverij (publisher) die tijdschriften via de post (broker) naar de abonnees (subscribers) stuurt.

In onderstaande voorbeeld wordt de temperatuur gemeten door een sensor (publisher) in een woning via een broker gestuurd naar een laptop, tablet en mobiel (subscribers).



MQTT is ook een goede keuze als het gaat om onbetrouwbare netwerken met lage bandbreedte en variabele wachttijden. Zodra een netwerk richting de subscribers tijdelijk niet beschikbaar is worden berichten opgeslagen in een buffer en alsnog verzonden zodra de subscriber weer online is. Andersom, indien het netwerk richting de publisher niet beschikbaar is kunnen de subscribers hier via een melding bericht van krijgen.

Part 1: The 35 euro IoT project

MQTT is niet nieuw, het is 1999 ontwikkeld door Dr. Andy Standford-Clark van IBM en Arlen Nipper van Arcom (nu Eurotech). Initieel was het bedoeld voor het monitoren van de olie en gasindustrie. Hoewel MQTT nog steeds met IBM wordt geassocieerd is het inmiddels een open protocol beheerd door de Organization for the Advancement of Structured Information Standards (OASIS). MQTT is een behoorlijk uitontwikkeld product dat o.a. wordt gebruikt door Facebook, Amazon en vele IoT-projecten.

3.3.2 De werking van MQTT

MQTT werkt met sessies (sessions) die bestaan uit 4 stadia (stages):

1. Connectie (connection)
2. Authenticatie (authentication)
3. Communicatie (communication)
 - a. QoS
 - b. Operaties
 - c. Topic strings
 - d. Overzicht MQTT berichten
4. Terminatie (termination)

3.3.2.1 Connection

Cliënt-computers maken via TCP/IP connectie met de broker. Hiervoor wordt een standaard of eigen configurerde poort gebruikt. De standaardpoorten zijn:

- 1883 voor niet versleutelde communicatie en
- 8883 voor versleutelde (SSL/TLS) communicatie.

Indien mogelijk worden verbindingen door de server hergebruikt.

3.3.2.2 Authentication

Authenticatie is niet altijd nodig. Er zijn brokers die ook anonieme toegang toestaan. Bij niet versleutelde communicatie worden gebruikersnaam en wachtwoord leesbaar doorgegeven. Bij versleutelde communicatie vindt er een SSL/TLS handshake met de broker plaats waarbij gebruik wordt gemaakt van certificaten.

3.3.2.3 Communication

MQTT is een lichtgewicht protocol omdat berichten in principe klein zijn. Een bericht bestaat over het algemeen uit:

- Een zogenaamde fixed header van 2 bytes
- Een optionele header,
- Een bericht (message payload) van maximaal 256MB en
- Een zogenaamde Quality of Service (QoS).

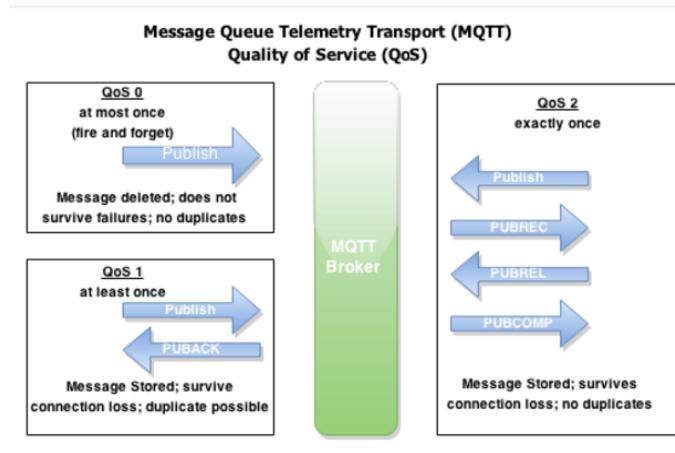
Part 1: The 35 euro IoT project

3.3.2.4 QoS

Er zijn drie verschillende levels van QoS. De QoS bepaalt hoe berichten worden behandeld door het MQTT protocol. Hogere QoS levels bieden hogere betrouwbaarheid maar dit gaat wel ten koste van bandbreedte en langere wachttijden.

QoS levels:

0. Unacknowledged Service (QoS0).
1. Acknowledged Service (QoS1).
2. Assured Service (QoS2).



3.3.2.5 Unacknowledged Service (QoS0)

Hierbij wordt gebruik gemaakt van de zogenaamde PUBLISH packet sequence. De publisher stuurt eenmalig het bericht naar de broker. De broker stuurt dit bericht eenmalig naar alle subscribers. Of het bericht ook daadwerkelijk aankomt is niet zeker. Ook bewaard de broker het bericht niet. Als het netwerk richting een subscriber niet beschikbaar wordt het bericht dus niet nog een keer verstuurd zodra het netwerk weer beschikbaar is. Synoniemen voor dit level zijn:

- At Most Once
- Fire and Forget
- QoS0

3.3.2.6 Acknowledged Service (QoS1)

Hierbij wordt gebruik gemaakt van de zogenaamde PUBLISH/PUBACK packet sequence tussen publisher en broker en ook tussen broker en subscribers. Het versturen en ontvangen van berichten wordt altijd geverifieerd. Berichten worden bewaard op de broker. Er is sprake van een zogenaamd retry mechanisme als een subscriber niet op tijd een bevestiging van ontvangst geeft (acknowledgement). Dit kan er wel in resulteren dat een subscriber meerdere kopieën van hetzelfde bericht kan krijgen. Synoniemen voor dit level zijn:

- At Least Once
- QoS1

Part 1: The 35 euro IoT project

3.3.2.7 Assured Service (QoS2)

In dit level wordt een bericht in twee pakketjes verstuurd. Het eerst pakket wordt een PUBLISH/PUBREC genoemd en het tweede pakket een PUBREL/PUBCOMP. In deze variant wordt ervoor gezorgd dat een bericht altijd één keer aankomt. Berichten worden hier ook bewaard op de broker. Het maakt niet uit hoeveel retries er plaats vinden, het bericht komt precies één keer aan bij de subscribers. Synoniemen voor dit level zijn:

- Exactly Once
- QoS2

3.3.2.8 Operaties

Tijdens de communicatie fase kan een cliënt de volgende operaties uitvoeren:

- Publish.
- Subscribe.
- Unsubscribe.
- Ping.

3.3.2.9 Publish

Een publisher stuurt berichten naar de broker over een bepaald door de publisher bepaald onderwerp. Dit onderwerp wordt topic genoemd. De publisher stuurt een blok van binaire data (content) naar deze topic. MQTT ondersteunt het versturen van zogenaamde Binary Large Objects (BLOB) met een maximale grootte van 256MB. Het formaat van de content is applicatie specifiek.

3.3.2.10 Subscribe

Het abonneren op een topic door een subscriber gebeurd door het zogenaamde SUBSCRIBE/SUBACK packet pair.

3.3.2.11 Unsubscribe

Het opzeggen van een abonnement op een topic door een subscriber gebeurd door het zogenaamde UNSUBSCRIBE/UNSUBACK packet pair.

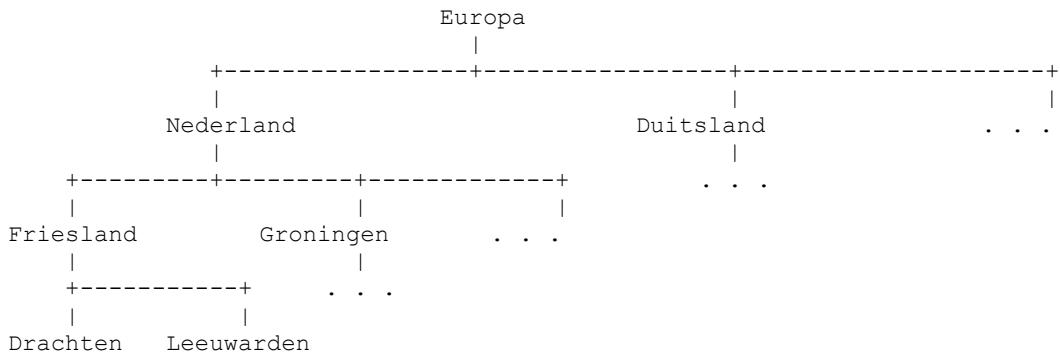
3.3.2.12 Ping

De zogenaamde PINGREQ/PINGRESP packet sequence wordt gebruikt om een zogenaamde ping naar de broker server te sturen. Het wordt gebruikt om de verbinding tussen broker, publishers en subscribers “levend” te houden. Het dient ook om te voorkomen dat een TCP-verbinding door een gateway of router wordt gesloten.

Part 1: The 35 euro IoT project

3.3.3 Topic strings

Het abonnement op een topic wordt vastgelegd in een zogenaamde topic string. Over het algemeen is de representatie van een boomstructuur die meerdere topics bevat (topic tree). Een voorbeeld van een topic tree:



Een topic string maakt gebruik van het speciale scheidingskarakter /.

Valide topic strings uit bovenstaande voorbeeld zijn:

/Europa
/Europa/Nederland
/Europa/Nederland/Friesland
/Europa/Nederland/Friesland/Drachten

Een cliënt kan zich abonneren op alle takken in de boom (/Europa) of op een deel van de boom (/Europa/Nederland/Friesland).

Er zijn ook een tweetal wildcard karakters te gebruiken:

- Enkelvoudige wildcard: +
- Meervoudige wildcard: #
- Uitsluitingskarakter: \$

3.3.3.1 Enkelvoudige wildcard

Met een + kan een enkelvoudig karakter als wildcard gebruikt worden.

Voorbeeld topic strings:

haas
haan

haa+ abonneert een cliënt op beide bovenstaande topics.

Part 1: The 35 euro IoT project

3.3.3.2 Meervoudige wildcard

Met een # kunnen meerdere karakters als wildcard gebruikt worden.

Voorbeeld topic strings:

Nederland

Duitsland

Frankrijk

#land abonneert een cliënt op de topics Nederland en Duitsland.

abonneert de cliënt op alle bovenstaande topics.

3.3.3.3 Uitsluitingskarakter

Het speciale karakter \$ dient om bepaalde topics richting de cliënt uit te sluiten. Het wordt speciaal gebruikt voor transport server specifieke berichten of voor systeem meldingen.

Overzicht MQTT berichten

| MQTT Message | Description |
|--------------|-------------------------------------|
| CONNECT | Client request to connect to server |
| CONNACK | Connect acknowledgment |
| PUBLISH | Publish message |
| PUBACK | Publish acknowledgment |
| PUBREC | Publish received |
| PUBREL | Publish release |
| PUBCOMP | Publish complete |
| SUBSCRIBE | Client subscribe request |
| SUBACK | Subscribe acknowledgment |
| UNSUBSCRIBE | Unsubscribe request |
| UNSUBACK | Unsubscribe acknowledgment |
| PINGREQ | PING request |
| PINGRESP | PING response |
| DISCONNECT | Client is disconnecting |

3.3.4 Termination

Wanneer een publisher of subscriber de MQTT sessie wil beëindigen stuurt deze een DISCONNECT bericht naar de broker en sluit daarna de connectie. We spreken van een graceful shutdown en het maakt het mogelijk dat een cliënt eventueel weer gemakkelijk de verbinding kan herstellen. Indien de verbinding met een publisher wegvalt zonder dat er een DISCONNECT bericht is gestuurd dan kan de broker eventueel een bericht naar de subscribers sturen. In de bericht staan dan instructies over de acties die een subscriber moet uitvoeren indien de verbinding met een publisher plotseling wegvalt. Dit bericht wordt een *last will and testament* genoemd.

Bron

Vrij vertaald uit: <http://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport>

Part 1: The 35 euro IoT project

3.3.5 MQTT implementaties

In dit boek worden de volgende MQTT-implementaties gebruik:

- Mosca als MQTT-broker.

Mosca wordt gebruikt in combinatie met NodeJS om de AngularJS applicatie te kunnen draaien. Maar is ook de broker voor Android app.

- Mosquitto als MQTT-broker en cliënt.

Mosquitto wordt gebruikt om de werking van MQTT te demonstreren. Daarnaast wordt er een publisher gebruikt om de werking van de AngularJS applicatie te testen (los van de hardware implementatie met de ESP-12E).

- Paho als MQTT Java cliënt.

In de te ontwikkelen Android app wordt Paho gebruikt als MQTT-cliënt. Paho is beschikbaar voor meerdere platformen. Naast Java ook voor C/C++, Python, Java Script, C# en Arduino.

Part 1: The 35 euro IoT project

3.4 NTP



Het Network Time Protocol of Netwerktijdprotocol (NTP) is een protocol waarmee computers die onderling met elkaar in verbinding staan, hun interne klok kunnen synchroniseren met andere computers.

3.4.1 Techniek

NTP is gebaseerd op de aannname dat de netwerkvertraging voorspelbaar is. Het computernetwerk wordt hiërarchisch ingedeeld, waarbij de computer met de nauwkeurigste tijd bron wordt aangeduid met "stratum 1". De computersystemen die via NTP daar de tijd van afhalen, zijn per definitie "stratum 2".

Het protocol beschikt over enkele intelligente functies. Zo kan een NTP-cliënt van meerdere NTP-bronnen gebruikmaken, waarbij de NTP-cliënt zelf uitzoekt welke bron het beste werkt. Aan de hand van een aantal beslisingscriteria kiest zo'n NTP-cliënt een bron en synchroniseert zich daaraan. Kleine verschillen in tijd tussen bron en cliënt worden door de cliënt bijgewerkt door de tijds-verwerking op de cliënt computer iets sneller of langzamer te laten werken. Daardoor kan zonder sprongen in tijd het verschil worden weggewerkt. De cliënt blijft echter alle NTP-bronnen in de gaten houden en kiest een andere bron voor synchronisatie zodra die een stabielere tijd biedt.

Als tijd bron kan bijvoorbeeld een atoomklok dienen of een DCF77- of gps-ontvanger.

3.4.2 Nederland

In Nederland kan gebruik worden gemaakt van een pool van NTP-servers onder de naam nl.pool.ntp.org.

Deze pool bestaat uit de volgende 4 servers:

- 0.nl.pool.ntp.org
- 1.nl.pool.ntp.org
- 2.nl.pool.ntp.org
- 3.nl.pool.ntp.org

In onze project maken we gebruik van de pool server nl.pool.ntp.org.

NTP-servers worden drukbezocht door en daarom kan het even duren voor er een echte update binnenkomt. Hiermee wordt in het project rekening gehouden.

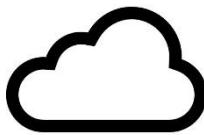
Bron

Vrij vertaald naar: https://nl.wikipedia.org/wiki/Network_Time_Protocol

Part 1: The 35 euro IoT project

3.5 The Cloud

3.5.1 Inleiding



Cloud computing is het via een netwerk – vaak het internet – op aanvraag beschikbaar stellen van hardware, software en gegevens, ongeveer zoals elektriciteit uit het lichtnet. De term is afkomstig uit de schematechnieken uit de informatica, waar een groot, decentraal netwerk (zoals het internet) met behulp van een wolk wordt aangeduid.

De Cloud (Nederlands: wolk) staat voor een netwerk dat met al de computers die erop aangesloten zijn een soort 'wolk van computers' vormt, waarbij de eindgebruiker niet weet op hoeveel of welke computer(s) de software draait of waar die computers precies staan. De gebruiker hoeft op deze manier geen eigenaar meer te zijn van de gebruikte hard- en software en is dus ook niet verantwoordelijk voor het onderhoud. De details van de informatietechnologische infrastructuur worden aan het oog onttrokken en de gebruiker beschikt over een "eigen", in omvang en mogelijkheden schaalbare, virtuele infrastructuur. De Cloud is dus een techniek waarmee schaalbare onlinediensten kunnen worden aangeboden. Zonder de mogelijkheid tot schalen heeft een aangeboden onlinedienst geen betrekking op Cloud computing.

In dit boek zal er feitelijk geen sprake van Cloud computing. Wel wordt de data opgeslagen in een database die via het netwerk benaderbaar is. Daarmee wordt een soort van Cloud simulatie gerealiseerd.

Bron

Vrij naar: https://nl.wikipedia.org/wiki/Cloud_computing

3.5.2 MongoDB



MongoDB is een open source document georiënteerde database. Het behoort tot de zogenaamde NoSQL database. Zo kent MongoDB geen vaste database schema's. MongoDB is hierin uitermate flexibel.

De database kan gemakkelijk gedistribueerd worden en de data wordt dan over meerdere computers verspreid om gedistribueerde gegevensverwerking mogelijk te maken. MongoDB is geen relationeel databasemanagementsysteem (RDBMS) en er is geen ondersteuning voor joins. Ook is de ondersteuning voor transacties beperkt.

Documenten worden opgeslagen in het zogenaamde BSON (Binaire JSON) formaat. Het is uitermate geschikt voor extreem grote databases. MongoDB is dan ook afgeleid van het Amerikaanse woord humongous wat extreem groot betekent.

In dit boek wordt MongoDB gebruikt om de dataopslag in de Cloud te simuleren.

Bron

Vrij naar: <https://nl.wikipedia.org/wiki/MongoDB>

4 Boodschappenlijstje

4.1 Inleiding

Het project bestaat uit zowel een aantal hardware als softwarecomponenten. De onderdelen zijn over het algemeen goed beschikbaar. Door deze in China te bestellen wordt het project niet al te duur. Zoals ik het nu bereken zou het totale project ongeveer €35,00 aan hardware kosten (stand mei 2016). De totale kosten zijn voor een deel wel afhankelijk van de dollarkoers.

4.2 Hardware

4.2.1 Kostenplaatje

De totale kosten van de hardware zijn ongeveer 35 euro indien de hardware in China wordt besteld.

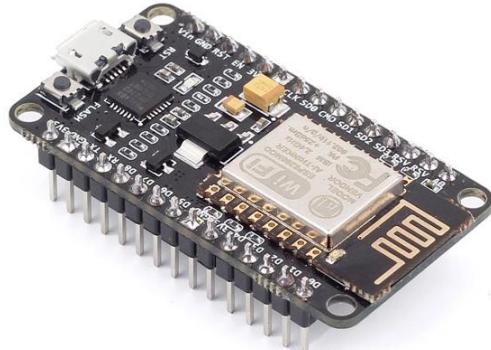
| Hardware | Kosten |
|-------------------------|---------|
| ESP-12E Development Kit | € 6,00 |
| DHT22 | € 4,00 |
| DS1307 RTC | € 2,00 |
| Level Shifter | € 1,00 |
| Breadboard | € 4,00 |
| Breadboard voeding | € 3,00 |
| Jumper Cables | € 3,00 |
| Weerstanden | € 5,00 |
| Voedingadapter 9 Volt | € 4,00 |
| USB-micro kabel | € 3,00 |
| Totaal | € 35,00 |

Indien besteld in Europa of Nederland zullen de totalen kosten rond de 50 euro liggen.

Part 1: The 35 euro IoT project

4.2.2 ESP-12E Development Kit

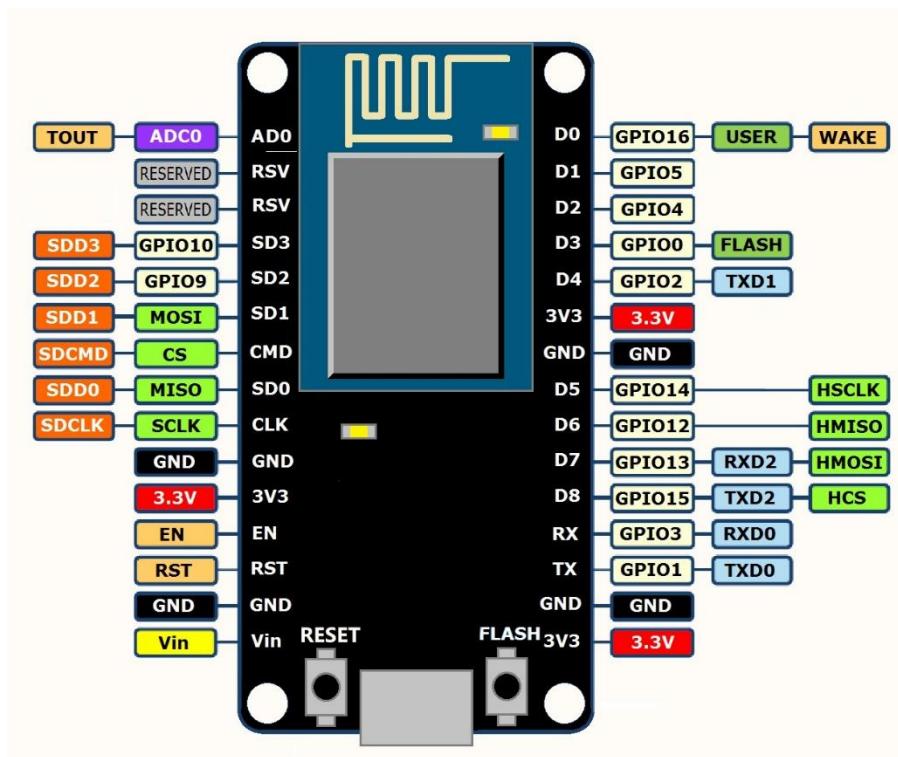
De ESP-12E bevat een goedkope ESP8266 Wi-Fi Chip en heeft daarnaast meerdere digitale pinnen en één enkele analoge pin.



Een ESP-12E indien in China besteld rond de 6 euro.

Let op: gebruik de versie met NodeMCU v1.0 of hoger.

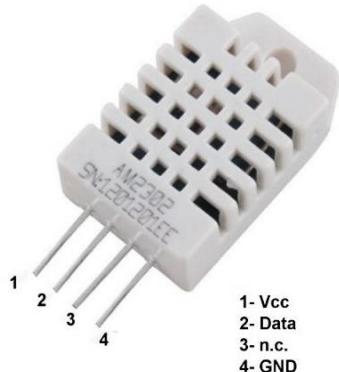
Pin lay-out ESP-12E Development Kit:



Part 1: The 35 euro IoT project

4.2.3 DHT22 Sensor

De DHT22 is een digitale temperatuur en luchtvochtigheidssensor.



Het temperatuur bereik ligt tussen de -40 en 80 °C met een nauwkeurigheid van -/+0,5°C.

Het vochtigheid bereik ligt tussen de 0 en 100% met en precisie van -/+ 2%.

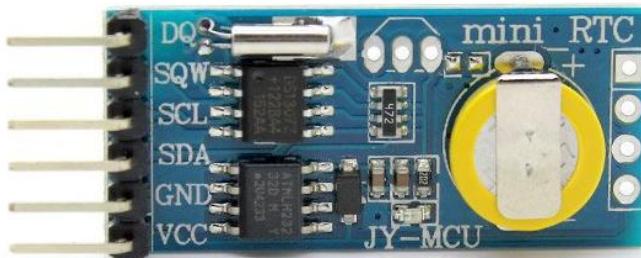
Een DHT22 kost indien in China besteld rond de 4 euro.

Als alternatief kan de iets minder nauwkeurige en goedkopere DHT11 gebruikt worden.



4.2.4 DS1307 RTC

Een DS1307 RTC (Real Time Clock) wordt gebruikt als klok in het project. Er zijn verschillende uitvoeringen. In dit project wordt de RTC uit de afbeelding gebruikt. Voor gebruik is een level shifter nodig omdat deze RTC op 5 Volt werkt.



Een DS1307 kost indien in China besteld rond de 2 euro.

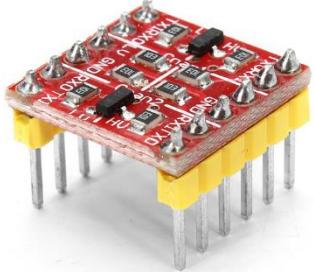
Als alternatief kan de even dure DS3231 gebruikt worden. Voordeel van deze laatste is dat er geen level shifter nodig is omdat deze al op 3,3 Volt werkt.



Part 1: The 35 euro IoT project

4.2.5 Level Shifter

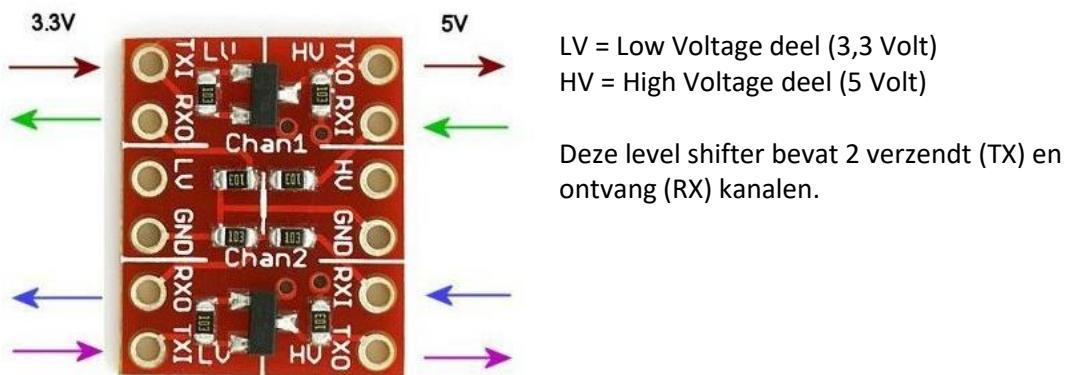
Voor het aansluiten de 5 Volt DS1307 RTC is een level shifter nodig. Hierdoor wordt het mogelijk het 5 Volt signaal om te zetten naar een 3,3 Volt signaal.



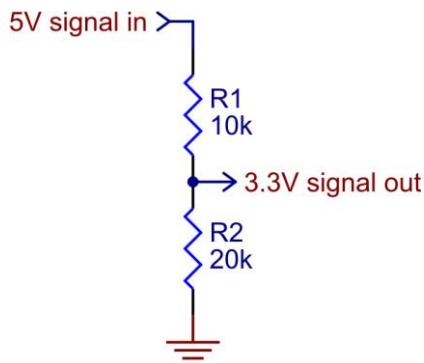
Officieel is dit een zogenaamde 3.3V 5V TTL Bi-directional Logic Level Converter.

Dit component kost indien besteld in China ongeveer 1 euro.

Schema:

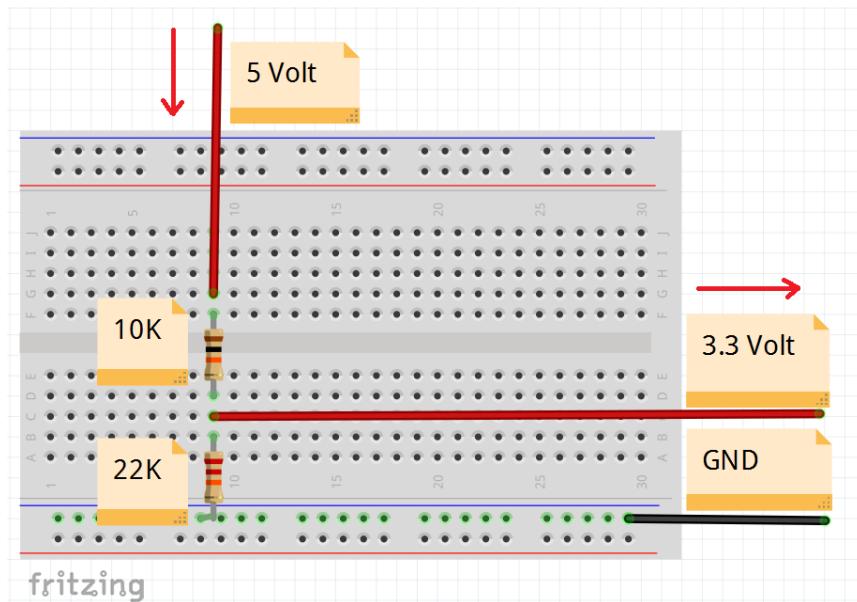


Alternatief voor een level shifter is een zogenaamde voltage divider. Hierbij worden 2 weerstanden gebruikt van bijv. 10 en 22K.



Part 1: The 35 euro IoT project

Dit is eenvoudig op een breadboard te realiseren:



4.2.6 Arduino power adapter



Arduino 9 Volt power adapter (1A). Let erop dat er gebruik wordt gemaakt van en 2,1mm plug. De pin moet wel center positive zijn (buitenkant = GND, binnen stekker = 9V).



Indien besteld in China ongeveer 4 euro.

4.2.7 USB micro kabel

USB-kabel met micro aansluiting.



Indien besteld in China ongeveer 3 euro.

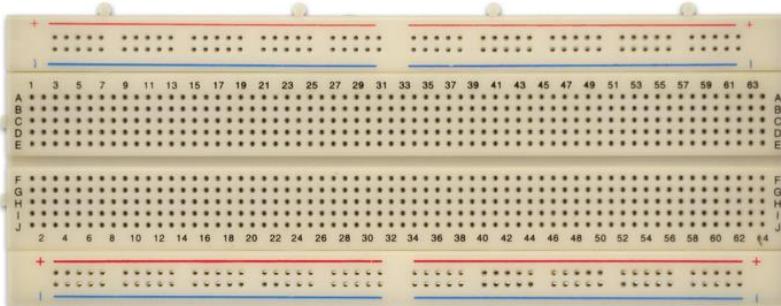
Part 1: The 35 euro IoT project

4.2.8 Breadboard en overige benodigdheden

Naast de genoemde componenten zijn de volgende componenten nodig:

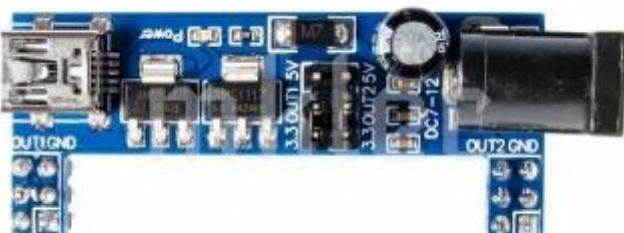
- Breadboard:

Breadboards zijn er in verschillende afmetingen. Het breadboard op de afbeelding kost indien besteld in China ongeveer 4 euro.



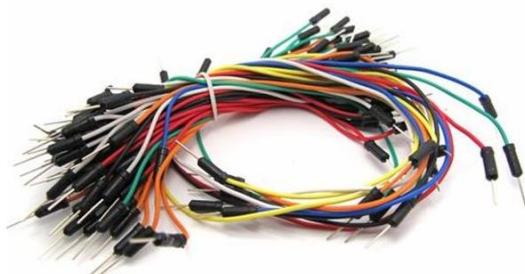
- Breadboard voeding (3,3 / 5 Volt):

Een breadboard voeding past op een breadboard. Er kan geschakeld worden tussen 0, 3,3 en 5 Volt. Een breadboard voeding kost indien besteld in China ongeveer 3 euro.



- Jumper cables:

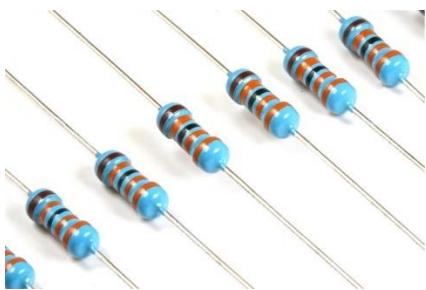
Een setje kabels van 68 stuks kost indien besteld in China ongeveer 3 euro.



- Weerstanden:

Weerstanden worden over het algemeen per set verkocht. Een set van 400 verschillende weerstanden kost indien besteld in China ongeveer 5 euro.

Part 1: The 35 euro IoT project

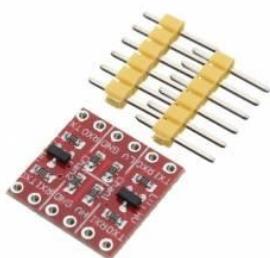


Tip: het is meestal voordeliger om een set van een breadboard, voeding, jumper cables en/of weerstanden te kopen.

4.2.9 Kopen in China

Het bestellen van electronica in China is vaak voordelijker omdat er over het algemeen geen verzendkosten in rekening worden gebracht. Er zijn wel een paar aandachtspunten:

- Laat het totaalbedrag van de bestelling niet boven de € 22,00 komen. Boven dit bedrag is er een kans dat er alsnog invoerrechten en inklaarkosten (€12,50 ongeacht de waarde van het pakket) moeten worden betaald. Alternatief is het bestellen bij een dependance in de Europese Unie. Dit is vaak wel iets duurder en voor gratis verzenden geldt er vaak een minimum bestelbedrag.
- Hou rekening met lange wachttijden. Het duurt gemiddeld tussen de 3 en 4 weken (soms langer) voordat de goederen afgeleverd worden. Positief hierbij is dat ik het persoonlijk nog niet heb meegemaakt dat het bestelde niet werd geleverd.
- Er is een kans dat de goederen beschadigd worden afgeleverd. Dit geldt bijvoorbeeld voor verbogen pinnetjes. Zelf heb ik dat nog maar één keer meegemaakt, maar een waarschuwing is op z'n plaats denk ik.
- Als solderen een probleem is kijk dan of de componenten geassembleerd worden afgeleverd. Voorbeeld zijn de level shifters die in twee varianten geleverd worden.



Niet geassembleerd

Geassembleerd

Voorbeelden van Chinees websites zijn:

- www.dx.com
- www.banggood.com
- www.miniinthebox.com/nl/

Europese dependances:

- eud.dx.com
- eu.banggood.com

Part 1: The 35 euro IoT project

4.2.10 Kopen in Nederland of de EU

Het bestellen van hardware in Nederland of de EU is over het algemeen een stuk duurder dan in China. Grote voordeel is wel dat de bestellingen niet lang onderweg zijn. Bij veel websites geldt tegenwoordig dat bestellingen de volgende (werk)dag al geleverd worden.

Een ander groot voordeel is dat er geen invoerrechten en inklaarkosten betaald hoeven te worden. Vaak moeten er beneden een bepaald minimum bestelbedrag wel verzendkosten worden betaald. Voorwaarde voor gratis verzending is vaak wel een betaling via iDeal.

Een nadeel kan wel zijn dat websites niet alle onderdelen kunnen leveren of op voorraad hebben. Het is dus soms even zoeken. Ook kan prijsvergelijken de nodige euro's opleveren.

Bekende Nederlandse electronica websites zijn:

- www.conrad.nl (gratis verzending vanaf € 35,00).
- www.iprototype.nl (gratis verzending vanaf € 75,00).
- www.sossolutions.nl (gratis verzending, maar vooral gericht op de Raspberry Pi).
- www.tinytronics.nl (verzendkosten € 2,50).
- www.benselectronics.nl (verzendkosten € 3,95).
- www.microkloon.nl (verzendkosten € 2,19).

Alternatief is het zoeken naar tweedehands onderdelen op bijvoorbeeld:

- www.marktplaats.nl
- www.tweedehands.nl
- www.tweedehandscomputers.nl

Verzendkosten zijn hier vooraf niet aan te geven. Vaak is het in deze gevallen ook beter om de spullen zelf af te halen.

Part 1: The 35 euro IoT project

4.3 Software

4.3.1 Inleiding

Voor het uitvoeren en testen van het project zijn een aantal softwareinstallaties nodig. Het betreft hier allemaal open source of freewaresoftware.

Benodigde software:

- Voor het programmeren van de ESP-12E wordt de Arduino IDE gebruikt.
- Voor AngularJS worden een aantal JavaScript oplossingen gebruikt:
 - AngularJS
 - AngularJS Canvas Gauge
 - AngularJS Resource
 - Gauge
- Voor het uitvoeren van de AngularJS scripts wordt NodeJS gebruikt.
- Voor de MQTT broker en cliënt-computers worden Mosquitto, Mosca en Paho gebruikt.
- Voor het testen van een MQTT publisher met Paho wordt Eclipse met Java gebruikt.
- Voor het maken van de Androidoplossing wordt de Android Studio gebruikt.
- Voor het maken van grafieken wordt Chart.js gebruikt.
- Optioneel kan Virtual Router gebruikt worden als oplossing voor het gemis aan Wi-Fi.

In de volgende paragrafen worden de installaties van de diverse softwarecomponenten beschreven. Elke paragraaf wordt indien van toepassing afgesloten met een simpele voorbeeld implementatie.

De gebruikte versies van de softwarepakketten zijn van mei of juni 2016. Het is inmiddels goed mogelijk dat er hogere softwareversies beschikbaar zijn. Over het algemeen is het verstandig om de laatste versie van een bepaald softwarepakket te installeren.

Part 1: The 35 euro IoT project

4.3.2 Arduino IDE

4.3.2.1 Installatie

Open een browser en ga naar: <https://www.arduino.cc/en/Main/Software>

Download the Arduino Software



Download de Windows installer. Kies “JUST DOWNLOAD” (doneren mag natuurlijk ook!).

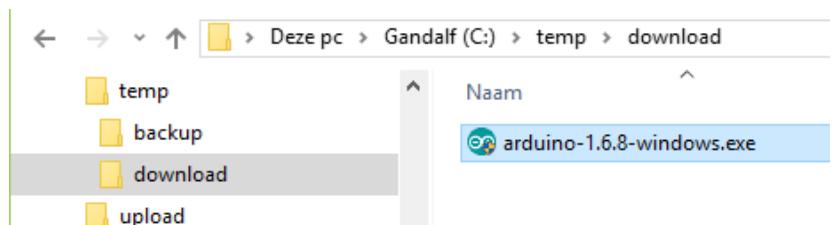
Support the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). [Learn more on how your contribution will be used.](#)

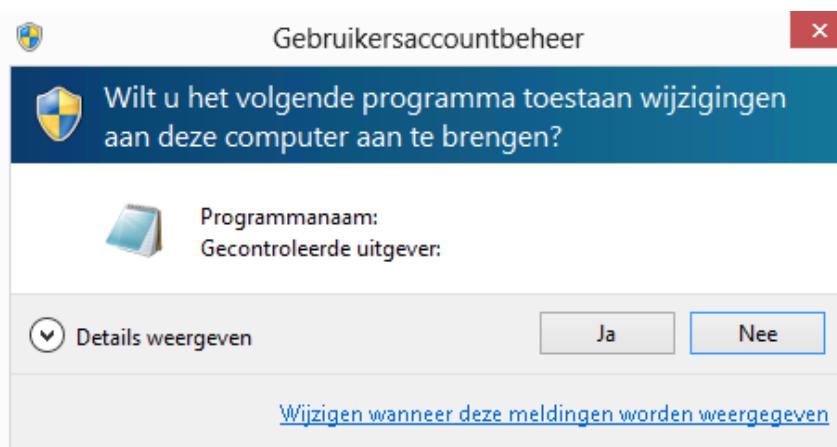


Part 1: The 35 euro IoT project

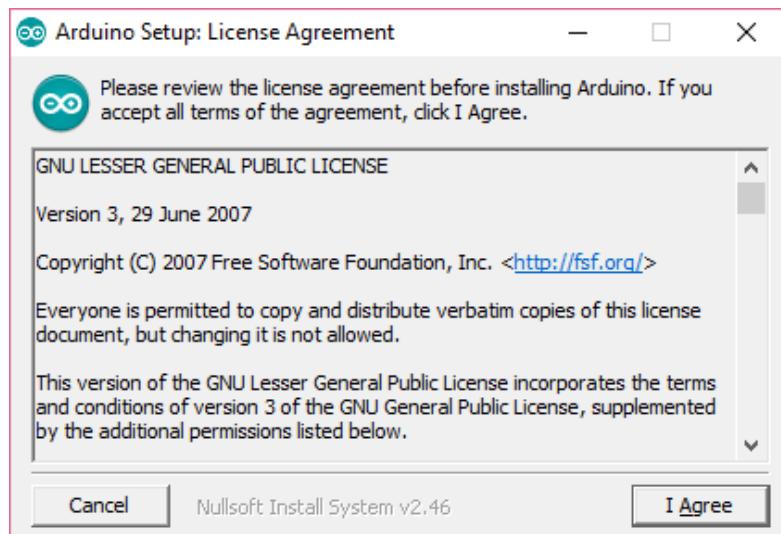
Dubbelklik met de muis op de installer.



→ Ja

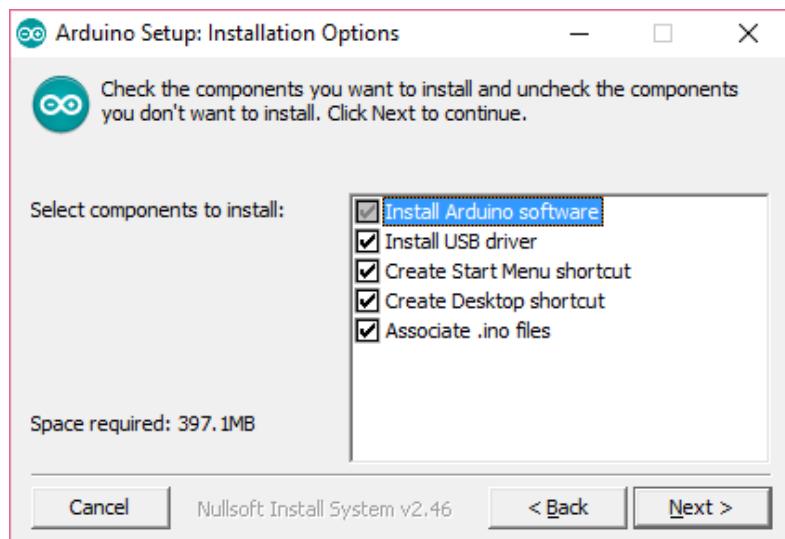


→ I Agree

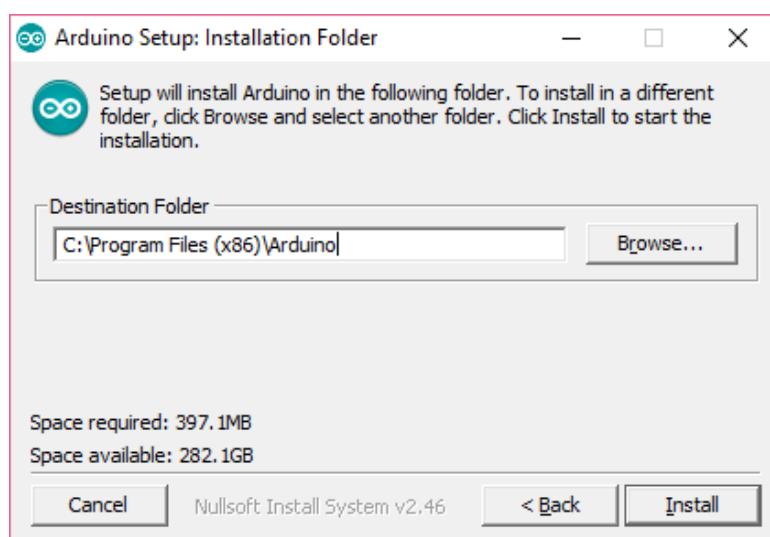


Part 1: The 35 euro IoT project

→ Next

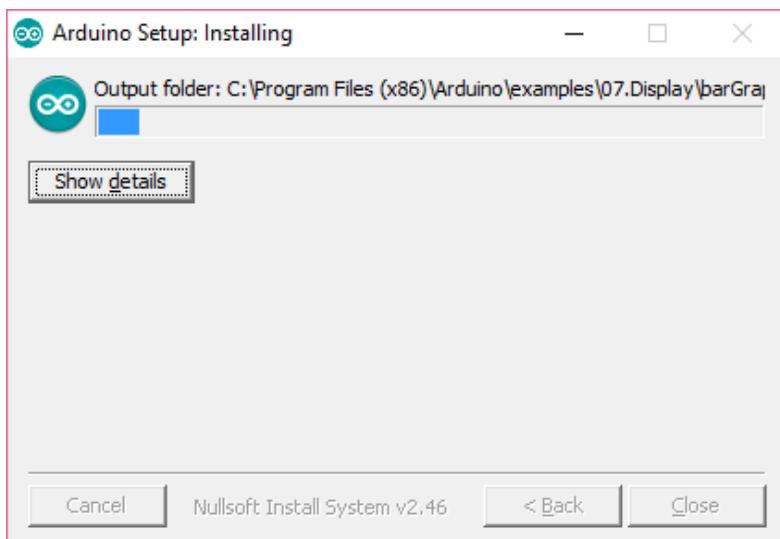


→ Install

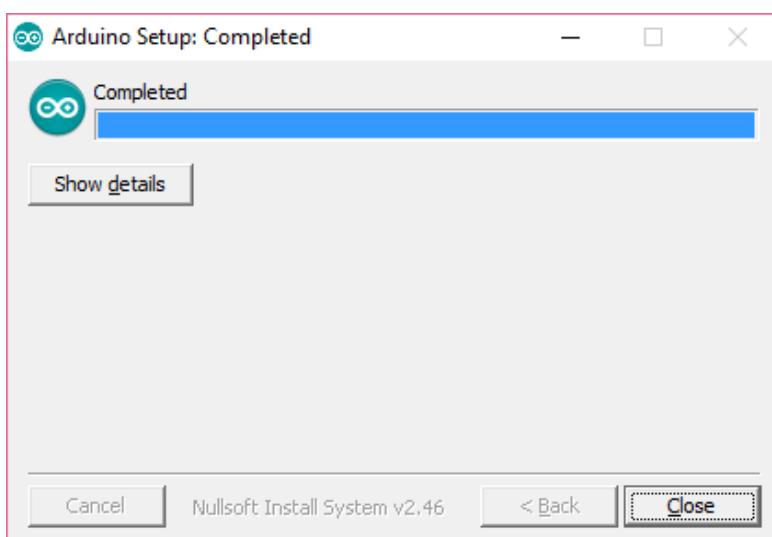


Part 1: The 35 euro IoT project

Even geduld...



→ Close



Start Arduino via icoontje op de desktop

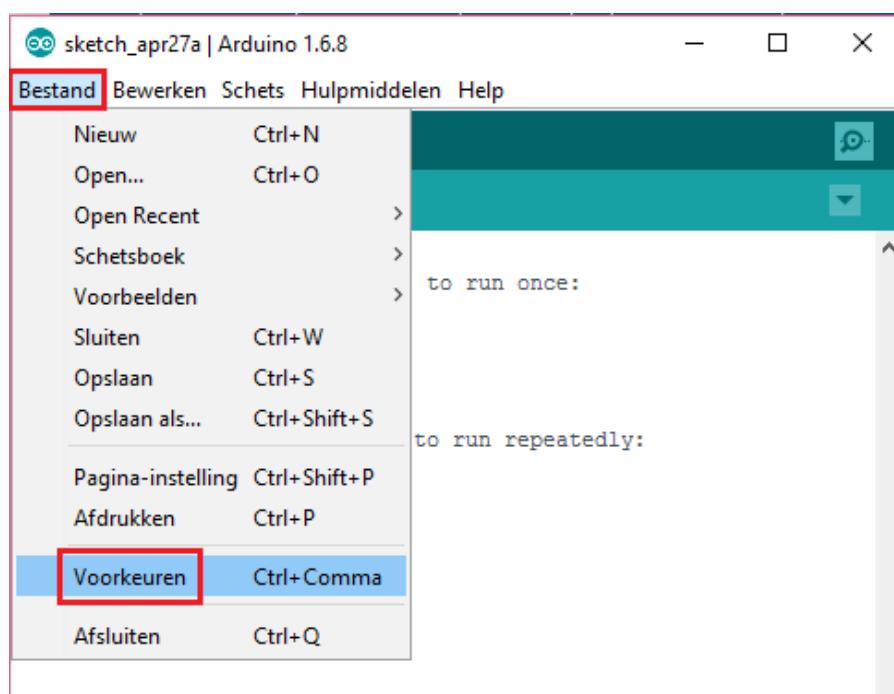


Part 1: The 35 euro IoT project

Even geduld...



Menu → Bestand → Voorkeuren



Additionele Bordensbeheerder URL's:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

- OK

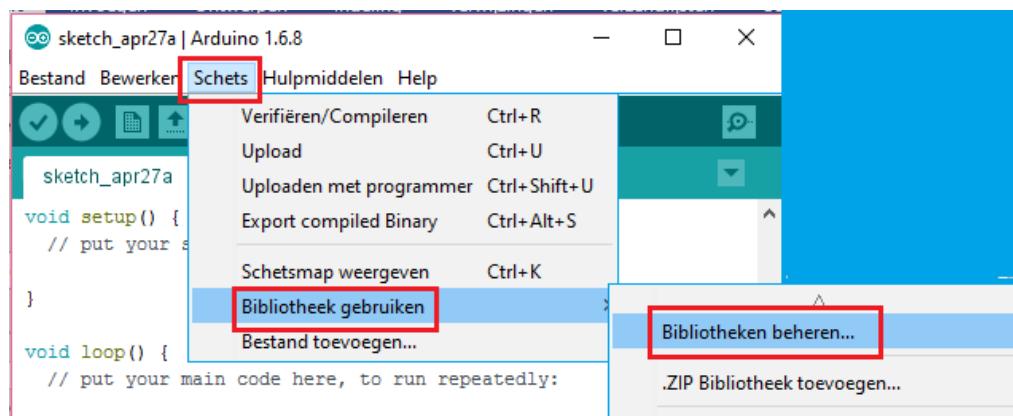
A screenshot of the "Voorkeuren" (Preferences) dialog box. It contains a checkbox labeled "Opslaan bij het verifiëren of uploaden" which is checked. Below it is a text input field labeled "Additionele Bordensbeheerder URLs" containing the URL "http://arduino.esp8266.com/stable/package_esp8266com_index.json". To the right of the input field is a small "OK" button. At the bottom of the dialog box is the text "Meer voorkeuren kunnen rechtstreeks in het bestand worden bewerkt".

Part 1: The 35 euro IoT project

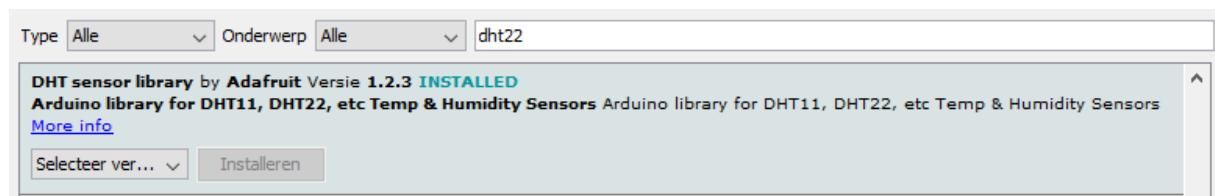
4.3.2.2 Arduino Bibliotheken

Voor dit project zijn een aantal extra bibliotheken nodig. Deze zijn te installeren via de bibliotheek beheerder.

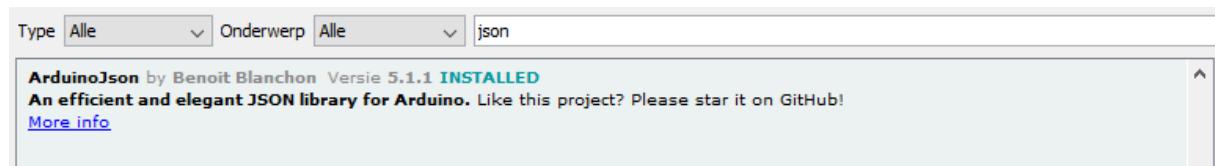
Menu → Schets → Bibliotheek gebruiken → Bibliotheeken beheren...



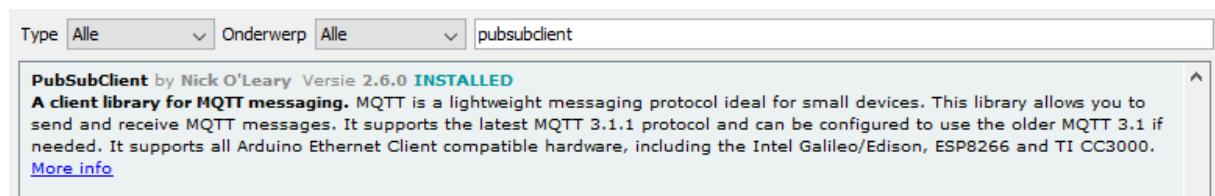
Installeer de volgende extra bibliotheken: dht22



ArduinoJSON



PubSubClient



Part 1: The 35 euro IoT project

RTClib

The screenshot shows a search results page for 'rtcilib' in the Adafruit library manager. The results list one item: 'RTClib by Adafruit Versie 1.1.0 INSTALLED'. Below the title, it says 'A fork of Jeelab's fantastic RTC library' and 'A fork of Jeelab's fantastic RTC library'. There is a 'More info' link at the bottom.

NTPClientLib

The screenshot shows a search results page for 'ntpclientlib' in the Adafruit library manager. The results list one item: 'NtpClientLib by German Martin Versie 1.3.2 INSTALLED'. Below the title, it says 'Ntp Client Library Library to get system sync from a NTP server. Based on code from NTP client example. Currently, it works on ESP8266 based boards. I've made it compatible with Arduino boards w Ethernet module but I have not had the opportunity to test it. Please, add an issue to GitHub if you find a bug.' and a 'More info' link at the bottom.

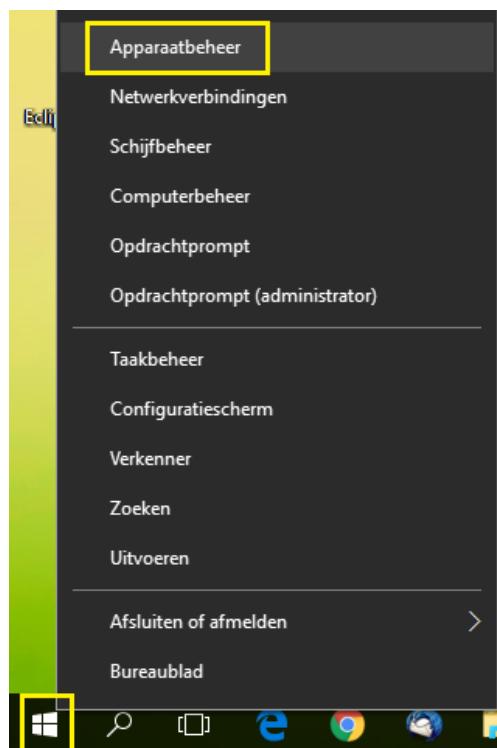
Opmerkingen:

NtpClientLib versie 2.0.x werkt niet, maar versie 1.3.2 wel, dus niet de nieuwste downloaden.

4.3.2.3 Blink

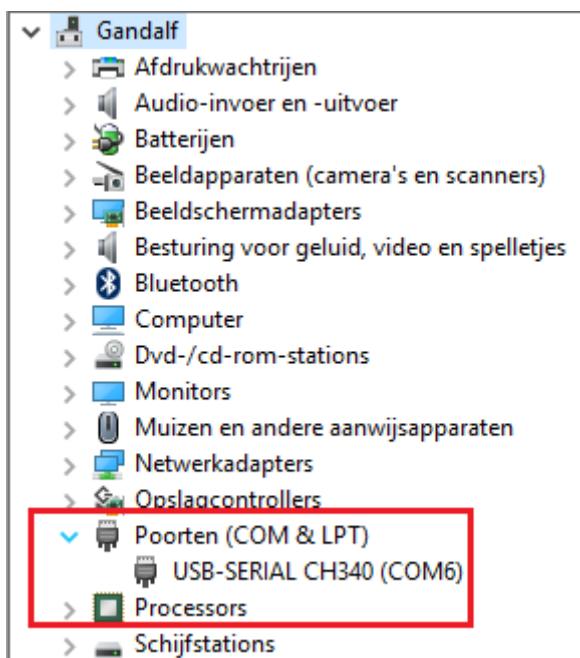
Sluit de ESP-12E aan op een USB-aansluiting op de Windows PC, laptop of tablet. Onder Windows 7 en hoger wordt er automatisch een zogenaamde COM-poort voor de aangesloten ESP-12E geïnstalleerd.

Ga naar Start → rechtermuisknop → Apparaatbeheer.



Part 1: The 35 euro IoT project

Open Poorten (COM & LPT)



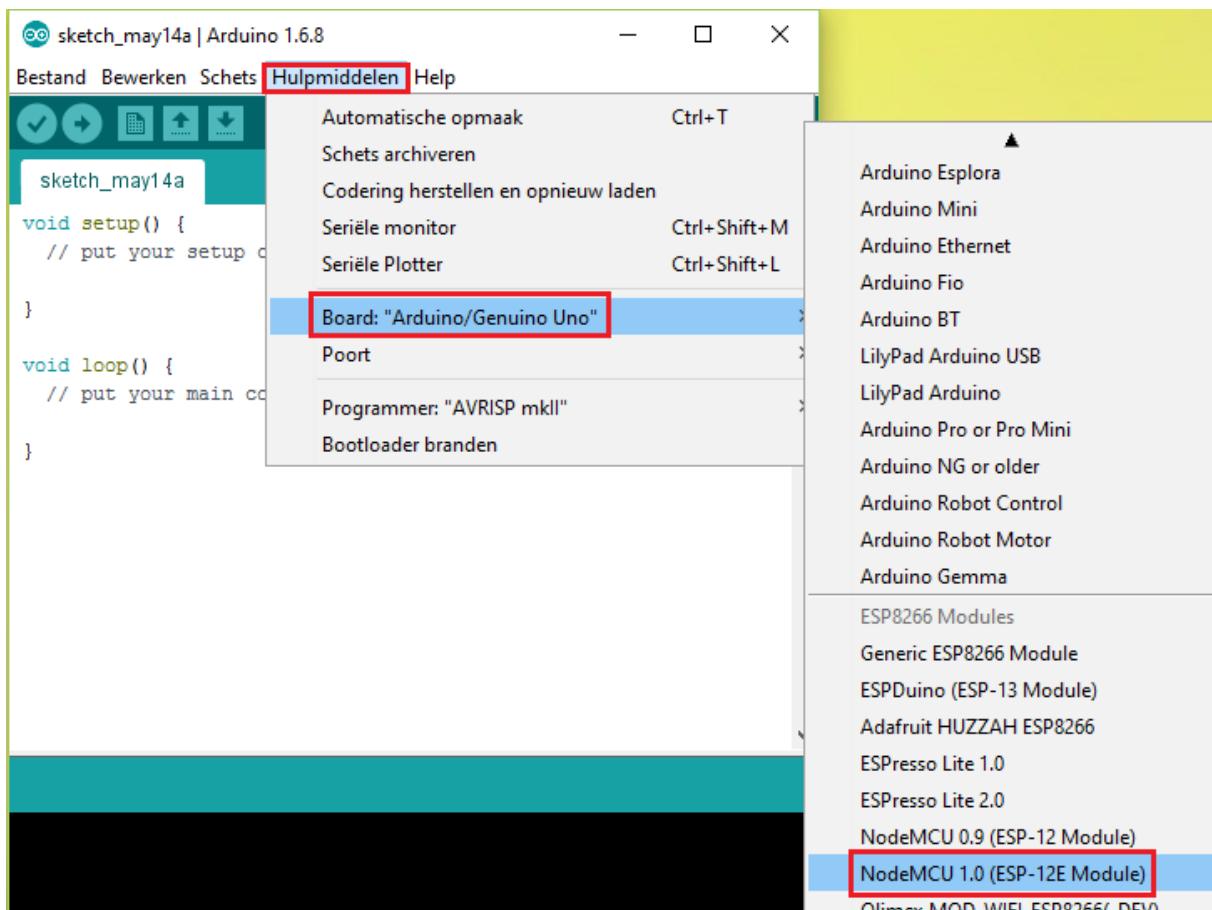
In dit voorbeeld is er een COM6-poort aangemaakt voor de ESP-12E. Het nummer van de aangemaakte COM-poort kan verschillen.

Open de Arduino IDE



Part 1: The 35 euro IoT project

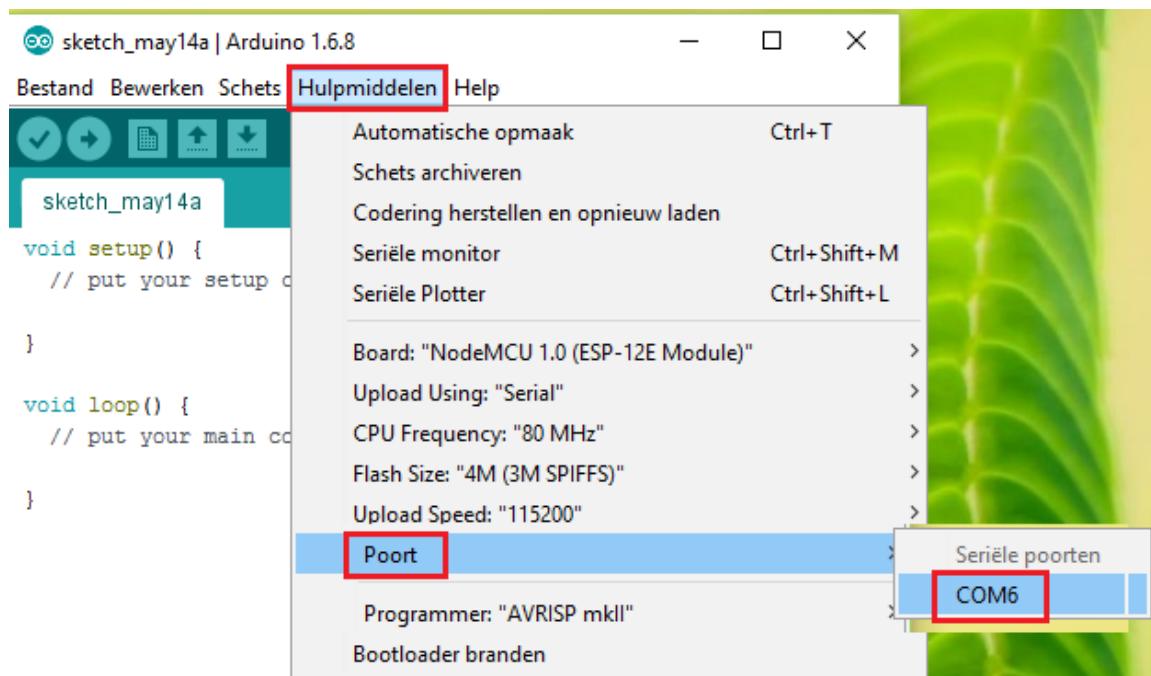
Menu → Hulpmiddelen → Board: ... → NodeMCU 1.0 (ESP-12E Module)



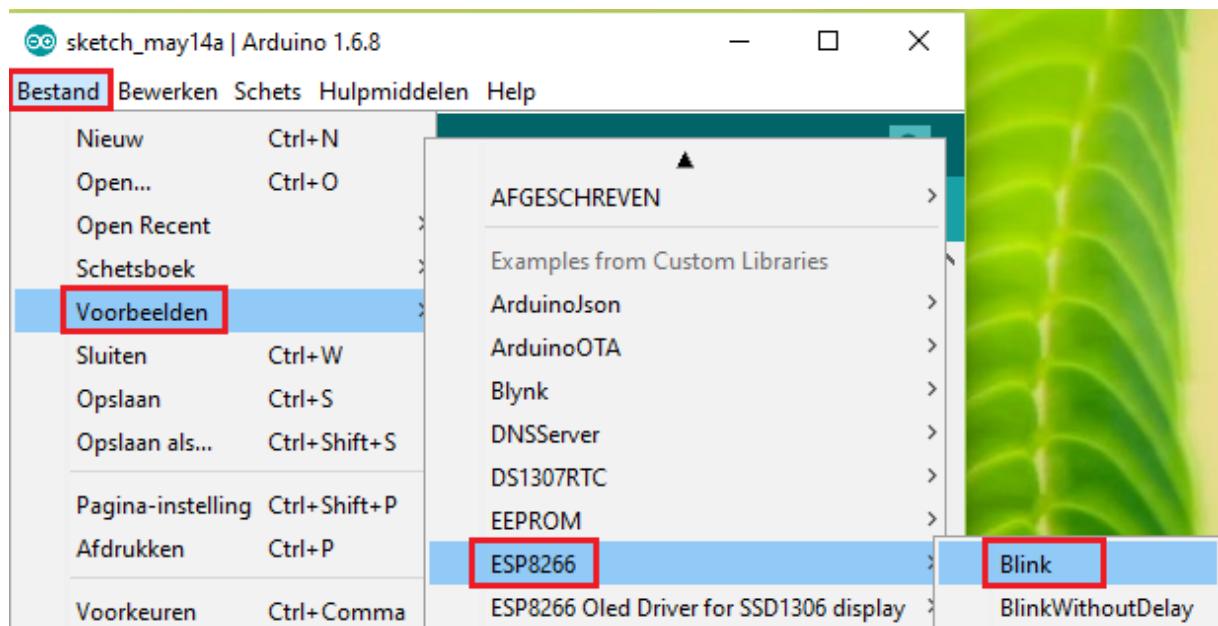
Let op: kies de juiste versie (1.0)!

Part 1: The 35 euro IoT project

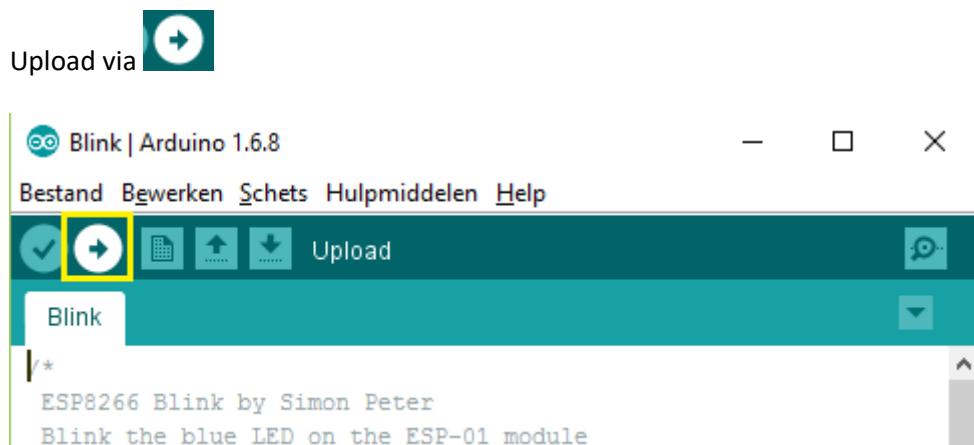
Menu → Hulpmiddelen → Poort → COM6



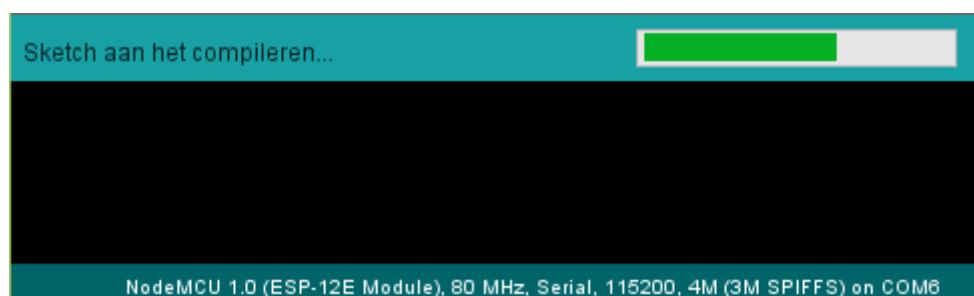
Bestand → Voorbeelden → ESP8266 → Blink



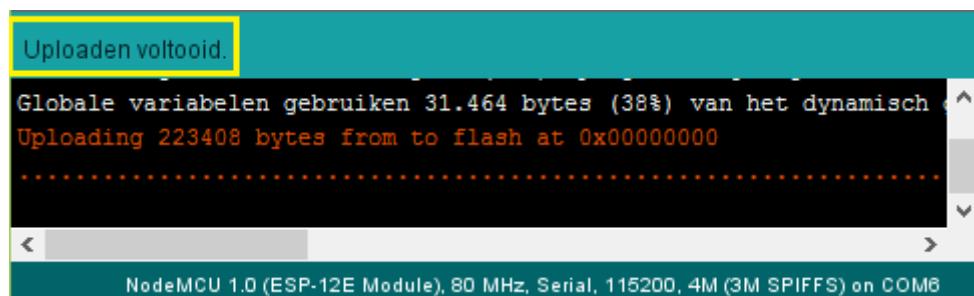
Part 1: The 35 euro IoT project



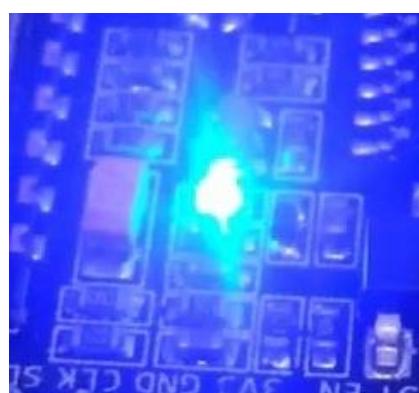
Even geduld...



Klaar:



De blauwe led op de ESP-12E zal nu aan en uit gaan.



Part 1: The 35 euro IoT project

Source:

```
// Setup
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);      // Initialiseer de LED_BUILTIN pin
                                         // als output pin
}

// Eindeloze loop
void loop() {
    digitalWrite(LED_BUILTIN, LOW);    // LED aan
    delay(1000);                     // Wacht 1 seconde
    digitalWrite(LED_BUILTIN, HIGH);   // LED uit
    delay(2000);                     // Wacht 2 seconden
}
```

Zie: Sources → Arduino → Blink → Blink.ino

Op de ESP-12E staat de ingebouwde led default aan (LOW). Daarom wordt de led met LOW aangezet en met HIGH wordt de led uitgezet. Normaal gesproken werkt het net andersom.

Part 1: The 35 euro IoT project

4.3.3 NodeJS

4.3.3.1 NodeJS installatie

Installeren NodeJS

Open een browser en ga naar: <https://nodejs.org/en/download/>

Downloads

Current version: v4.4.3

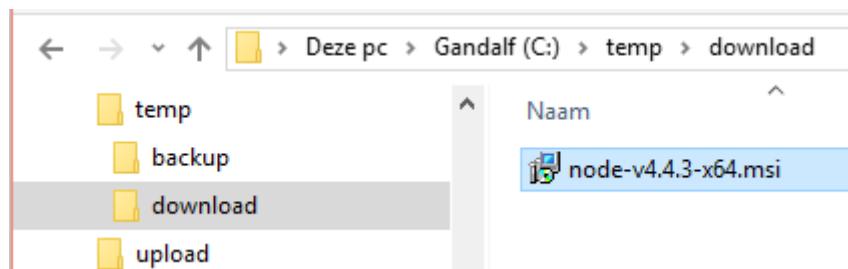
Download the Node.js source code or a pre-built installer for your platform, and start developing today.

| LTS Recommended For Most Users | Windows Installer node-v4.4.3-x64.msi | Macintosh Installer node-v4.4.3.pkg | Source Code node-v4.4.3.tar.gz |
|-----------------------------------|--|--|-----------------------------------|
| Current Latest Features | | | |
| Windows Installer (.msi) | 32-bit | 64-bit | |
| Windows Binary (.exe) | 32-bit | 64-bit | |
| Mac OS X Installer (.pkg) | | 64-bit | |
| Mac OS X Binaries (.tar.gz) | | 64-bit | |
| Linux Binaries (.tar.xz) | 32-bit | 64-bit | |
| Source Code | | | node-v4.4.3.tar.gz |

Selecteer de juiste installer, in dit voorbeeld:

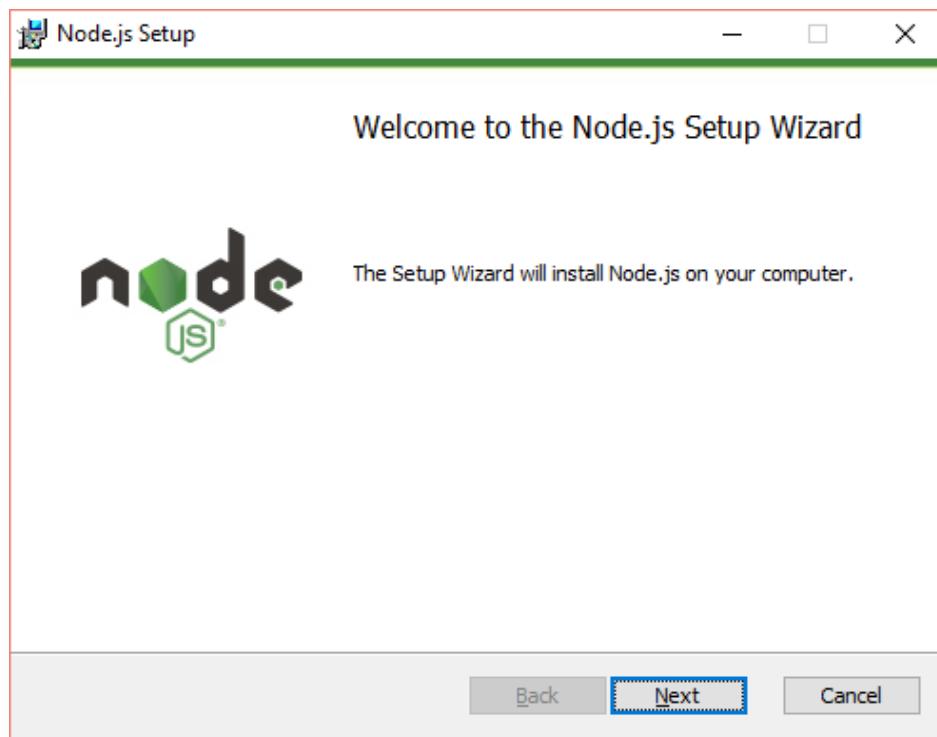


Start gedownloade installer:

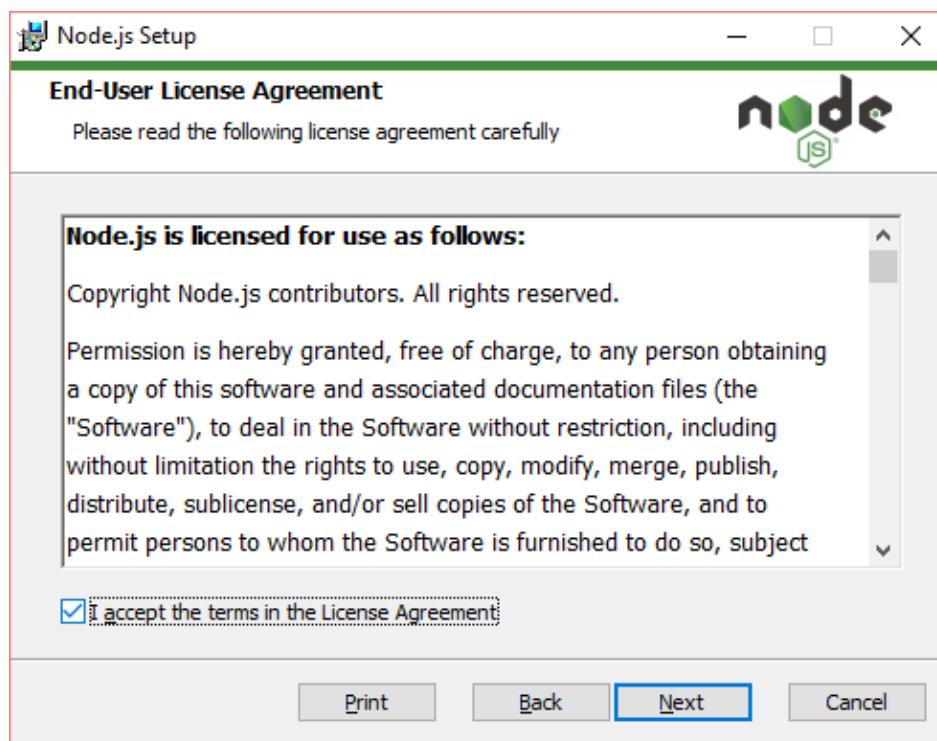


Part 1: The 35 euro IoT project

→ Next

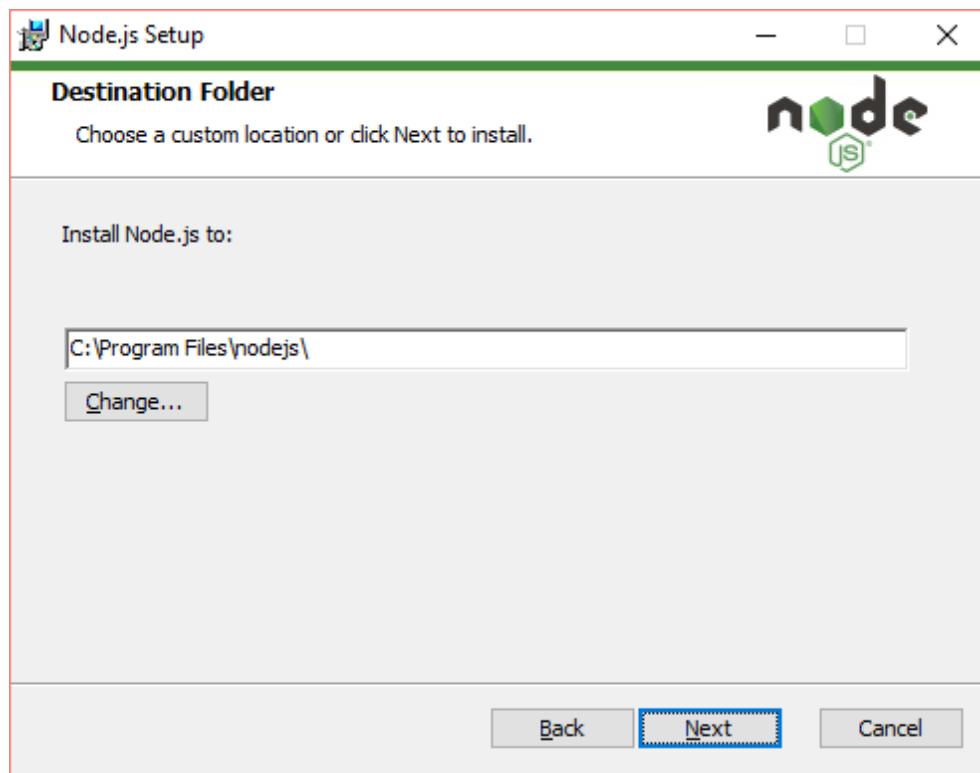


Vink "I accept the terms in the Licence Agreement" aan → Next

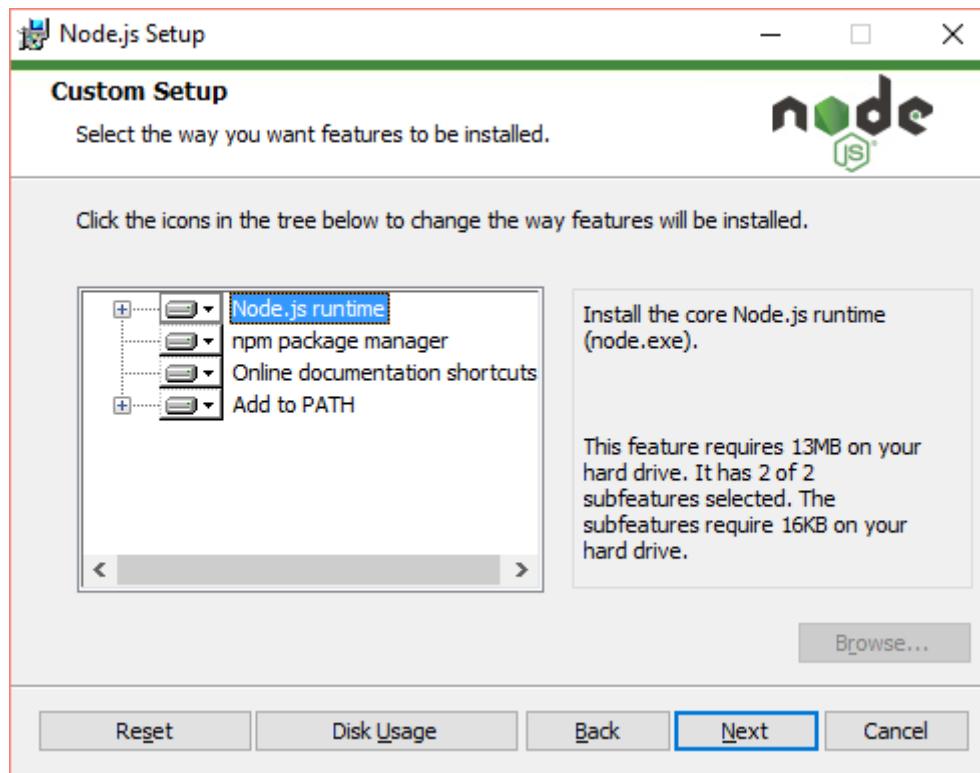


Part 1: The 35 euro IoT project

→ Next

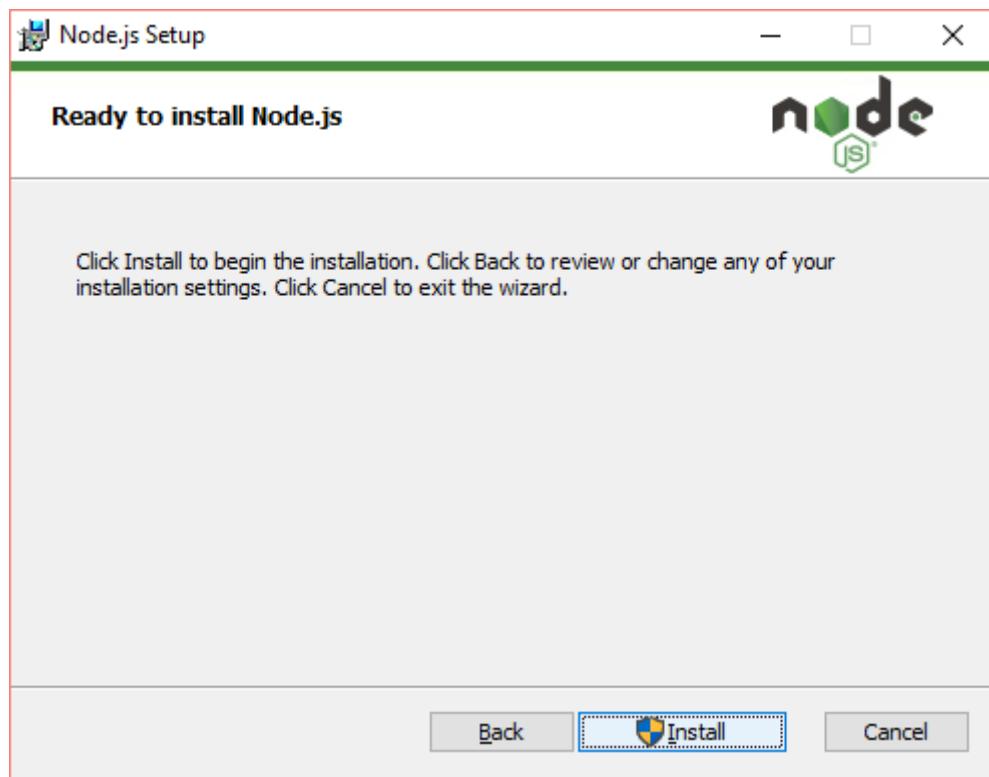


→ Next

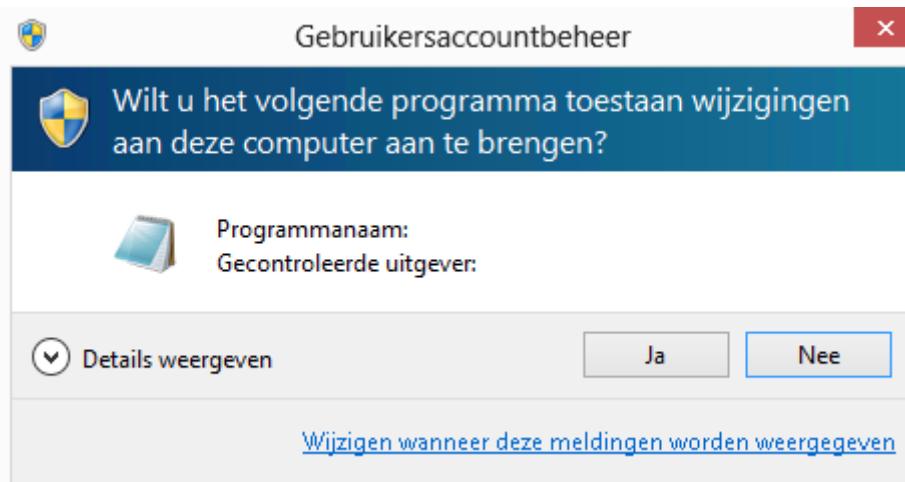


Part 1: The 35 euro IoT project

→ Install

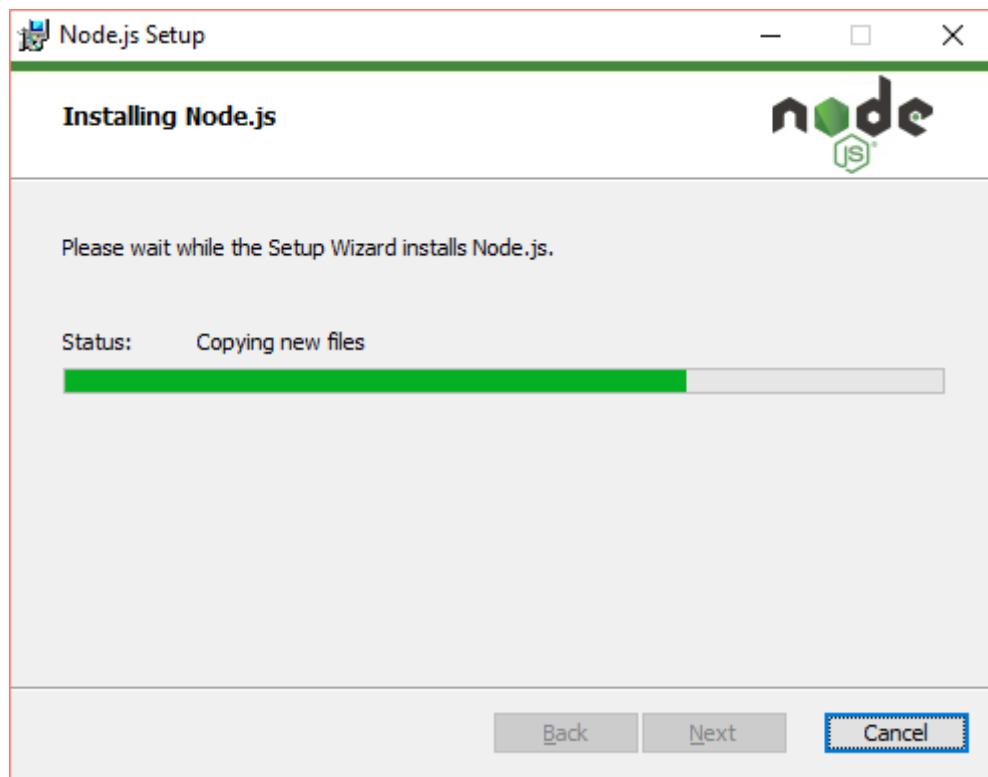


→ Ja

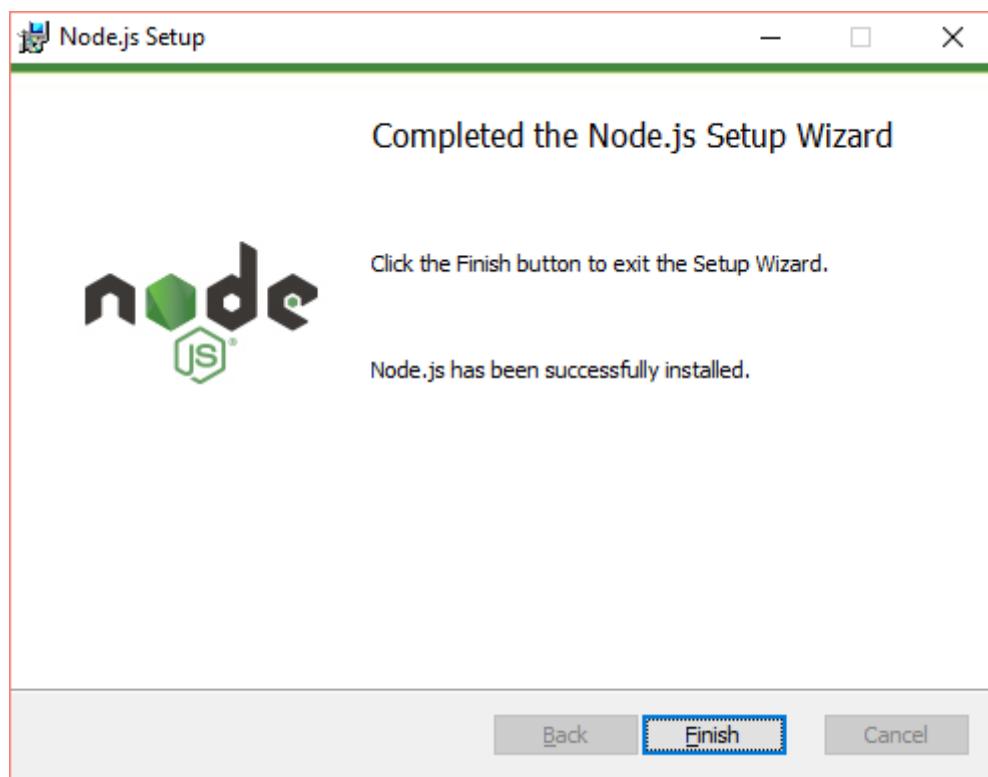


Part 1: The 35 euro IoT project

Even geduld...



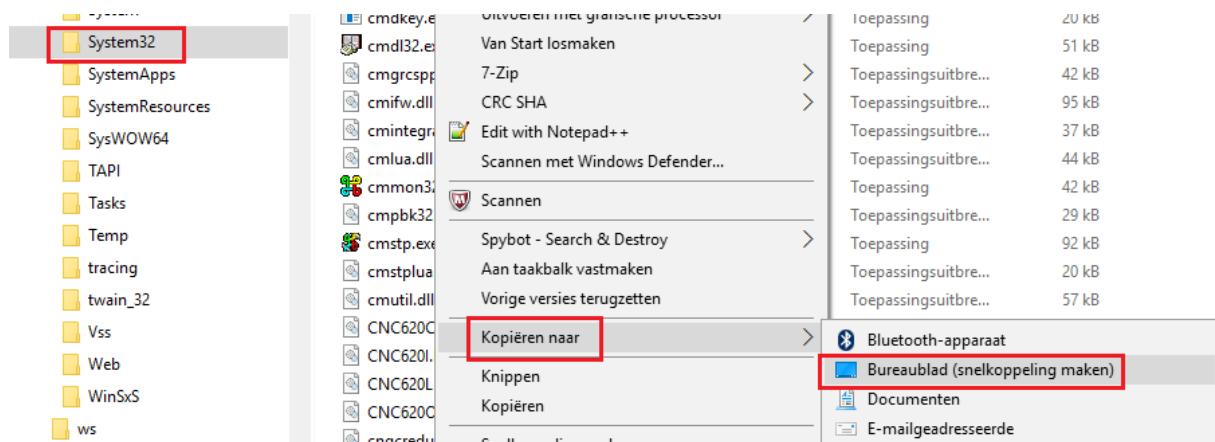
→ Finish



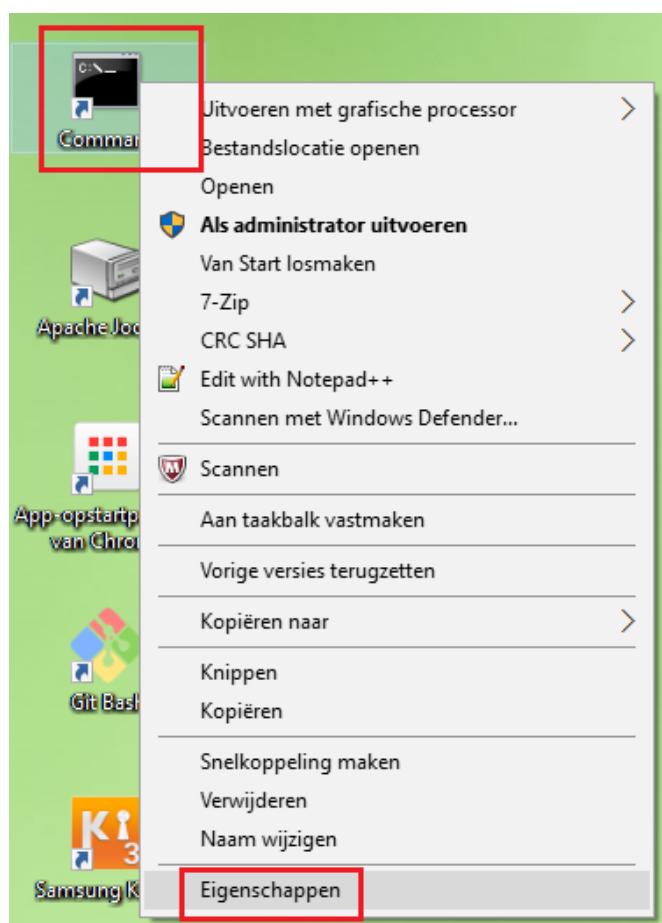
Part 1: The 35 euro IoT project

4.3.3.2 Installeren benodigde package

Ga naar C:\windows\system32 en zoek naar cmd.exe → rechtermuisknop → Kopiëren naar → Bureaublad (snelkoppeling maken)

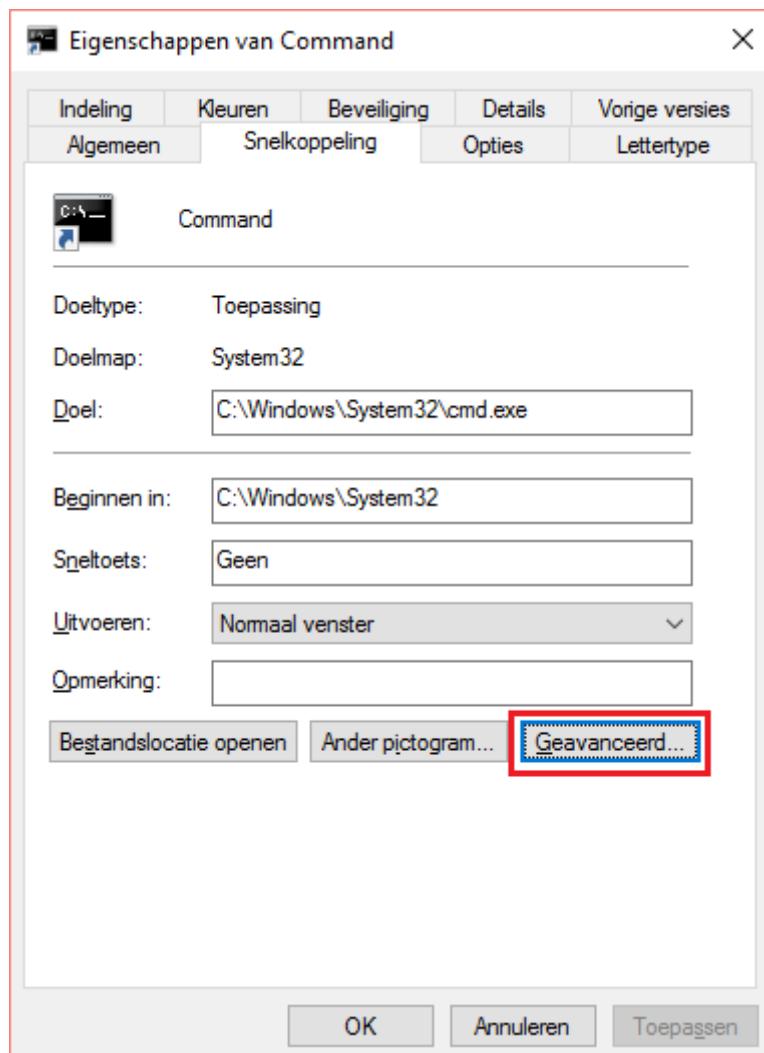


Selecteer snelkoppeling op het bureaublad met de rechtermuisknop → Eigenschappen

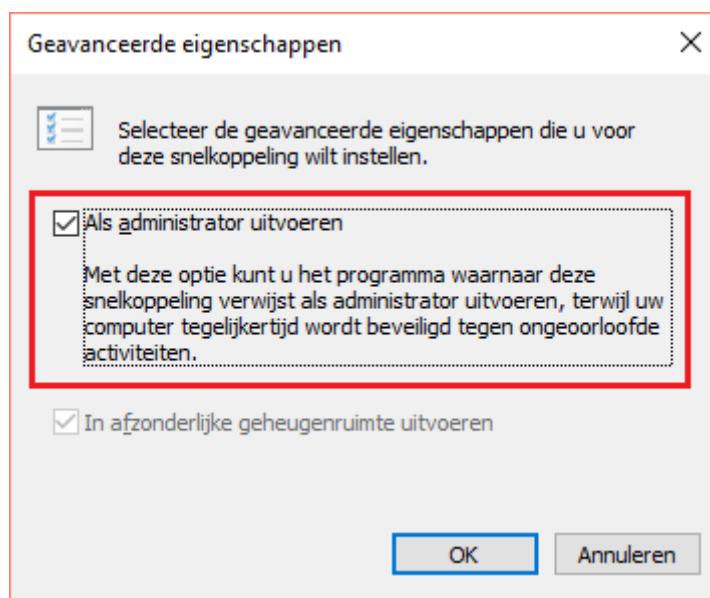


Part 1: The 35 euro IoT project

→ Geavanceerd

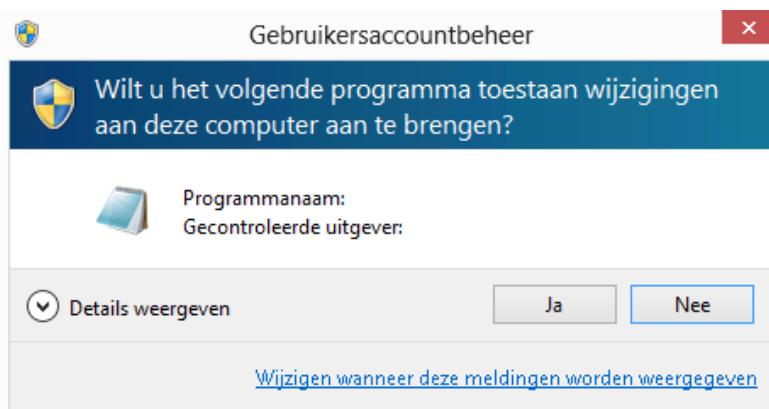


Vink "Als administrator uitvoeren" aan → OK → OK



Part 1: The 35 euro IoT project

Dubbelklik met de muis op de snelkoppeling → Ja



Opdracht: cd\program files\nodejs

```
Administrator: Command
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. Alle rechten voorbehouden.

C:\WINDOWS\system32>cd\program files\nodejs

C:\Program Files\nodejs>
```

Voer de volgende commando's uit:

- npm install express
- npm install serve-static
- npm install body-parser
- npm install mosca --save
- npm install mongodb --save

```
C:\Program Files\nodejs>npm install express
express@4.13.4 node_modules\express
|__ escape-html@1.0.3
|__ utils-merge@1.0.0
|__ methods@1.1.2
|__ vary@1.0.1
|__ fresh@0.3.0
|__ range-parser@1.0.3
|__ content-type@1.0.1
|__ parseurl@1.3.1
|__ cookie-signature@1.0.6
|__ etag@1.7.0
|__ serve-static@1.10.2
|__ content-disposition@0.5.1
|__ array-flatten@1.1.1
|__ cookie@0.1.5
|__ merge-descriptors@1.0.1
|__ path-to-regexp@0.1.7
|__ depd@1.1.0
|__ qs@4.0.0
|__ on-finished@2.3.0 (ee-first@1.1.1)
|__ finalhandler@0.4.1 (unpipe@1.0.0)
|__ debug@2.2.0 (ms@0.7.1)
|__ proxy-addr@1.0.10 (forwarded@0.1.0, ipaddr.js@1.0.5)
|__ accepts@1.2.13 (negotiator@0.5.3, mime-types@2.1.10)
|__ type-is@1.6.12 (media-typer@0.3.0, mime-types@2.1.10)
|__ send@0.13.1 (destroy@1.0.4, statuses@1.2.1, ms@0.7.1, mime@1.3.4, http-errors@1.3.1)
```

Part 1: The 35 euro IoT project

4.3.3.3 Server script aanmaken

In deze paragraaf wordt een voorbeeld NodeJS script aangemaakt. Deze dient als voorbeeld voor het AngularJS voorbeeld dat later in dit boek aan de orde komt.

Maak met een editor (Kladblok, Voetpad++, ..) een bestand mijnserver.js aan op bijv. c:\temp. Het rechtstreek aanmaken van dit bestand op c:\Program Files\nodejs lukt niet onder Windows 10 vanwege de schrijfrechten op deze directory.

Tip: via Notepad++ kan dit wel middels de extensie Notepad++ SaveAsAdmin plug-in

Het aangemaakte script verwijst naar een vaste locatie op de schijf (c:\hallo). Hiervoor wordt serveStatic('.../.../hallo') gebruikt (NodeJS staat op c:\program files\nodejs).

Source: mijnNodeServer.js

```
var
  express = require('express'),
  serveStatic = require('serve-static');
  nodePort = 5000;

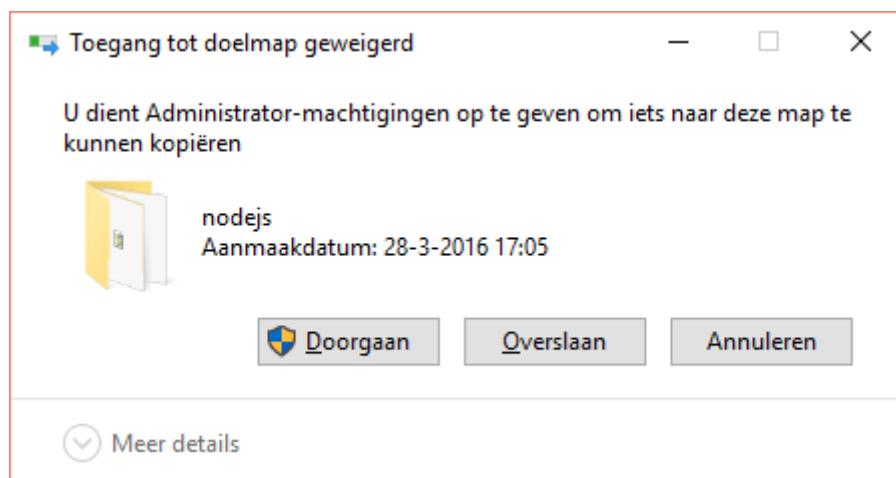
// Start NodeJS
app = express();
app.use( serveStatic('.../.../hallo') );
app.listen(nodePort);

console.log('Node server is up and running')
```

Zie: Sources → NodeJS → mijnNodeServer.js

In dit script worden eerste de benodigde modules geïmporteerd (require opdrachten). Daarna wordt de NodeJS Express server gestart op poort 5000.

Kopieer dit bestand naar: c:\Program Files\nodejs → Doorgaan



Part 1: The 35 euro IoT project

4.3.3.4 Starten server

Start de servers van C:\Program Files\nodejs met de opdracht: node mijnserver.js

```
Administrator: Command - node mijnNodeServer.js
C:\Program Files\nodejs>node mijnNodeServer.js
Node server is up and running
```

4.3.4 AngularJS

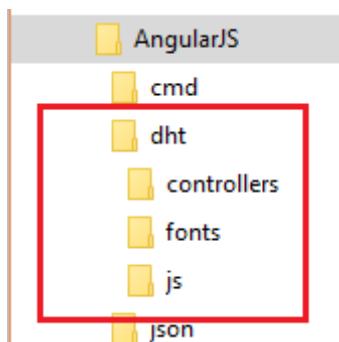
4.3.4.1 AngularJS installatie

Er is geen installatie nodig voor AngularJS. Het bestaat uit een aantal JavaScript libraries die vrij te downloaden zijn. In dit project gebruiken we:

- angular-min.js
- angular-resource.min.js

Voor dit project wordt de volgende project (directory structuur) gebruikt:

```
C: -+
  + AngularJS -+
    + dht --+
      + controllers
      + fonts
      + js
```



Open een browser en ga naar:

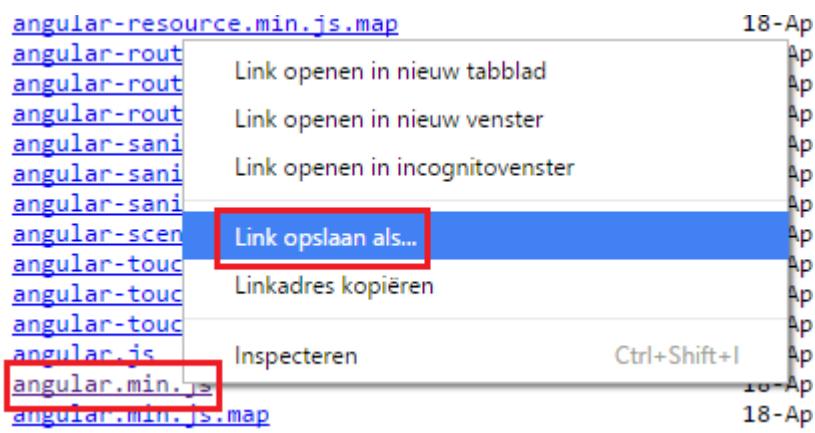
<https://code.angularjs.org/1.5.5/>

Part 1: The 35 euro IoT project

Download:

```
angular-resource.js  
angular-resource.min.js  
angular-resource.min.js.map  
angular-route.js  
angular-route.min.js  
angular-route.min.js.map  
angular-sanitize.js  
angular-sanitize.min.js  
angular-sanitize.min.js.map  
angular-scenario.js  
angular-touch.js  
angular-touch.min.js  
angular-touch.min.js.map  
angular.js  
angular.min.js  
angular.min.js.map
```

Selecteer het te downloaden script met de rechtermuisknop → Link opslaan als...



Sla de scripts op in C:\AngularJS\dht\js

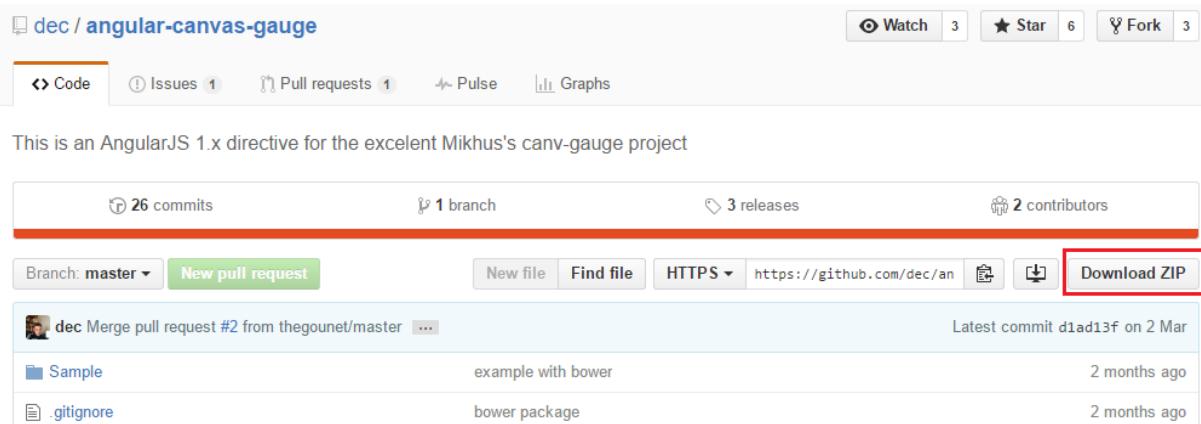
Part 1: The 35 euro IoT project

4.3.4.2 HTML5 Canvas Gauge 1

De implementatie van de temperatuur- en vochtigheidsmeter wordt geïmplementeerd in twee zogenaamde canvas gauge scripts.

Open een browser en ga naar <https://github.com/dec/angular-canvas-gauge>

Download het zip bestand.



This is an AngularJS 1.x directive for the excellent Mikhus's canv-gauge project

Branch: master ▾ New pull request

26 commits 1 branch 3 releases 2 contributors

Download ZIP

| | | |
|------------|--------------------|--------------|
| Sample | example with bower | 2 months ago |
| .gitignore | bower package | 2 months ago |

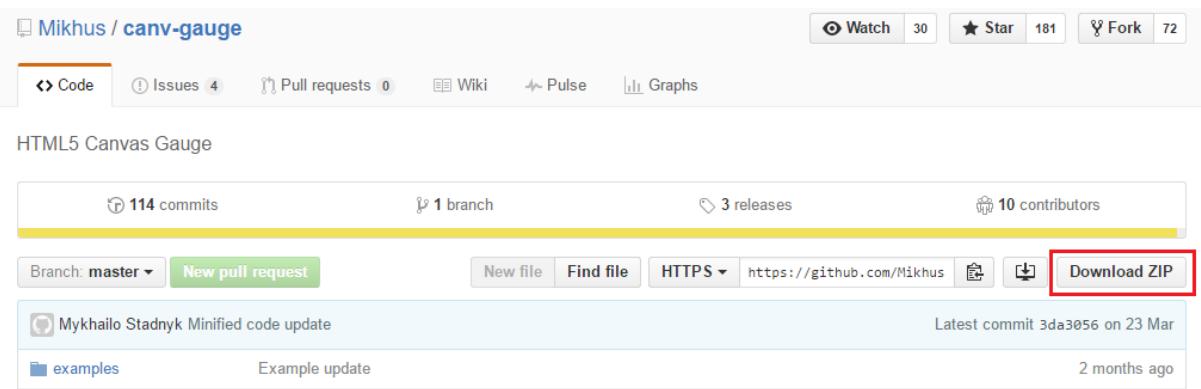
Pak het ZIP-bestand uit en kopieer angular-canvas-gauge.js naar C:\AngularJS\dht\js.
Kopieer de map fonts naar C:\AngularJS\dht.

4.3.4.3 HTML5 Canvas Gauge 2

Dit is een gauge implementatie met dank aan Met dank aan Mykhailo Stadnyk.

Open een browser en ga naar: <https://github.com/Mikhus/canv-gauge>

Download de zip.



HTML5 Canvas Gauge

114 commits 1 branch 3 releases 10 contributors

Download ZIP

| | | |
|---------------------------------------|---------------------------------|--------------|
| Mykhailo Stadnyk Minified code update | Latest commit 3da3056 on 23 Mar | |
| examples | Example update | 2 months ago |

Pak het ZIP-bestand en sla het bestand gauge.js in c:\AngularJS\dht22\js.

Part 1: The 35 euro IoT project

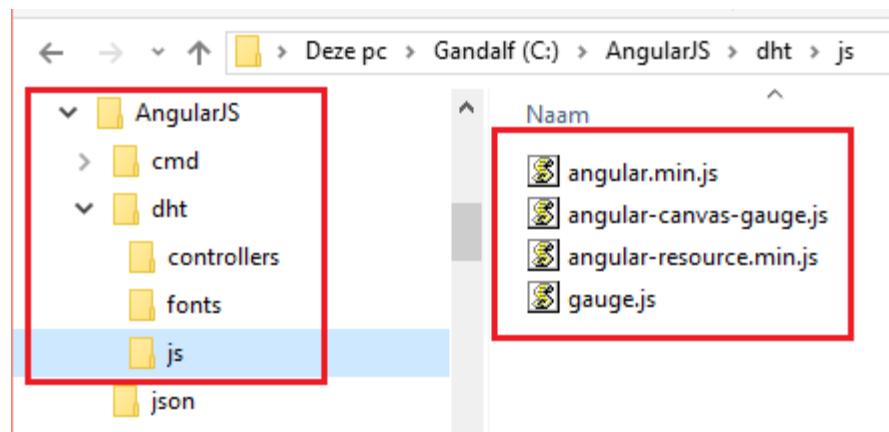
Breng de volgende fix aan (vanaf regel 160):

```
/**  
 * Sets a new value to gauge and updates the gauge view  
 *  
 * @param {number} val - the new value to set to the gauge  
 * @return {Gauge} this - returns self  
 */  
this.setValue = function (val) {  
  
    fromValue = config.animation ? value : val;  
    var dv = (config.maxValue - config.minValue) / 100;  
  
    /* bug fix: values < 0.00 and > -2.00 should be corrected! */  
    if( val < 0.00 && val > -2.00 ) {  
        toValue = val;  
    } else {  
        toValue = val > config.maxValue ?  
            config.maxValue + dv :  
            val < config.minValue ?  
                config.minValue - dv :  
                val;  
    }  
    value = val;  
  
    config.animation ? animate() : this.draw();  
    return this;  
};
```

Zie: Sources → AngularJS → dht → js → gauge.js

Deze fix is een correctie op de fout als de waarde tussen 0 en -2 valt. Er wordt dan -10 als uitkomst getoond.

De C:\AngularJS\dht\js bevat nu de volgende vier bestanden:

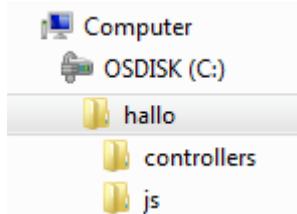


Part 1: The 35 euro IoT project

4.3.4.4 Een simpel voorbeeld.

De werking van AngularJS kan het beste worden getoond met een simpel voorbeeld.

Hiervoor wordt de volgende directory structuur gebruikt:



In de map hallo wordt het bestand index.html aangemaakt met de volgende inhoud:

```
<html ng-app="helloApp" >
  <head>
    <script type="text/javascript" src="js/angular.min.js"></script>
    <script type="text/javascript" src="js/app.js"></script>
    <script type="text/javascript" src="controllers/controller.js"></script>
  </head>

  <body>
    <div ng-controller="helloController">
      <p> {{hello}} </p>
    </div>
  </body>
</html>
```

Zie: Sources → AngularJS → hallo → index.html

In index.html zitten 2 directives:

- ng-app verwijzing naar de te gebruiken applicatie
- ng-controller verwijzing naar de te gebruiken controller

Daarnaast worden er een 3-tal Java-scripts geladen:

- angular.min.js de AngularJS library
- app.js de applicatie
- controller.js de controller

En tenslotte is er één datakoppeling aanwezig: {{hello}}.

De applicatie wordt verder uitgewerkt in app.js en controller.js.

Source app.js:

```
var helloApp = angular.module('helloApp', []);
```

Zie: Sources → AngularJS → hallo → js → app.js

Deze ene regel verwijst naar de ng-app directive in index.html.

Part 1: The 35 euro IoT project

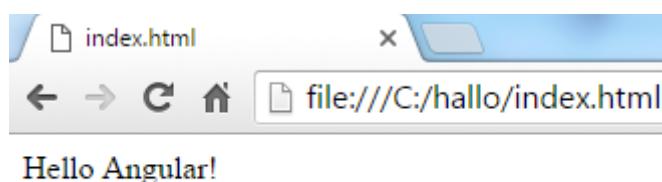
Source controller.js:

```
helloApp.controller('helloController', function($scope) {  
    $scope.hello = 'Hello Angular!';  
});
```

Zie: Sources → AngularJS → hallo → controllers → controllers.js

Dit is de feitelijke uitwerking van de app. In de controller wordt de datakoppeling {{hello}} in index.html van een waarde voorzien middels \$scope.hello.

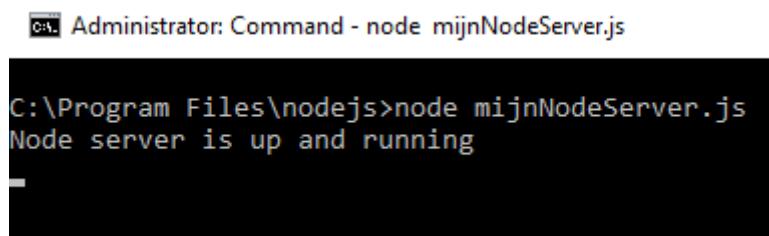
AngularJS is in principe niet afhankelijk van een server en kan derhalve via het bestand index.html gestart worden. Resultaat:



In dit voorbeeld is goed te zien hoe een deel van de variabele inhoud van het uiteindelijke getoonde scherm wordt opgebouwd via de app en de controller.

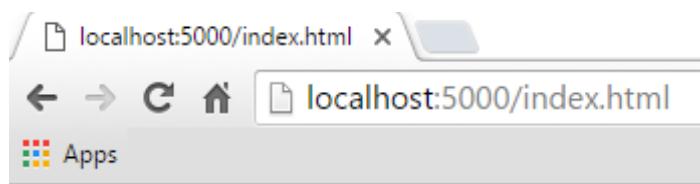
Het voorbeeld werkt ook via het eerder aangemaakte mijnNodeServer.js (zie NodeJS installatie).

Start de servers van C:\Program Files\nodejs met de opdracht: node mijnNodeServer.js



Open een browser en open de link:

<http://localhost:5000/index.html>



Part 1: The 35 euro IoT project

4.3.5 Chart.js

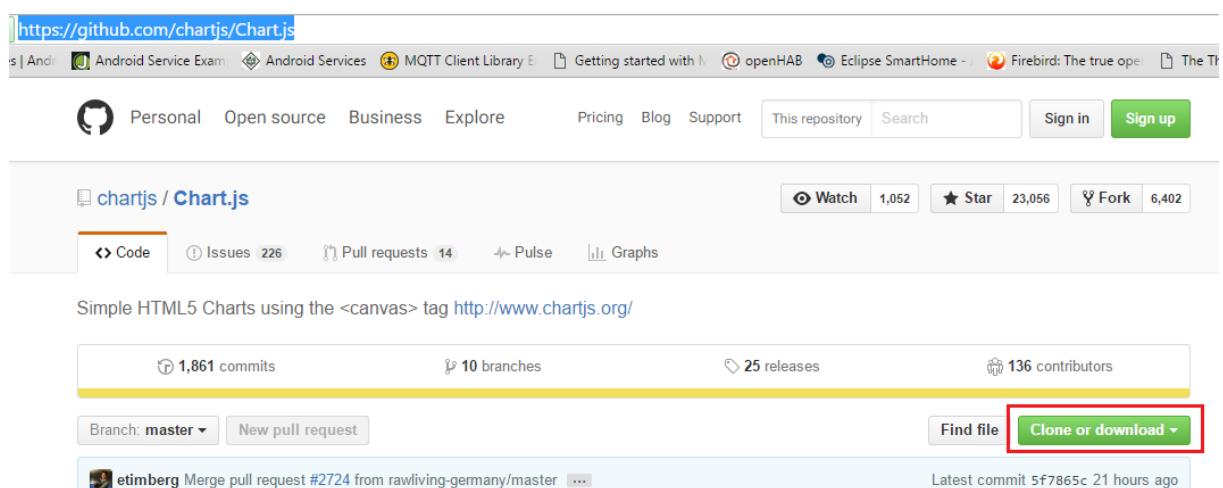
4.3.5.1 Download

Voor het tonen van grafieken wordt Chart.js gebruikt (<http://www.chartjs.org/>).

Open een browser en ga naar:

<https://github.com/chartjs/Chart.js>

→ Clone or download



The screenshot shows the GitHub repository page for 'chartjs / Chart.js'. At the top, there are navigation links for Personal, Open source, Business, Explore, Pricing, Blog, Support, and a search bar. Below the header, there's a summary section with metrics: 1,861 commits, 10 branches, 25 releases, and 136 contributors. A red box highlights the 'Clone or download' button. The main content area displays a commit history with a recent merge pull request from 'etimberg' and a timestamp for the latest commit.

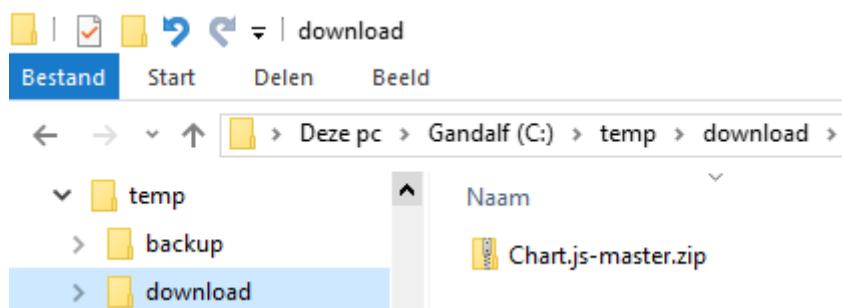
→ Download ZIP



The screenshot shows the GitHub repository page for 'chartjs / Chart.js' with a focus on the cloning options. It displays two methods: 'Clone with HTTPS' and 'Use SSH'. Below these, there's a link to 'Open in Desktop' and a large blue 'Download ZIP' button, which is highlighted with a red box. A timestamp indicates the file was updated 5 days ago.

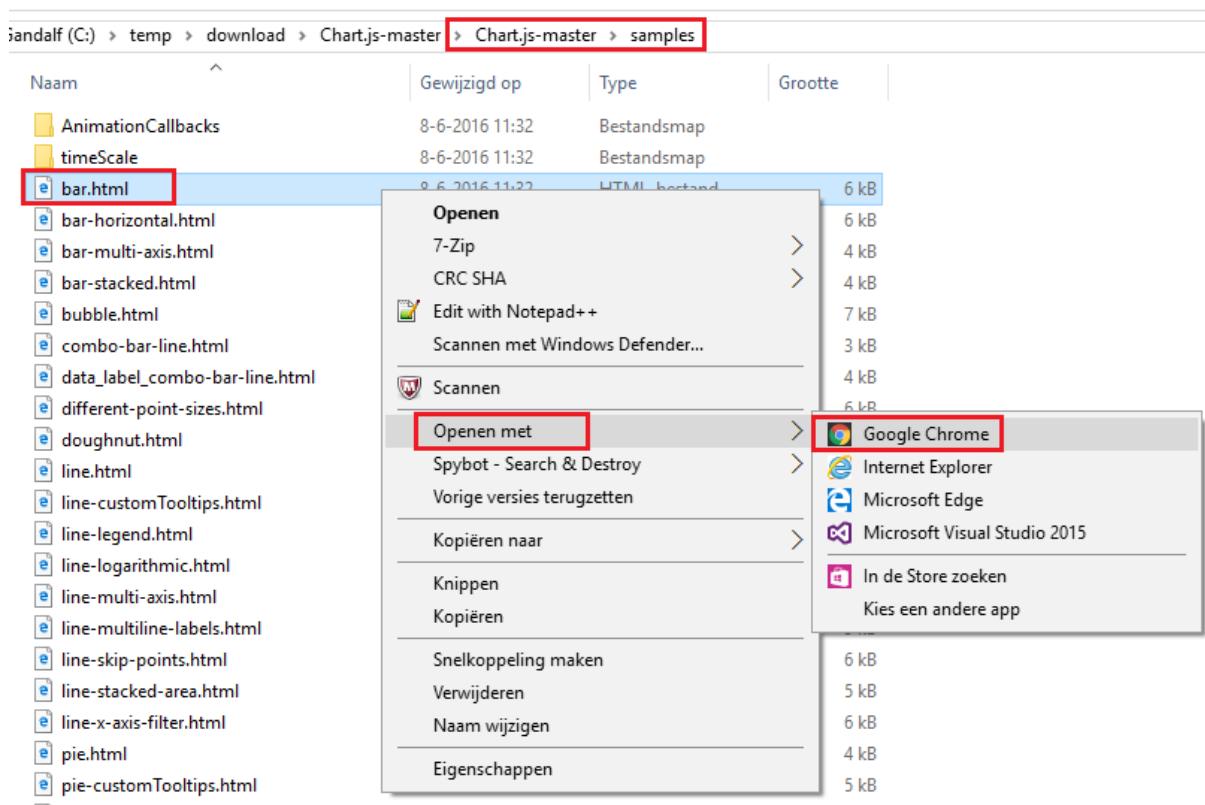
Part 1: The 35 euro IoT project

Pak het ZIP-bestand uit:



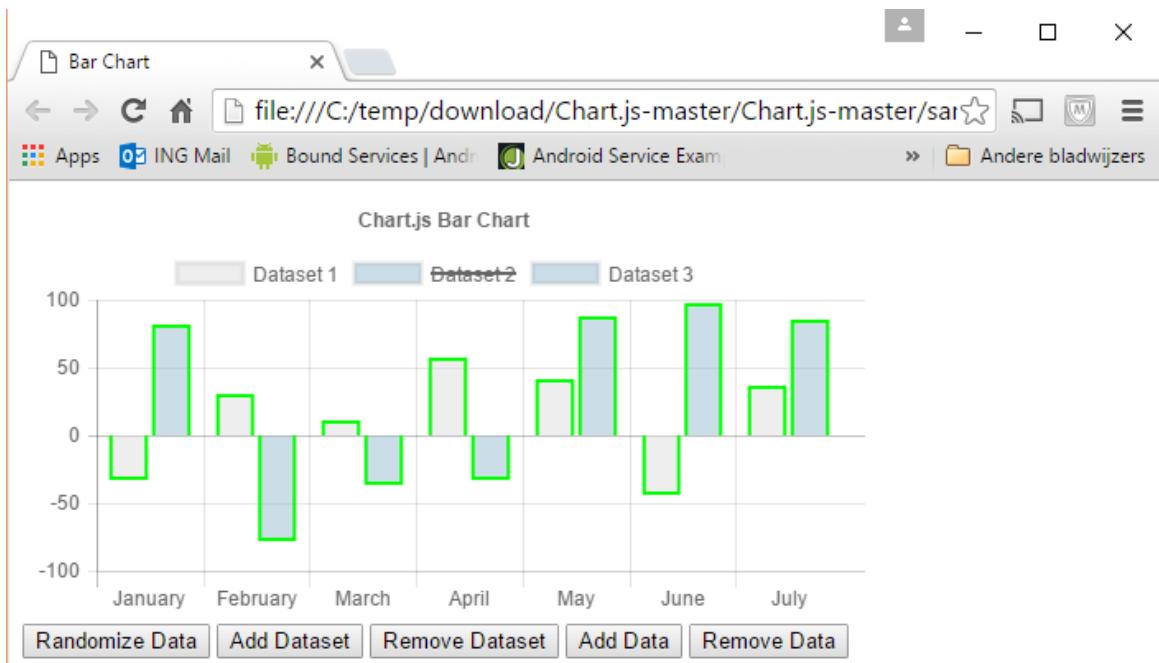
4.3.5.2 Test

Open een verkenner en ga naar het uitgepakte ZIP-bestand en zoek de map samples op → selecteer bar.html met de rechtermuisknop → Openen met → Google Chrome



Part 1: The 35 euro IoT project

Resultaat:



Part 1: The 35 euro IoT project

4.3.6 Mosquitto

4.3.6.1 Inleiding



Mosquitto

An Open Source MQTT v3.1/v3.1.1 Broker

Voor het gebruik van MQTT is een zogenaamde broker nodig. In dit boek wordt voor de broker Mosca in combinatie met NodeJS en AngularJS gebruikt.

Om de werking van MQTT te demonstreren wordt Mosquitto geïnstalleerd. Mosquitto is een open source MQTT-initiatief. De cliënt om berichten te publiceren wordt verder ook gebruikt om de werking van de AngularJS oplossing te testen (los van de ESP8266 hardware opstelling).

4.3.6.2 Installatie

Voor de werking van Mosquitto zijn een drietal DLL's nodig die apart moeten worden opgehaald.

4.3.6.2.1 Stap 1: SSL-dll's.

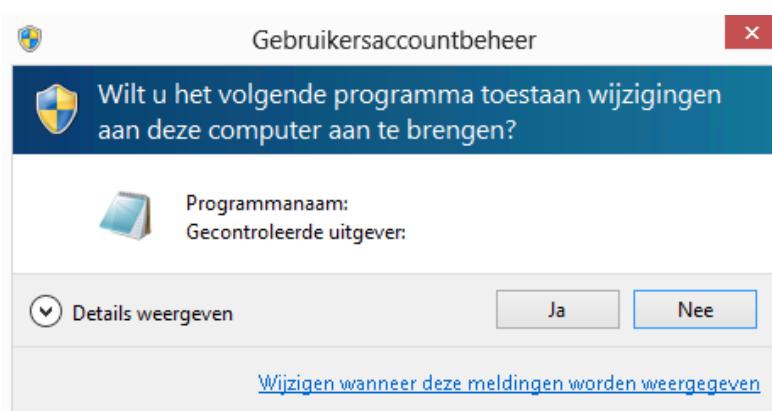
Open een browser en ga naar:

<http://slproweb.com/products/Win32OpenSSL.html>

| Download Win32 OpenSSL today using the links below! | | | |
|---|----------------|--------------|----------|
| File | Type | Description | Download |
| Win32 OpenSSL v1.0.2h Light | 2MB Installer | Inst. Op. | |
| Win32 OpenSSL v1.0.2h | 16MB Installer | Inst. and... | |

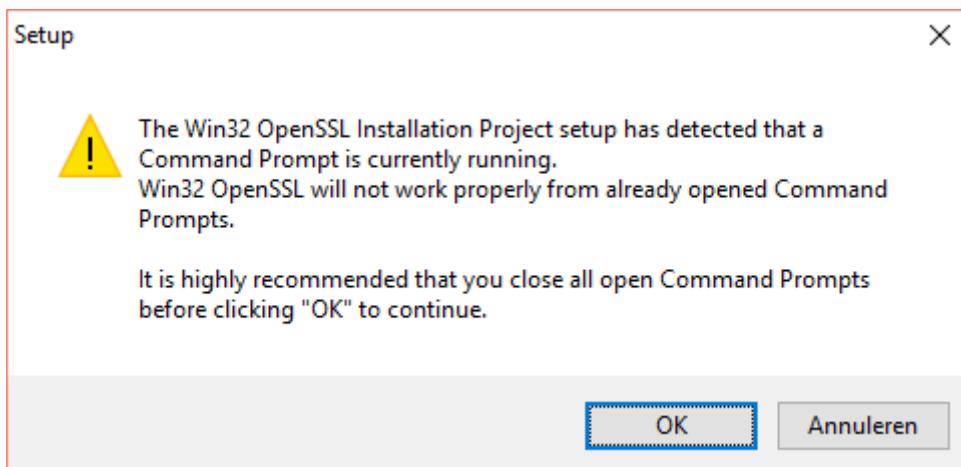
Download en installeer Win32OpenSSL_Light-1_0_2g.exe

→ Ja

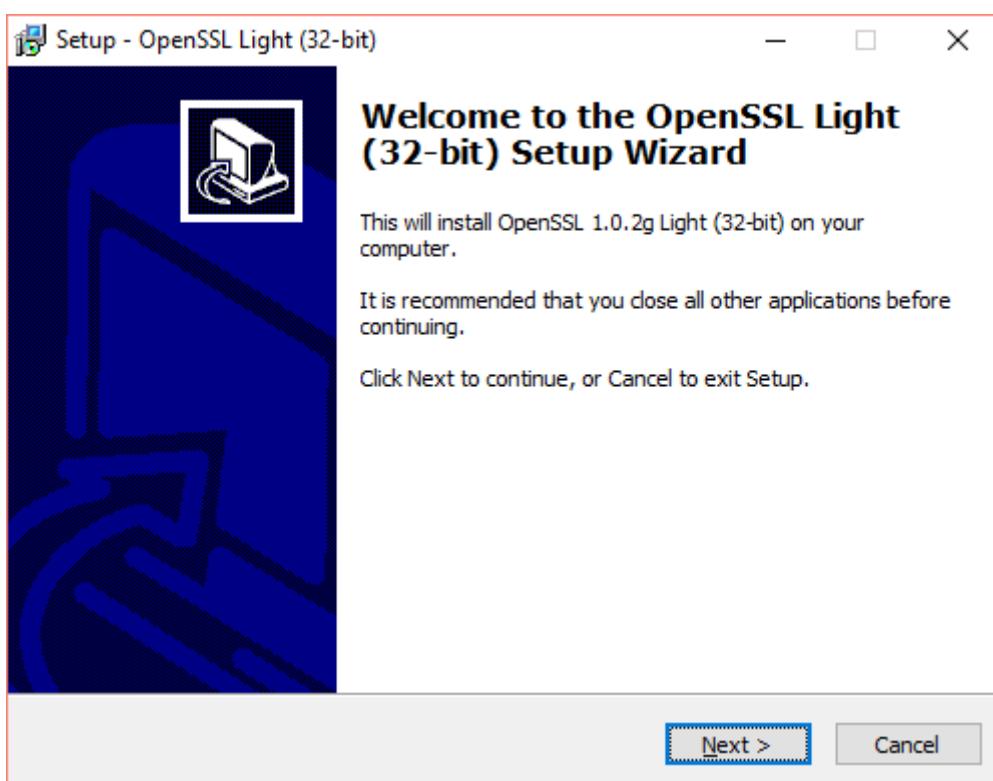


Part 1: The 35 euro IoT project

→ Negeer de waarschuwing → OK

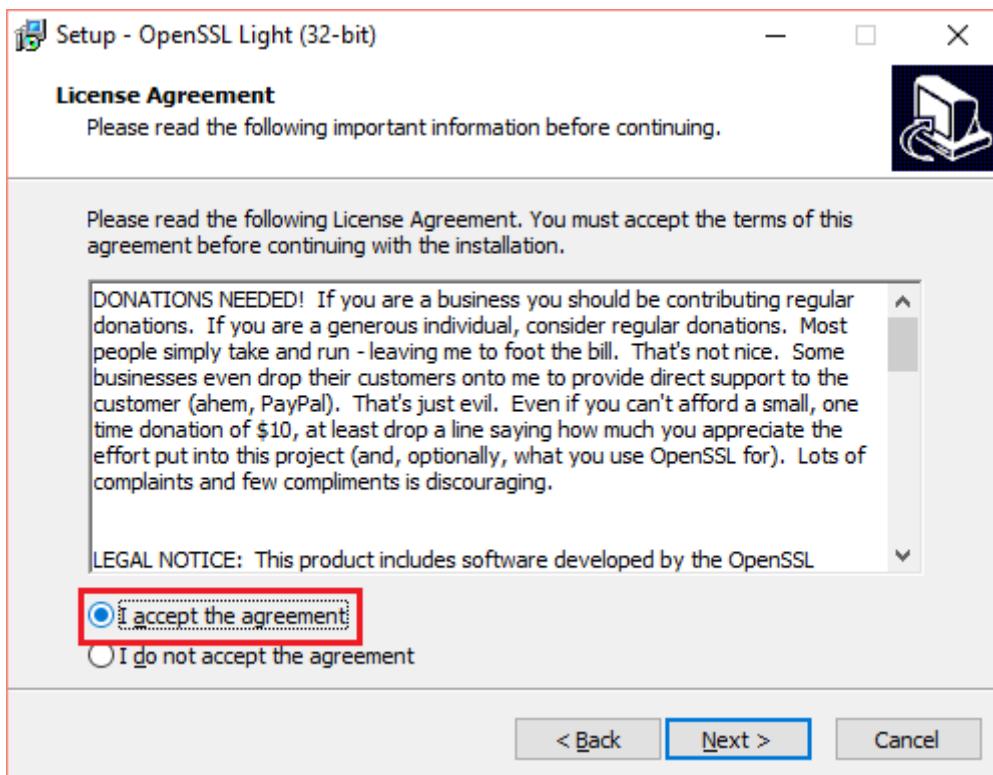


→ Next

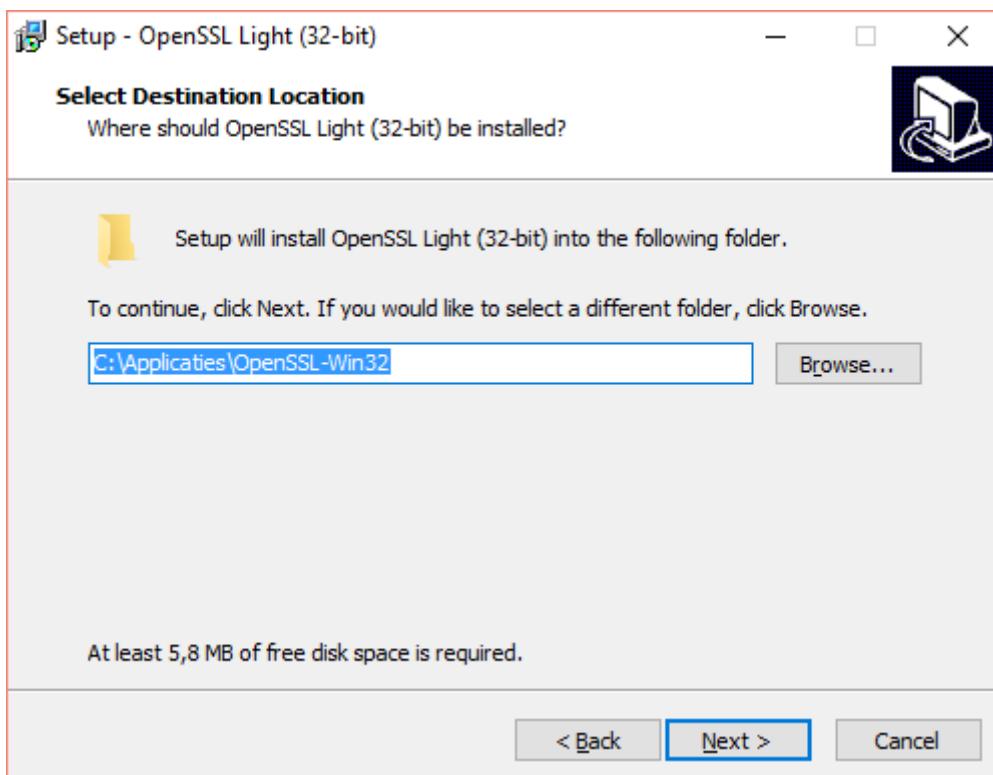


Part 1: The 35 euro IoT project

Vink "I accept..." aan → Next

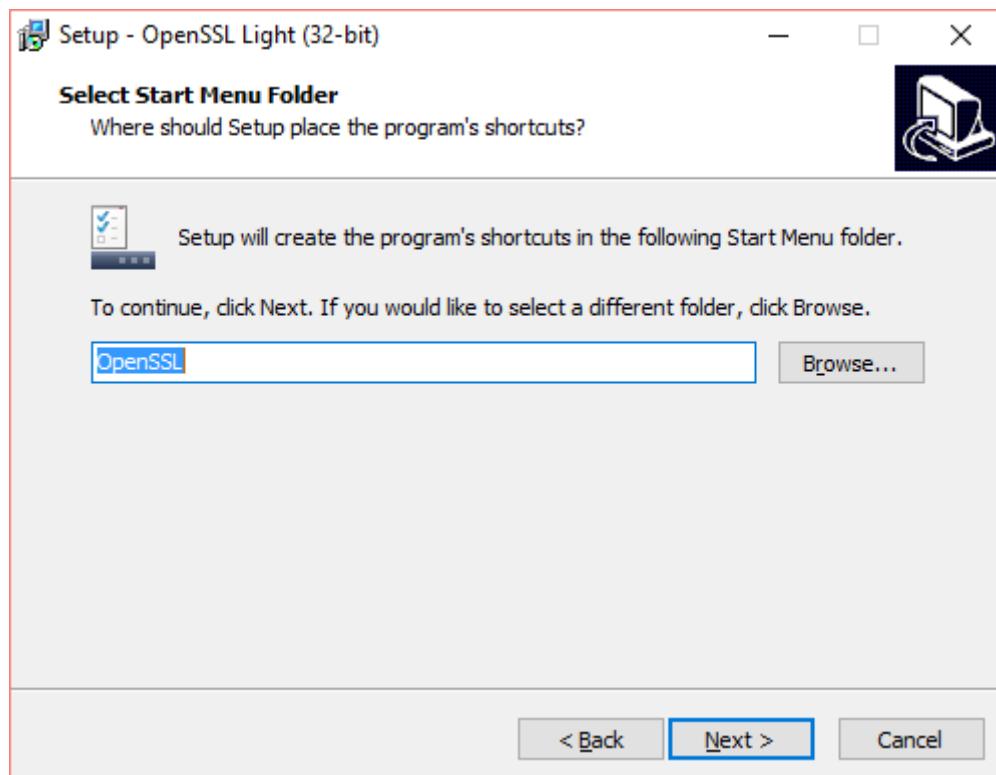


→ Next (kies eventueel een andere folder met Browse..., zoals in het voorbeeld)

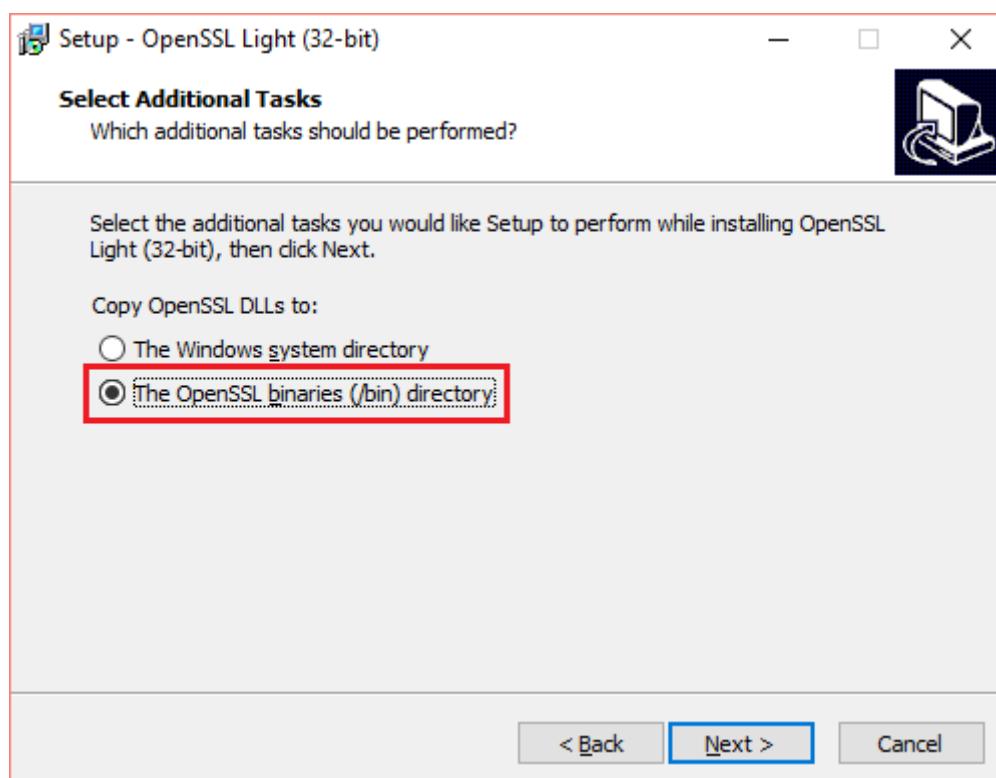


Part 1: The 35 euro IoT project

→ Next

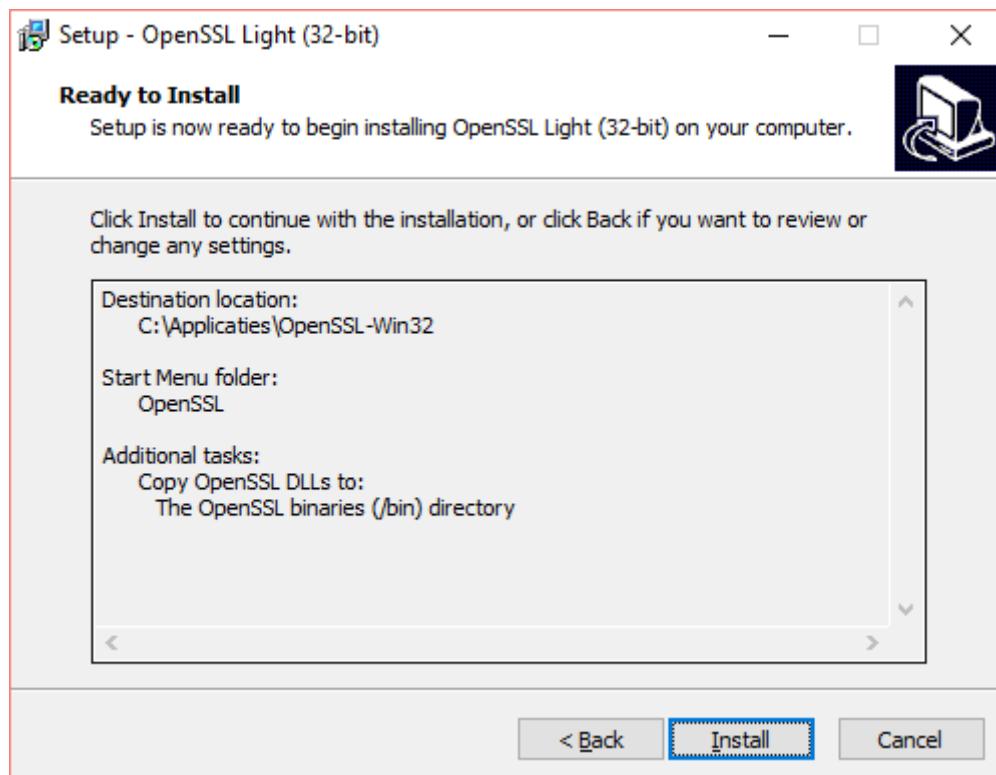


→ Kies een locatie voor de DLL's (handig om te weten omdat een tweetal DLL's straks nodig zijn voor de Mosquitto installatie) → Next

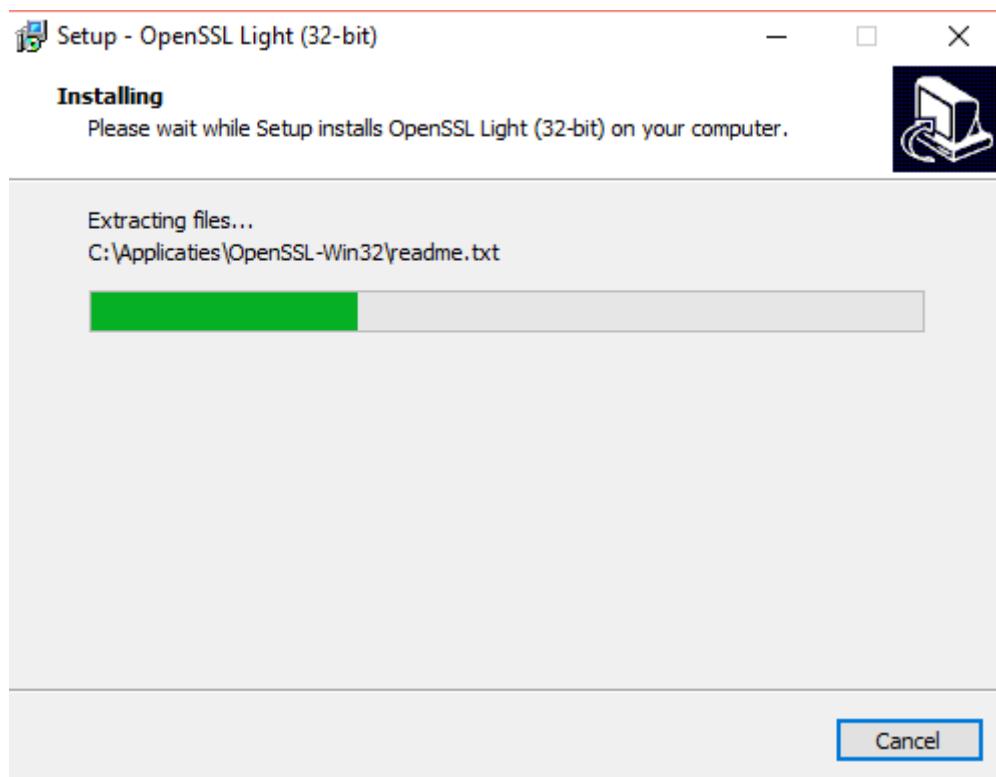


Part 1: The 35 euro IoT project

→ Install

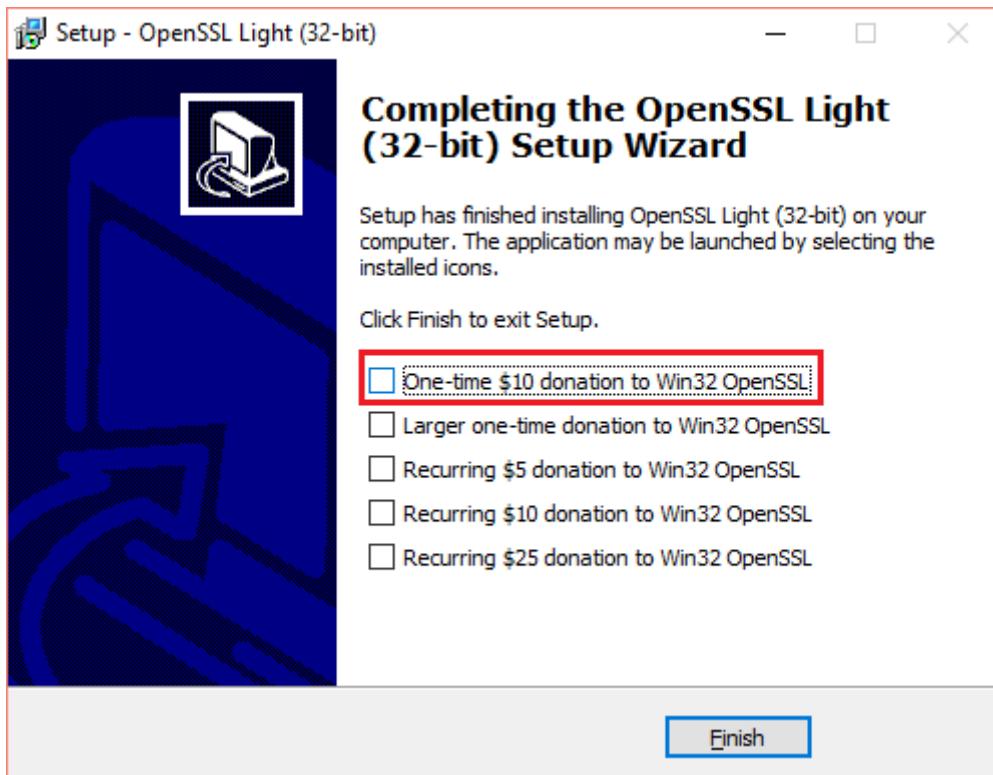


Even geduld...



Part 1: The 35 euro IoT project

Vink donatie uit (doneren mag natuurlijk altijd!) → Finish



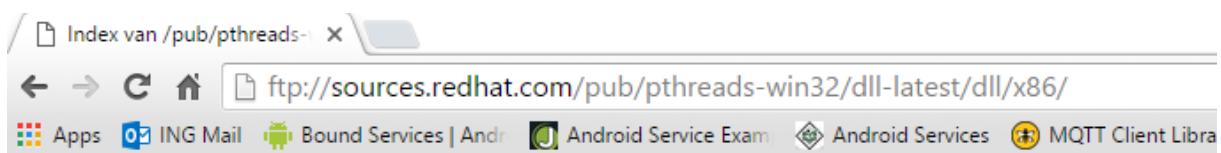
Part 1: The 35 euro IoT project

4.3.6.2.2 Stap 2: Thread-dll.

Open een browser en ga naar:

<ftp://sources.redhat.com/pub/pthreads-win32/dll-latest/dll/x86/>

Klik op pthreadVC2.dll om deze te downloaden.



Index van /pub/pthreads-win32/dll-latest/dll/x86/

| Naam | Grootte | Bijgewerkt op |
|-----------------------|---------|-------------------|
| [hoofddirectory] | | |
| md5.sum | 293 B | 05-02-15 00:00:00 |
| pthreadGC2.dll | 117 kB | 27-05-12 00:00:00 |
| pthreadGCE2.dll | 119 kB | 27-05-12 00:00:00 |
| pthreadVC2.dll | 54.5 kB | 27-05-12 00:00:00 |
| pthreadVCE2.dll | 60.5 kB | 27-05-12 00:00:00 |
| pthreadVSE2.dll | 56.0 kB | 27-05-12 00:00:00 |
| sha512.sum | 866 B | 05-02-15 00:00:00 |

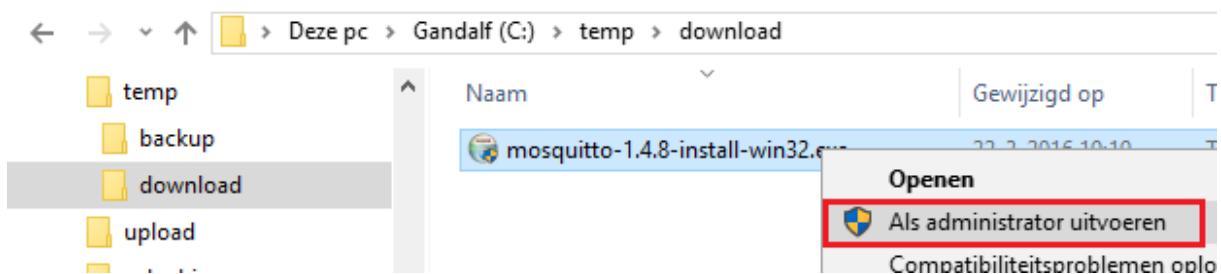
4.3.6.2.3 Stap 3: Mosquitto installatie.

Open een browser en ga naar:

<http://mosquitto.org/download>

The screenshot shows the 'Binary Installation' section for Windows. It lists two executables: 'mosquito-1.4.8-install-win32.exe' (~200 kB) and 'mosquito-1.4.8-install-cygwin.exe' (~200 kB). The 'mosquito-1.4.8-install-win32.exe' link is highlighted with a red box.

Klik op het gedownload bestand met de rechtermuisknop → Als administrator uitvoeren

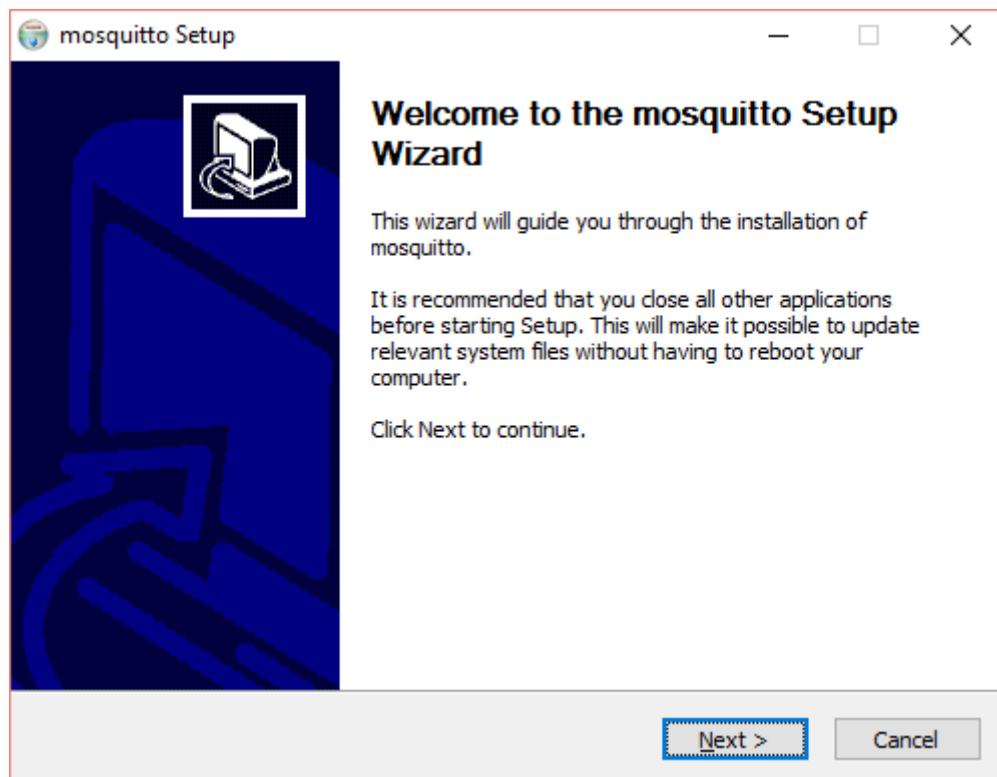


Part 1: The 35 euro IoT project

→ Ja

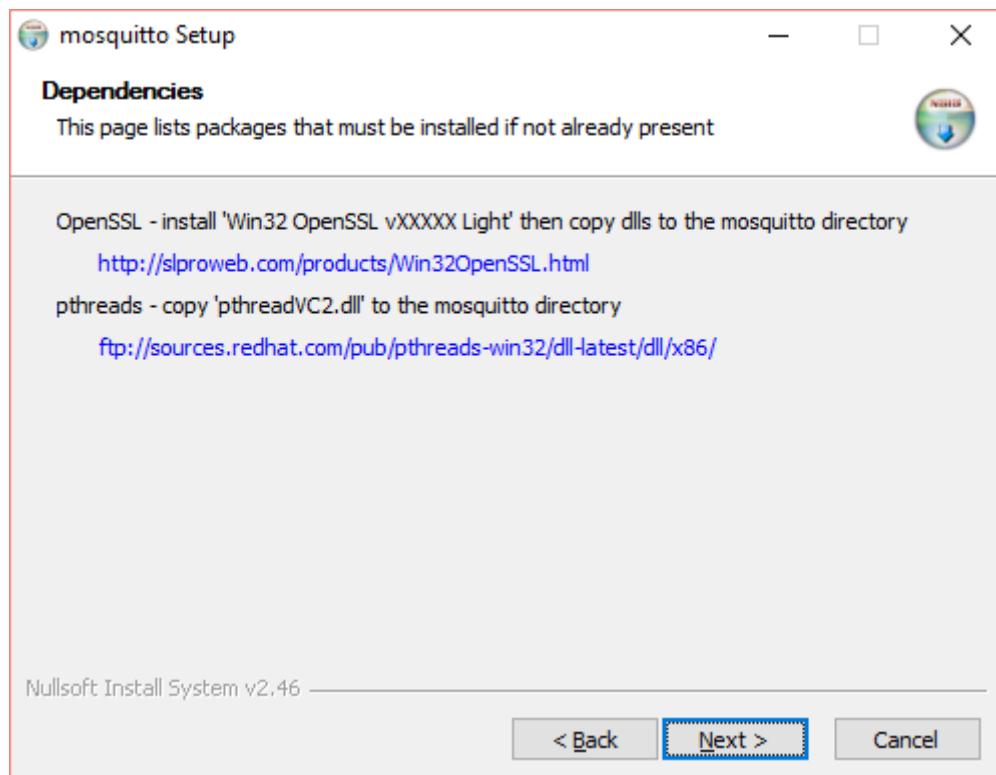


→ Next

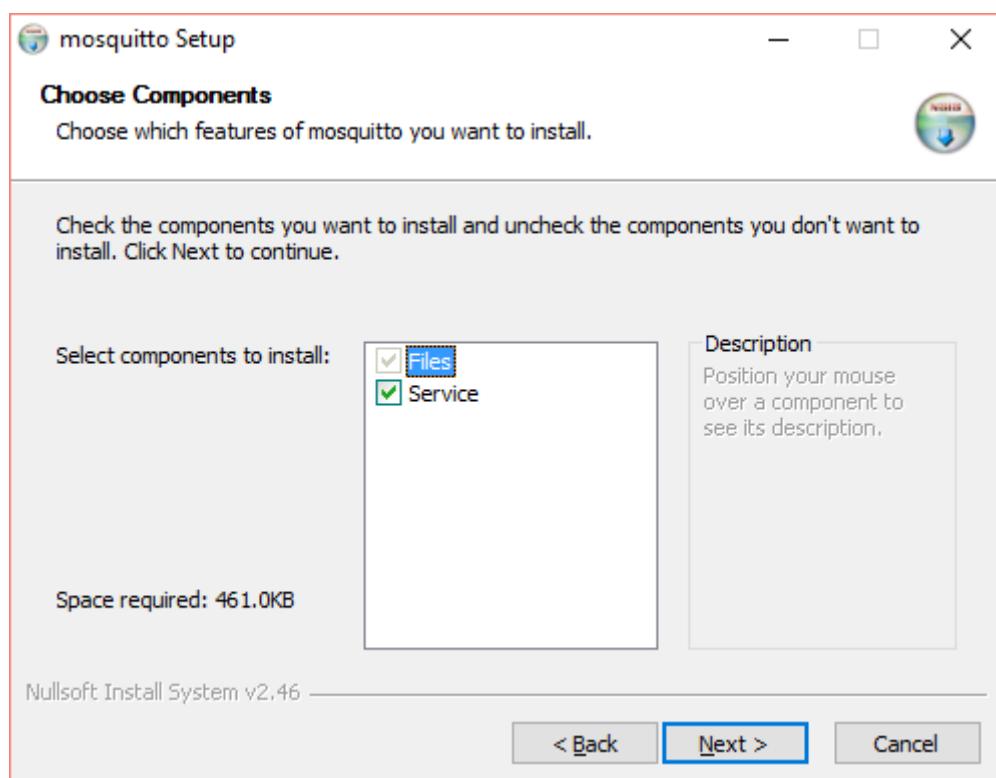


Part 1: The 35 euro IoT project

→ Next

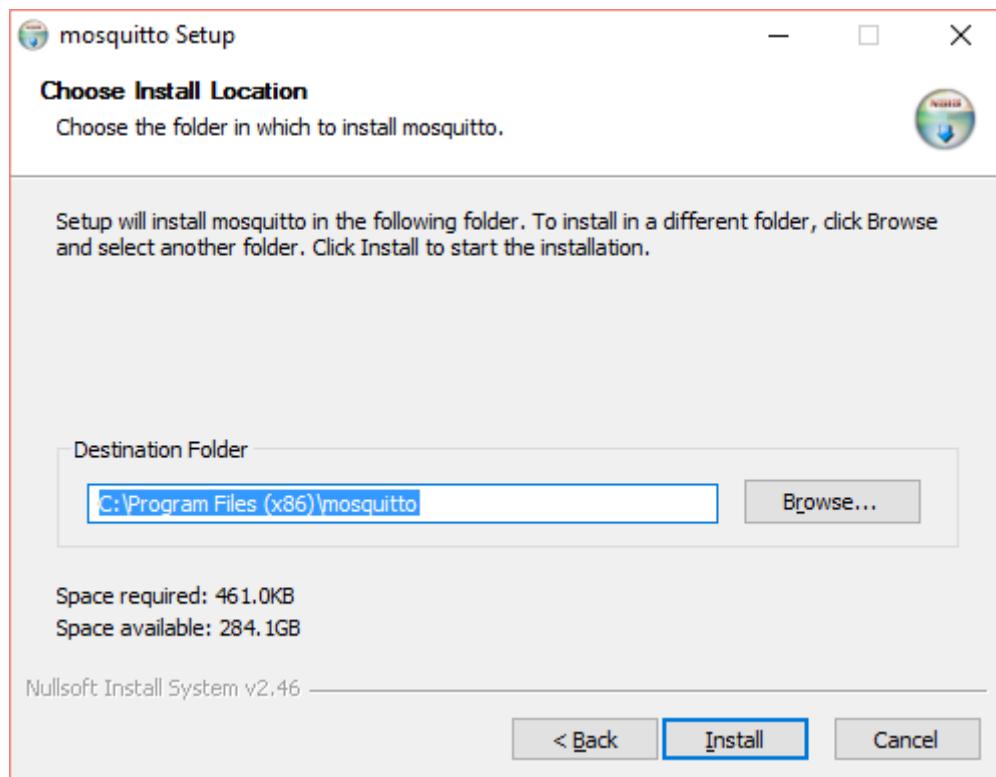


→ Next

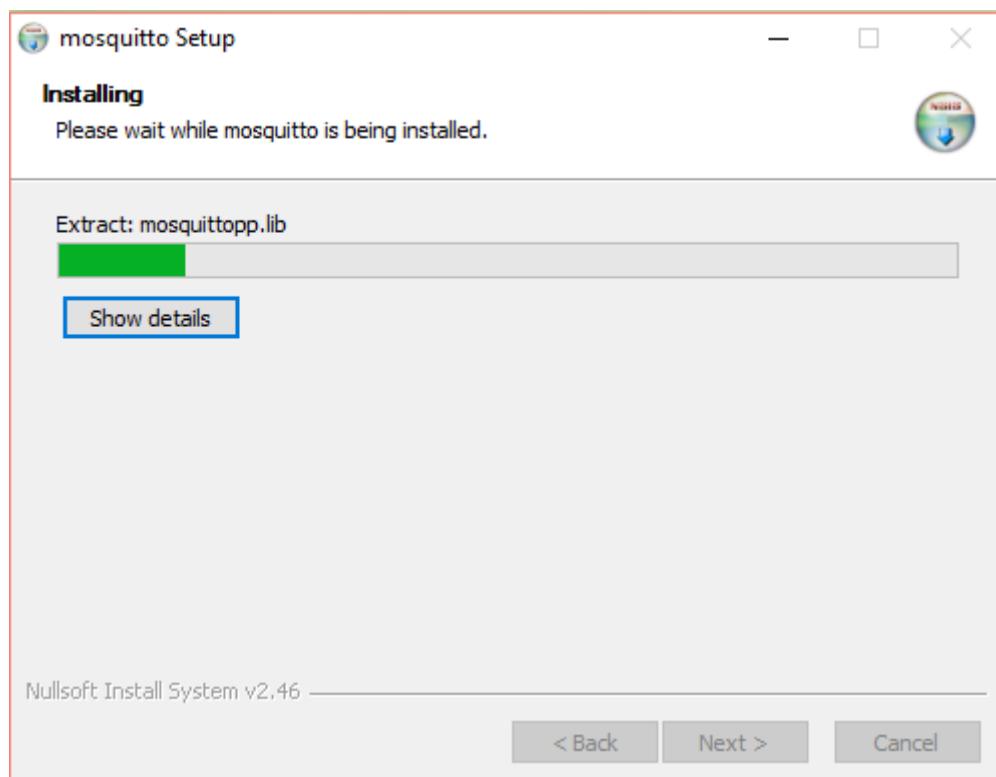


Part 1: The 35 euro IoT project

→ Install

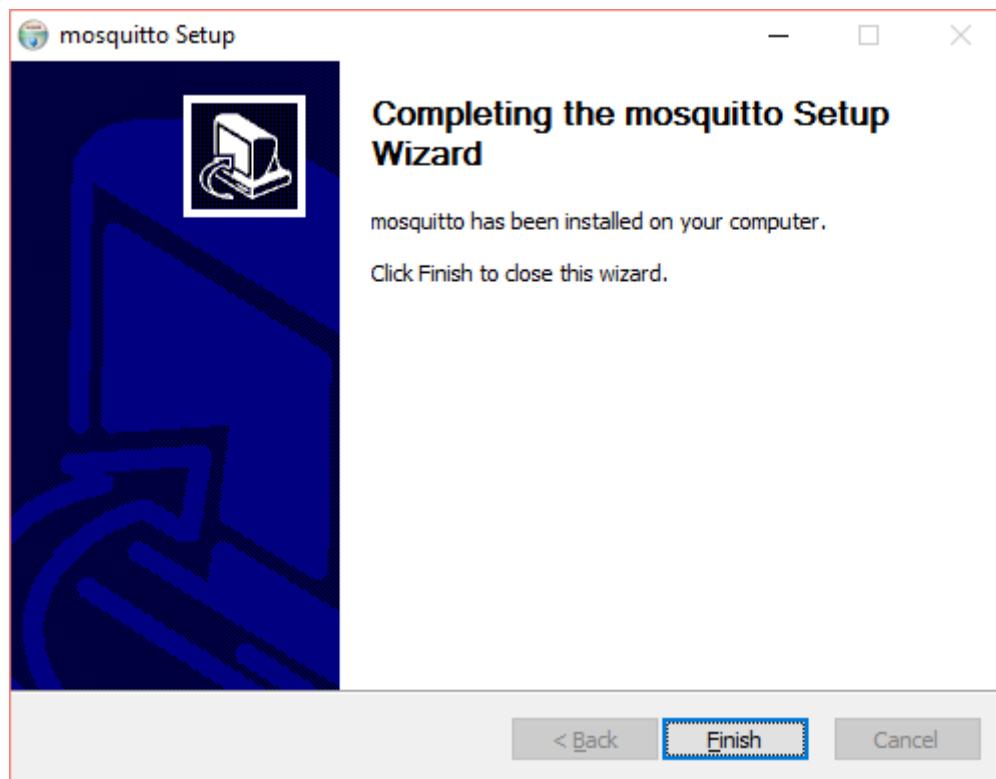


Even geduld...



Part 1: The 35 euro IoT project

→ Finish



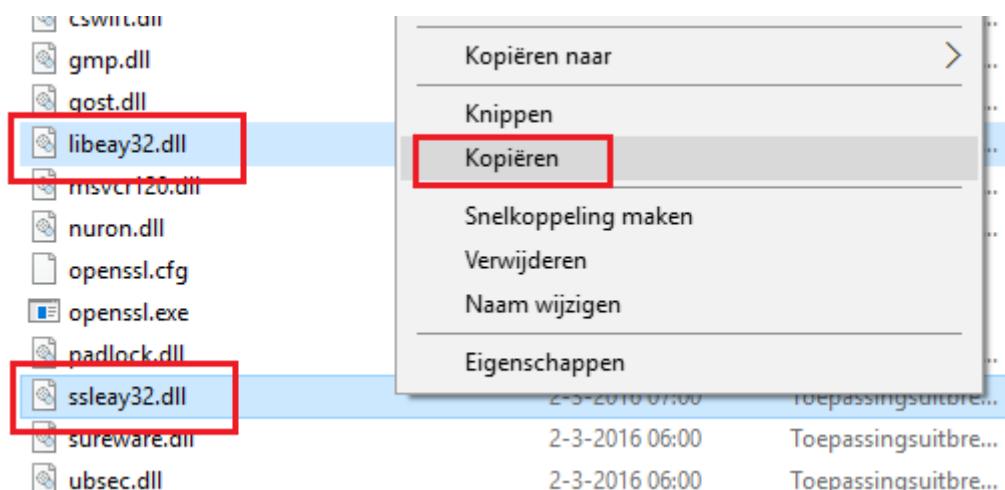
4.3.6.2.4 Stap 4: Installatie afronden

Kopieer de DLL's uit de SSL bin directory (zie stap 1) naar de Mosquitto directory.

Het gaat om de volgende twee DLL's:

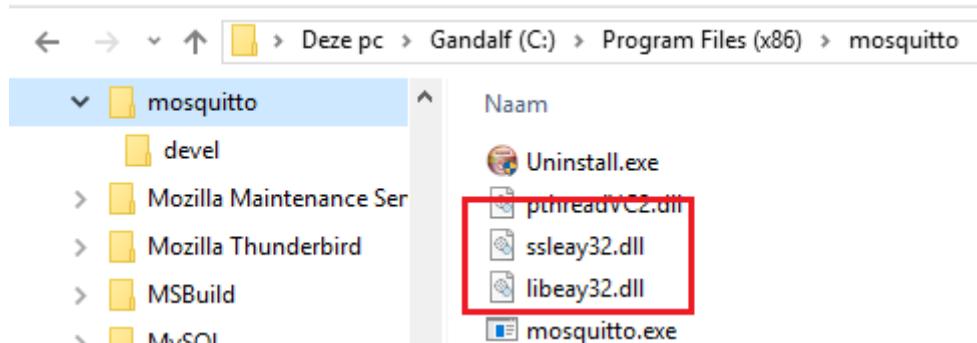
- libeay32.dll
- ssleay32.dll

Selecteer deze met de rechtermuisknop → Kopiëren

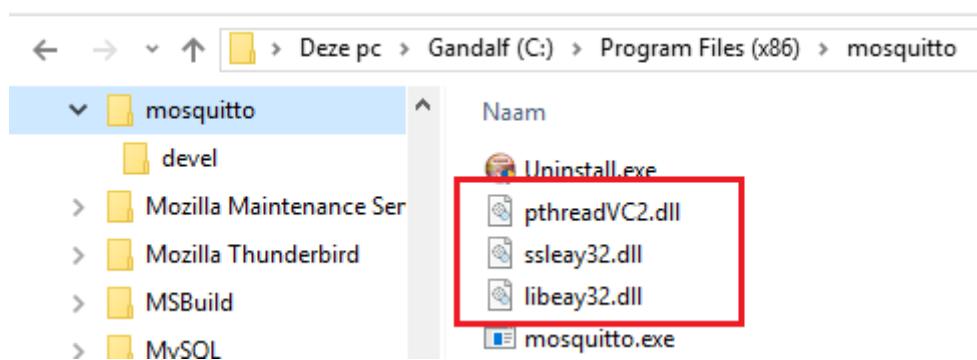


Part 1: The 35 euro IoT project

Plak deze in de C:\Program Files (x86)\mosquitto directory:



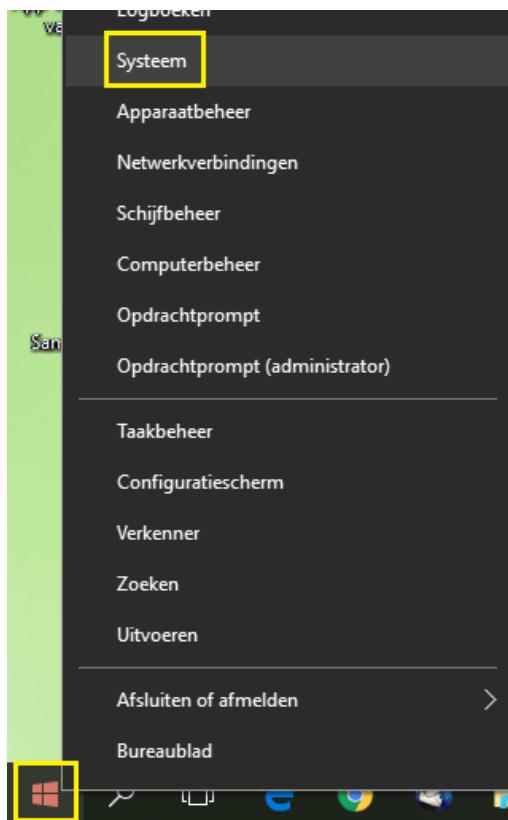
Herhaal dit voor de gedownloade pthreadVC2.dll (uit stap 2).



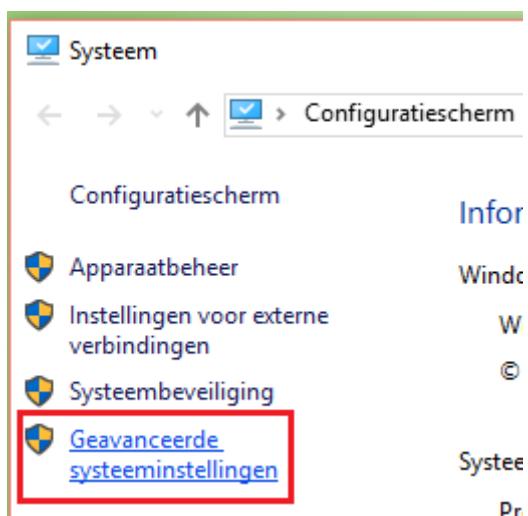
Part 1: The 35 euro IoT project

4.3.6.2.5 Stap 5: PATH-variabele aanpassen

Start → rechtermuisknop → Systeem

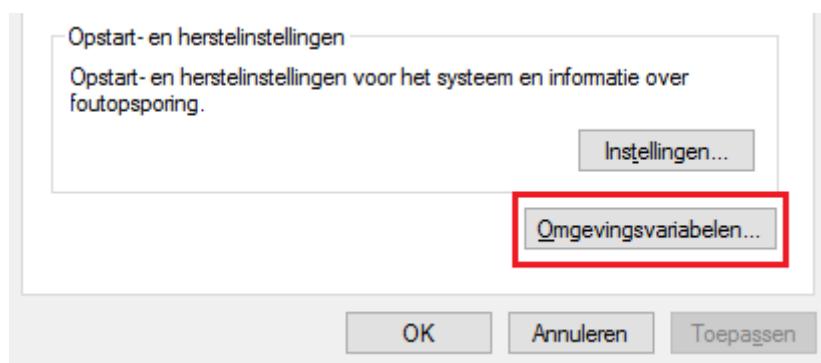


Geavanceerde systeeminstellingen

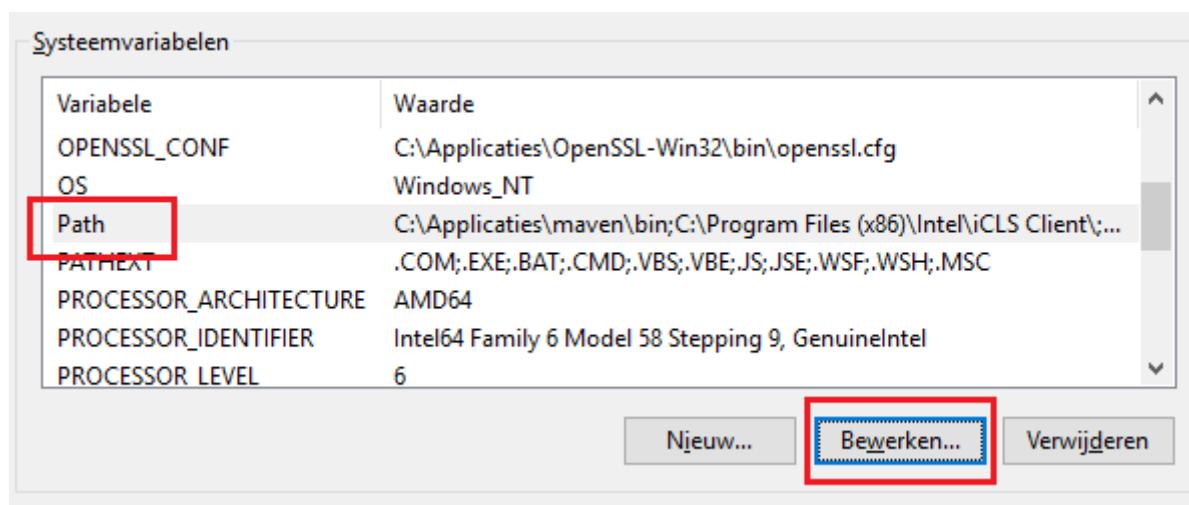


Part 1: The 35 euro IoT project

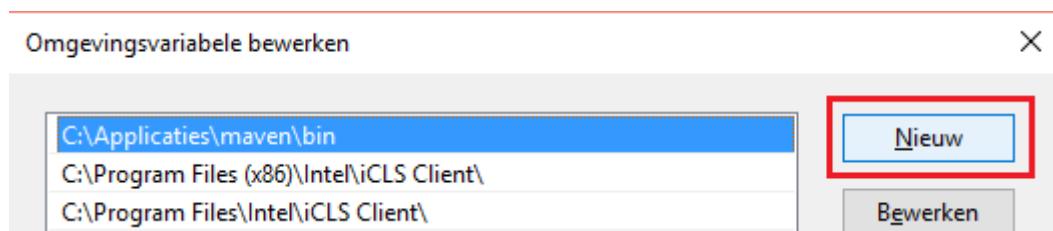
→ Omgevingsvariabelen



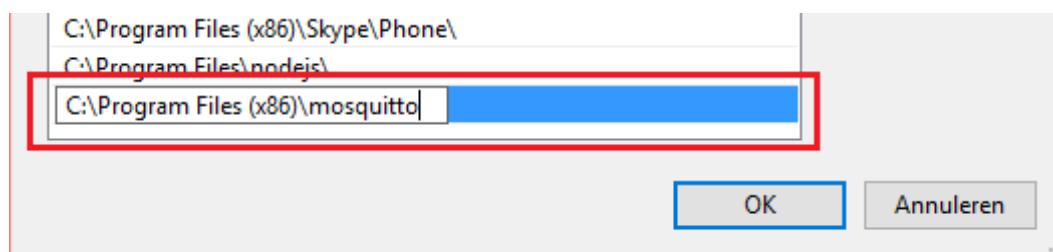
Selecteer Path → Bewerken...



→ Nieuw



Voeg C:\Program Files (x86)\mosquitto toe → OK (3 keer).



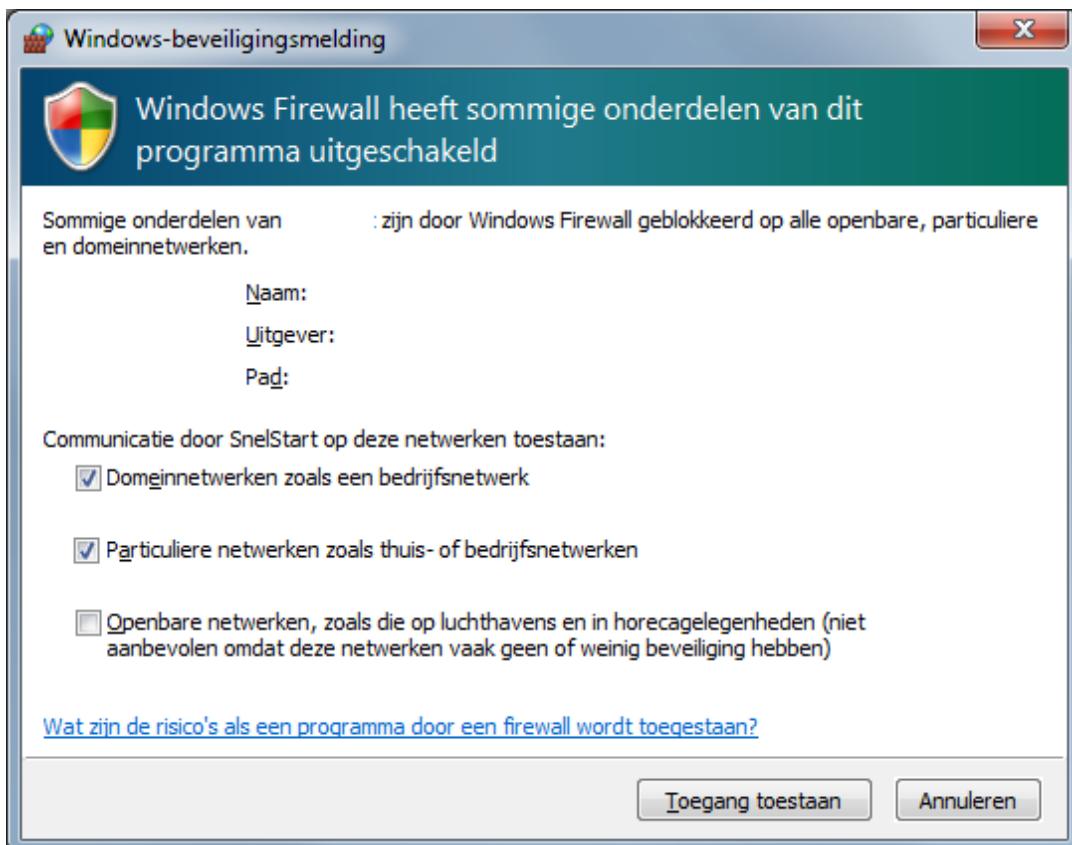
Part 1: The 35 euro IoT project

4.3.6.3 Broker

Open een command prompt met Administrator rechten → mosquitto

```
Administrator: Command - mosquitto
C:\temp>mosquitto
```

Indien er een firewall melding komt dan deze bevestigen.
Vink "Particuliere netwerken..." aan → Toegang toestaan



Open een nieuwe command prompt met Administrator rechten.

Geef als opdracht: netstat -an

```
Administrator: Command
C:\temp>netstat -an
```

Controleer of poort 1883 in gebruik is:

Part 1: The 35 euro IoT project

```
C:\temp>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0             LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0             LISTENING
  TCP    0.0.0.0:902            0.0.0.0:0             LISTENING
  TCP    0.0.0.0:912            0.0.0.0:0             LISTENING
  TCP    0.0.0.0:1883           0.0.0.0:0             LISTENING
  TCP    0.0.0.0:2801           0.0.0.0:0             LISTENING
  TCP    0.0.0.0:3306           0.0.0.0:0             LISTENING
```

Indien dit het geval is dan is de broker gestart.

4.3.6.4 Subscriber

Open een nieuwe command prompt met Administrator rechten.

Geef de opdracht: mosquitto_sub -t "#" -v

```
C:\temp>mosquitto_sub -t "#" -v
```

Hierbij abonneert de cliënt zich op alle topics (-t) door gebruik te maken van de wildcard #. Met de optie -v worden de ontvangen berichten getoond op de console.

4.3.6.5 Publisher

Open een nieuwe command prompt met Administrator rechten.

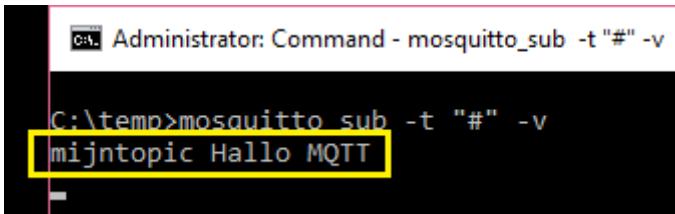
Geef de opdracht: mosquitto_pub -t "mijntopic" -m "Hallo MQTT" -h 127.0.0.1

```
C:\temp>mosquitto_pub -t "mijntopic" -m "Hallo MQTT" -h 127.0.0.1
C:\temp>
```

Hierbij wordt de boodschap "Hallo MQTT" (-m) gepubliceerd op topic mijntopic (-t) op host 127.0.0.1 (-h). De broker draait op dit moment op de localhost (127.0.0.1).

Part 1: The 35 euro IoT project

Op de command prompt waar de subscriber cliënt draait wordt het gepubliceerde bericht ontvangen:



```
Administrator: Command - mosquitto_sub -t "#" -v
C:\temp>mosquitto_sub -t "#" -v
mijntopic Hallo MQTT
```

4.3.7 Eclipse Java

4.3.7.1 Inleiding

Voor het testen en demonstreren van een MQTT cliënt wordt een Java programma gebruikt. Hiervoor is een Eclipse Java installatie nodig. Na de installatie wordt een Hello World programma gemaakt om de installatie te testen.

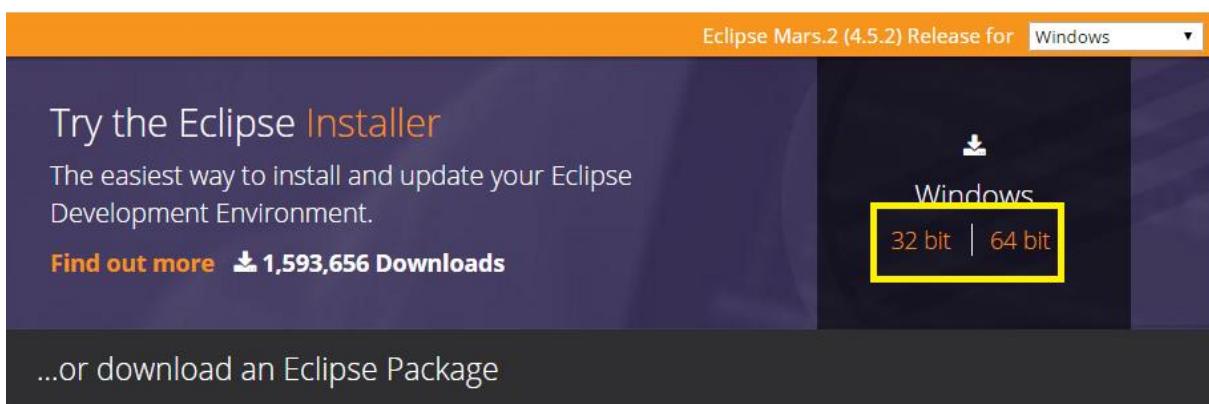
4.3.7.2 Installatie

Maak op een harde schijf een workspace directory aan, bijv: c:\workspace



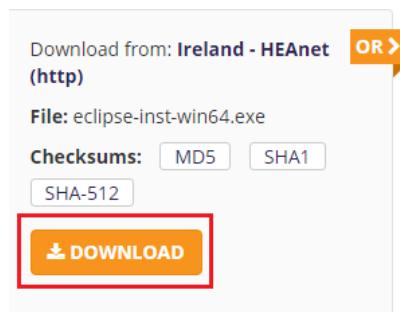
Open een browser en ga naar: <https://eclipse.org/downloads/>

Kies de juiste installer (32 of 64 bits):

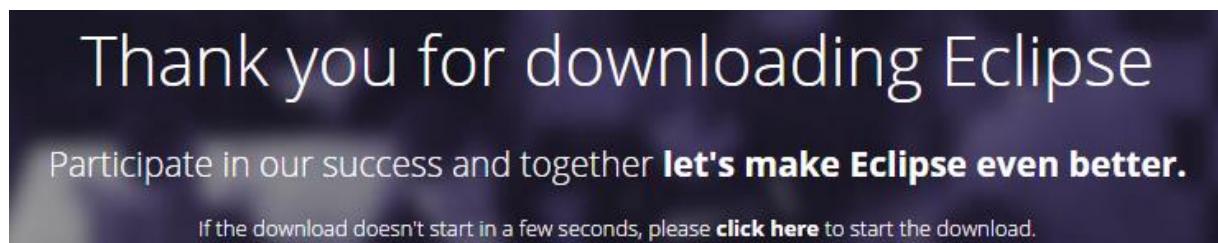


Part 1: The 35 euro IoT project

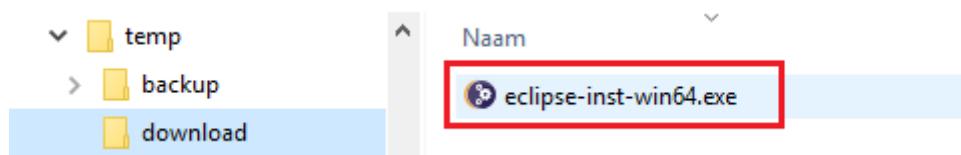
→ Download



Even geduld...



→ Dubbelklik met de muis op de installer



Even geduld...

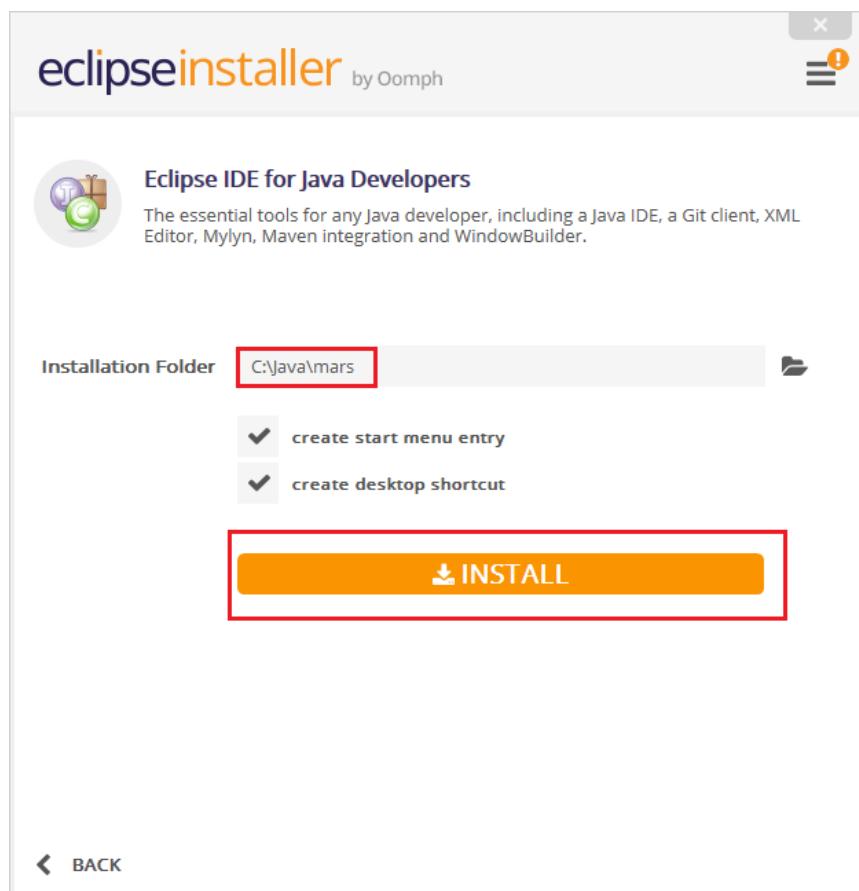


Part 1: The 35 euro IoT project

→ Selecteer Eclipse IDE for Java Developers

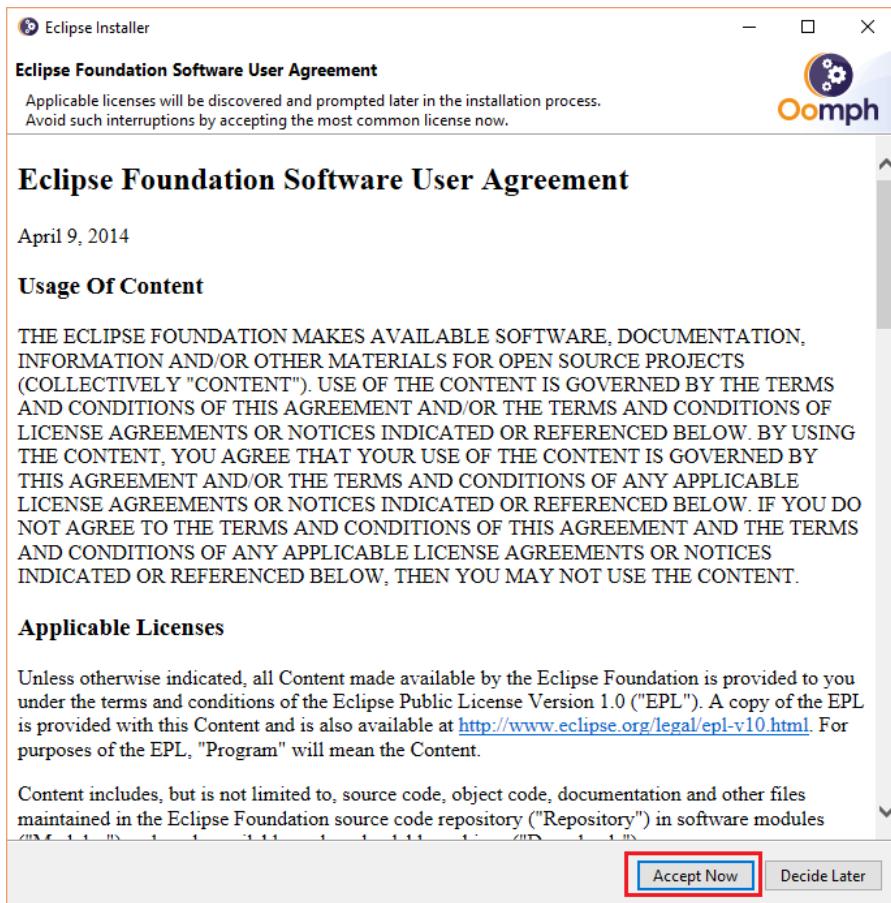


Kies een eigen installatie folder (zie voorbeeld) of neem de default folder over → Install

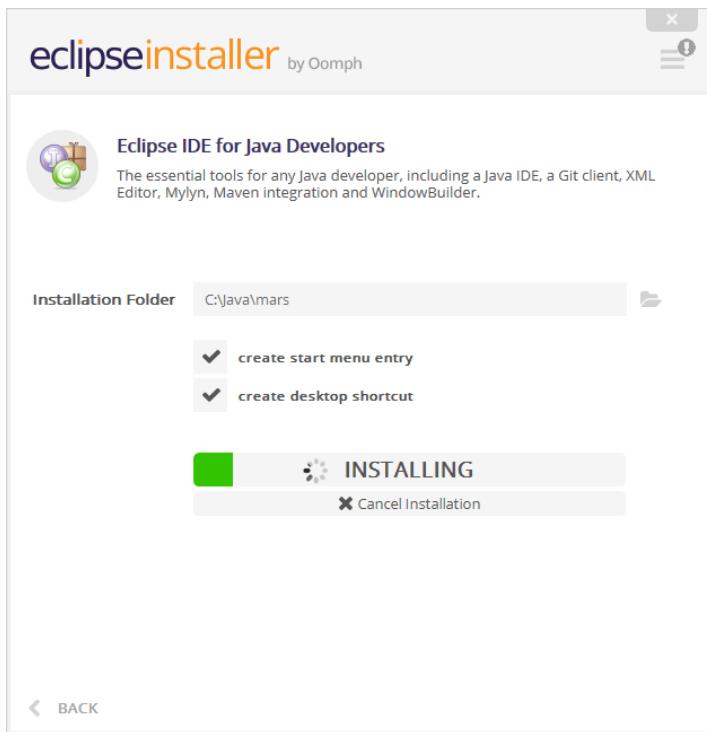


Part 1: The 35 euro IoT project

→ Accept now



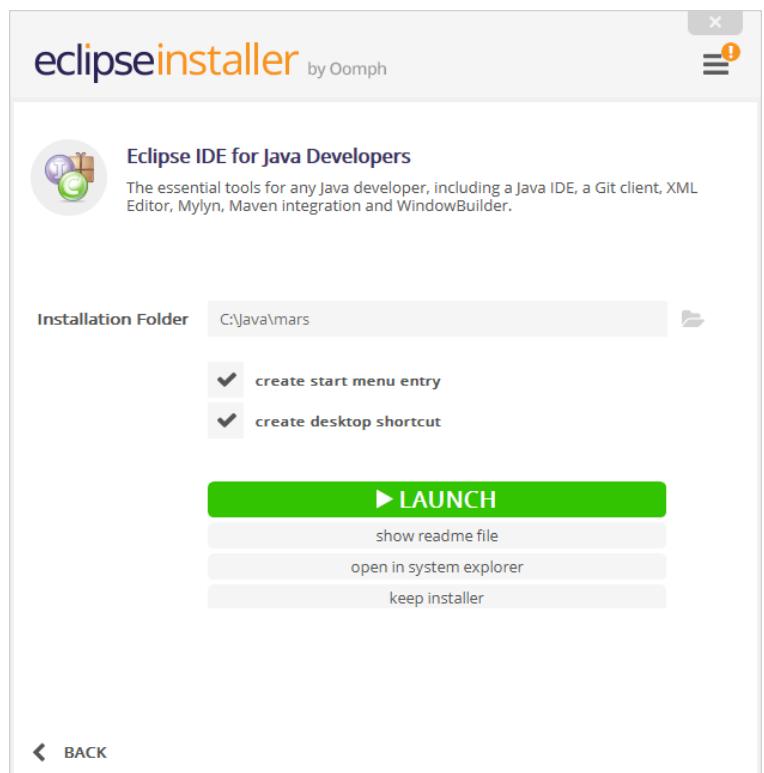
Even geduld...



Part 1: The 35 euro IoT project

4.3.7.3 Java Hello World

→ Launch

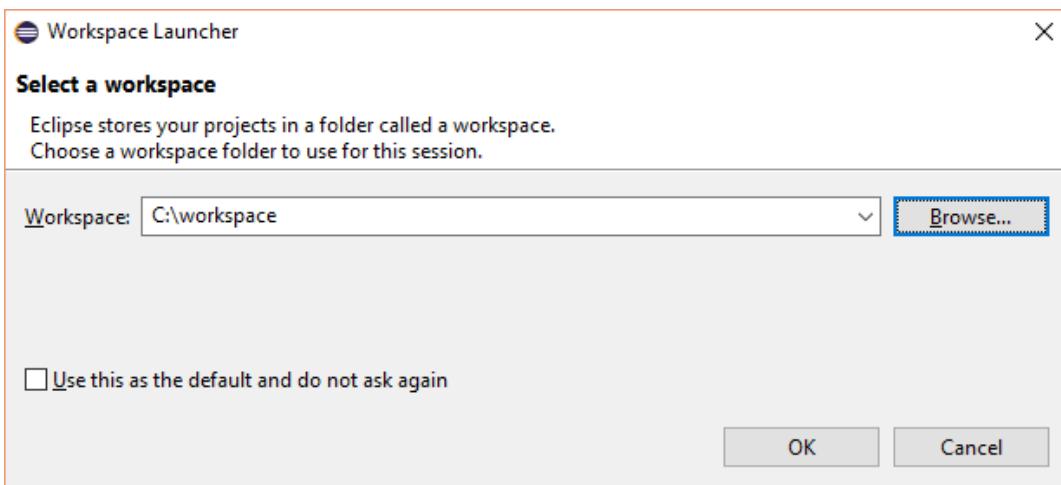


Even geduld...

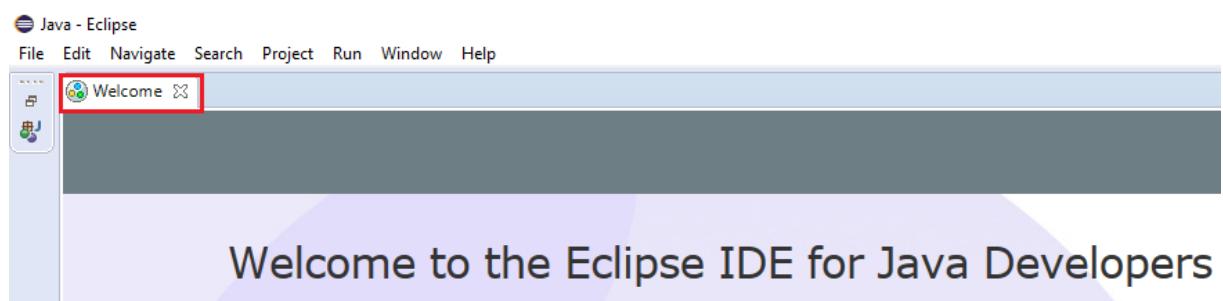


Part 1: The 35 euro IoT project

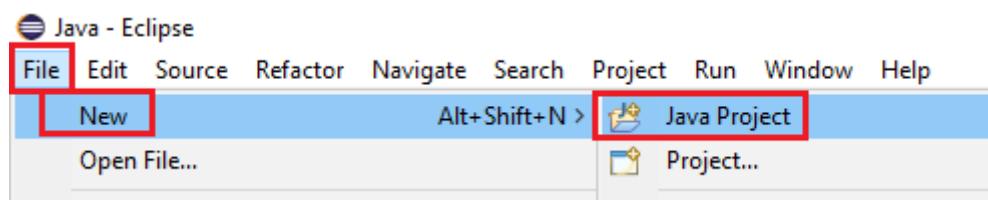
Selecteer de eerder aangemaakte workspace → Browse → c:\workspace → OK



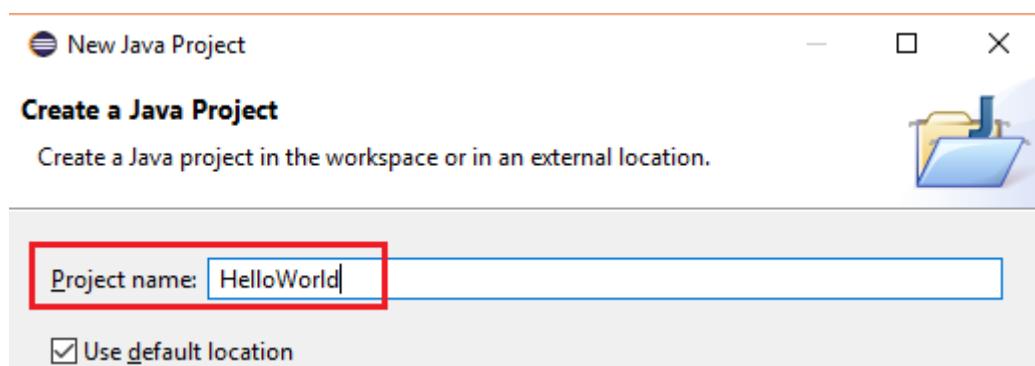
Sluit Welcome scherm



Menu → File → New → Java Project

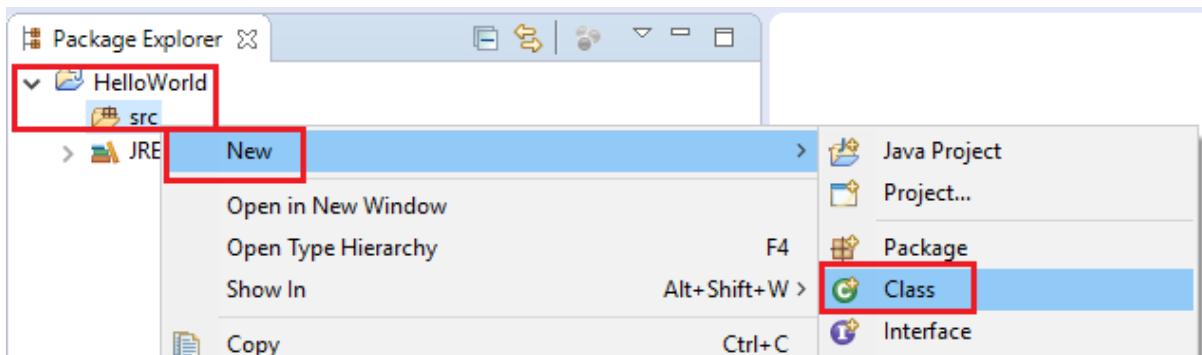


Project name: HelloWorld → Finish

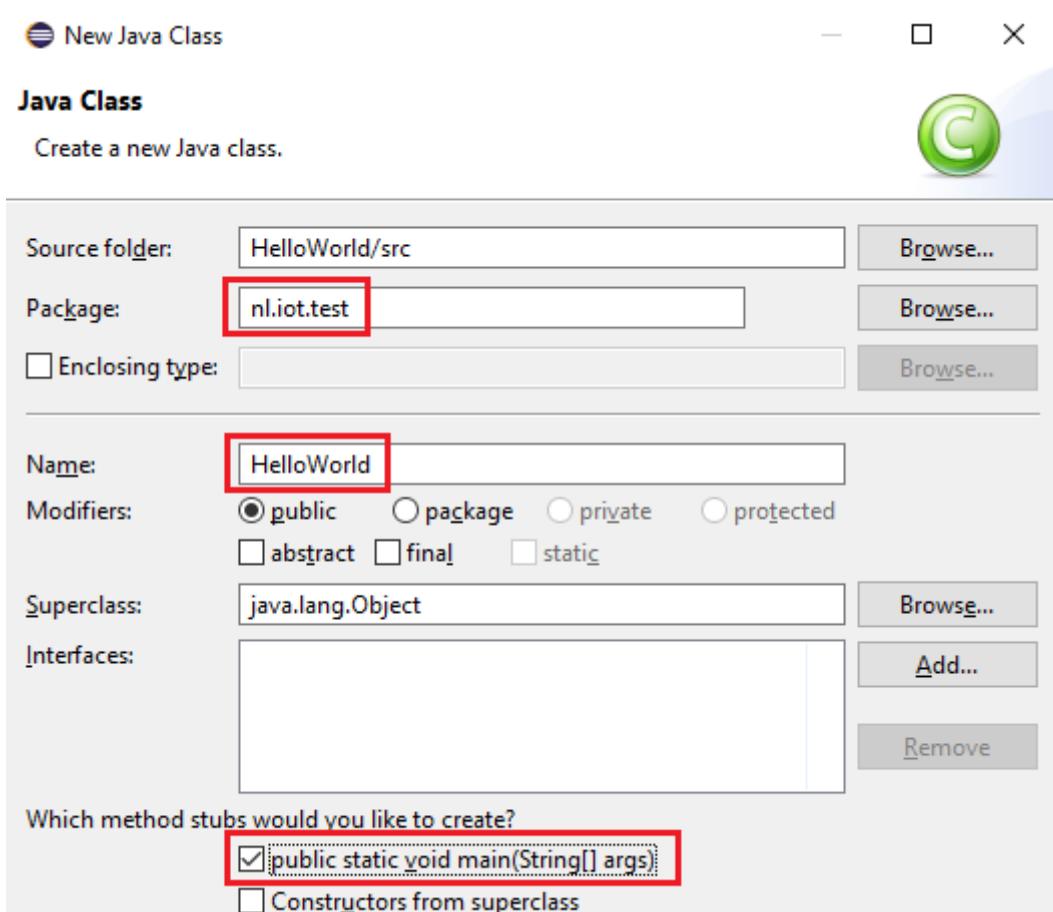


Klap het project in de Package Explorer uit → src → rechter muisknop → New → Class

Part 1: The 35 euro IoT project



Package: nl.iot.test
Name: HelloWorld
Vink "public static void main..." aan → Finish



Part 1: The 35 euro IoT project

Wijzig de source naar:

```
package nl.iot.test;

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello IOT World");

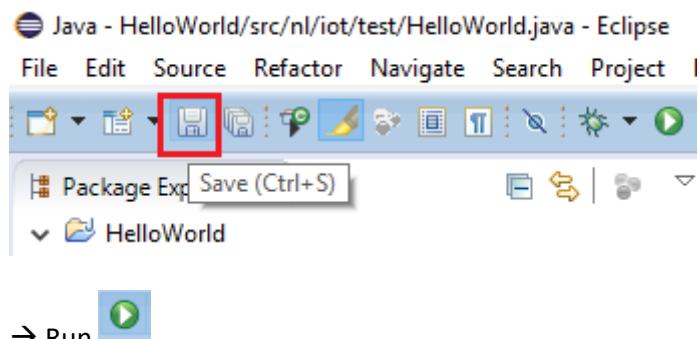
    }

}
```

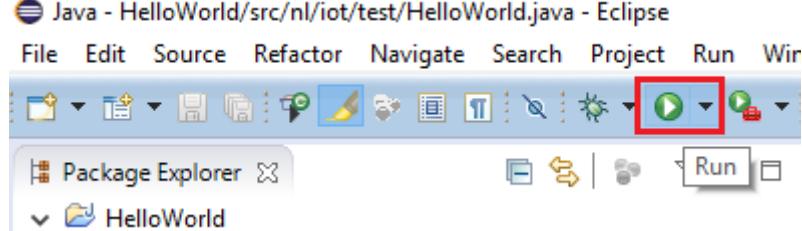
Zie: Sources → Java → HelloWorld → src → nl → test → HelloWorld.java

→ Save 

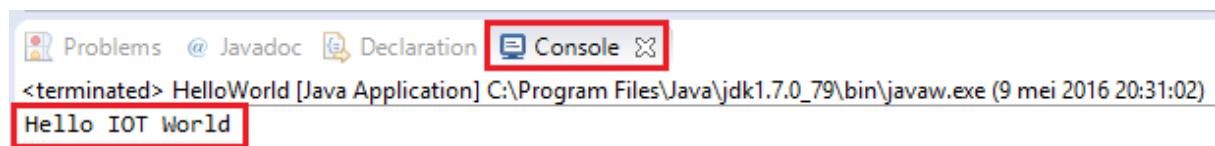
De source bestaat uit de klasse HelloWorld. Deze klasse bevat één statische functie main. Dit is het zogenaamde main entrypoint van de Java applicatie. Via de statische klasse System wordt de boodschap "Hello IOT World" getoond op de console. Deze laatste klasse maakt standaard deel uit van de Java Runtime omgeving en hoeft niet expliciet te worden geïmporteerd.



→ Run 



Resultaat is rechtsonder in de Console te zien:



Na de installatie van Paho (volgende paragraaf) wordt Eclipse gebruikt om een MQTT cliënt te maken.

Part 1: The 35 euro IoT project

4.3.8 PAHO Java cliënt

4.3.8.1 Inleiding

Voor ondermeer Java is een MQTT implementatie gemaakt met de naam Paho. Pāho komt uit het Maori dialect en betekent zoveel als verzenden, bekend maken en versturen. Na de installatie wordt een Java MQTT cliënt gemaakt om Paho in combinatie met de Mosquitto broker te testen.

4.3.8.2 Installatie

De MQTT Java cliënt library wordt in dit boek gebruikt voor de Android app en Java cliënt.

Open een browser en ga naar:

[https://repo.eclipse.org/content/repositories/paho-releases/org/eclipse/paho/org.eclipse.paho.client.mqttv3/1.0.2/ *\)](https://repo.eclipse.org/content/repositories/paho-releases/org/eclipse/paho/org.eclipse.paho.client.mqttv3/1.0.2/)

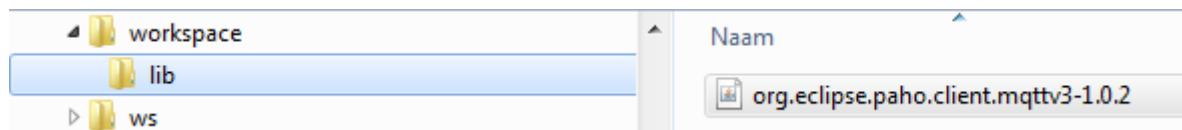
*) Selecteer bij het downloaden voor de laatste versie (in dit voorbeeld 1.0.2).

Klik met de muis op `org.eclipse.paho.client.mqttv3-1.0.2.jar` om deze te downloaden.

| Name | Last Modified | Size |
|---|------------------------------|--------|
| Parent Directory | | |
| org.eclipse.paho.client.mqttv3-1.0.2-p2artifacts.xml | Wed Feb 18 12:11:31 EST 2015 | 1314 |
| org.eclipse.paho.client.mqttv3-1.0.2-p2artifacts.xml.md5 | Wed Feb 18 12:11:32 EST 2015 | 32 |
| org.eclipse.paho.client.mqttv3-1.0.2-p2artifacts.xml.sha1 | Wed Feb 18 12:11:32 EST 2015 | 40 |
| org.eclipse.paho.client.mqttv3-1.0.2-p2metadata.xml | Wed Feb 18 12:11:31 EST 2015 | 2423 |
| org.eclipse.paho.client.mqttv3-1.0.2-p2metadata.xml.md5 | Wed Feb 18 12:11:31 EST 2015 | 32 |
| org.eclipse.paho.client.mqttv3-1.0.2-p2metadata.xml.sha1 | Wed Feb 18 12:11:31 EST 2015 | 40 |
| org.eclipse.paho.client.mqttv3-1.0.2-pack200.jar.pack.gz | Wed Feb 18 12:11:31 EST 2015 | 67733 |
| org.eclipse.paho.client.mqttv3-1.0.2-pack200.jar.pack.gz.md5 | Wed Feb 18 12:11:31 EST 2015 | 32 |
| org.eclipse.paho.client.mqttv3-1.0.2-pack200.jar.pack.gz.sha1 | Wed Feb 18 12:11:31 EST 2015 | 40 |
| org.eclipse.paho.client.mqttv3-1.0.2.jar | Wed Feb 18 12:11:31 EST 2015 | 171372 |
| org.eclipse.paho.client.mqttv3-1.0.2.jar.md5 | Wed Feb 18 12:11:31 EST 2015 | 32 |
| org.eclipse.paho.client.mqttv3-1.0.2.jar.sha1 | Wed Feb 18 12:11:31 EST 2015 | 40 |
| org.eclipse.paho.client.mqttv3-1.0.2.pom | Wed Feb 18 12:11:31 EST 2015 | 3498 |
| org.eclipse.paho.client.mqttv3-1.0.2.pom.md5 | Wed Feb 18 12:11:31 EST 2015 | 32 |
| org.eclipse.paho.client.mqttv3-1.0.2.pom.sha1 | Wed Feb 18 12:11:31 EST 2015 | 40 |

Part 1: The 35 euro IoT project

Open een verkenner en ga naar de eerder aangemaakte workspace voor Java. Maak een directory lib aan en kopieer de het gedownloade jar-bestand hierin.



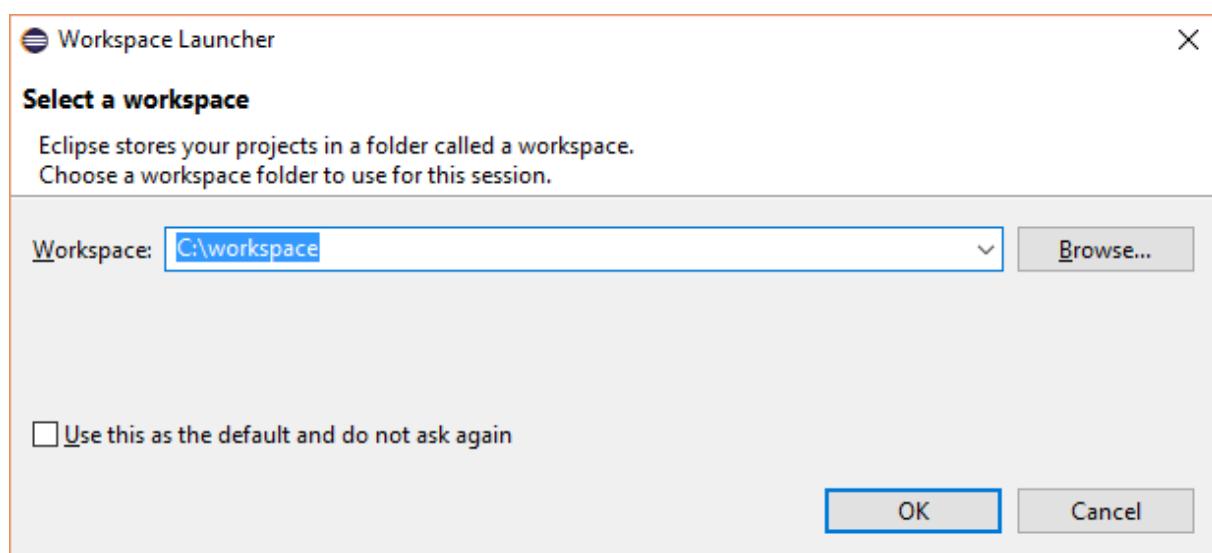
Zie: Sources → Java → lib → org.eclipse.paho.client.mqttv3-1.0.2.jar

4.3.8.3 Java MQTT cliënt

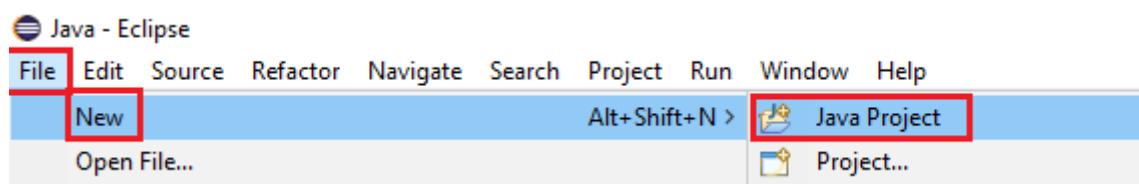
Start Eclipse via het icoontje op de desktop.



→ OK

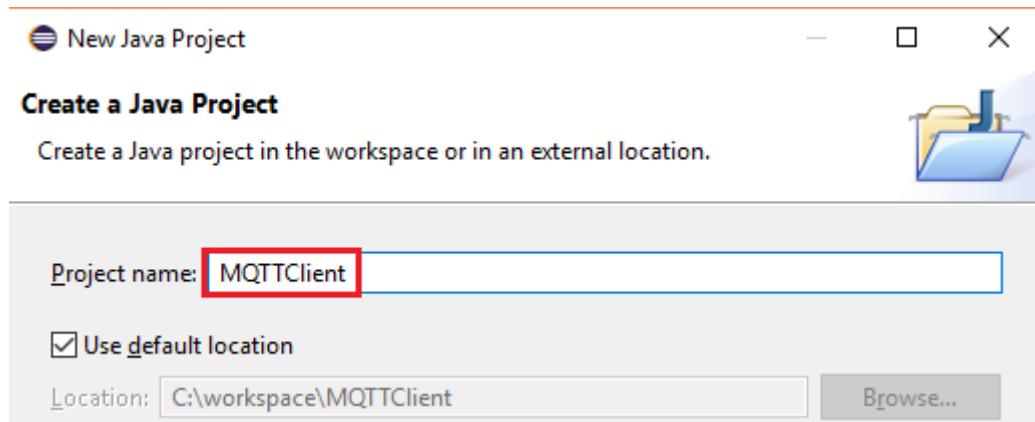


→ File → New → Java Project

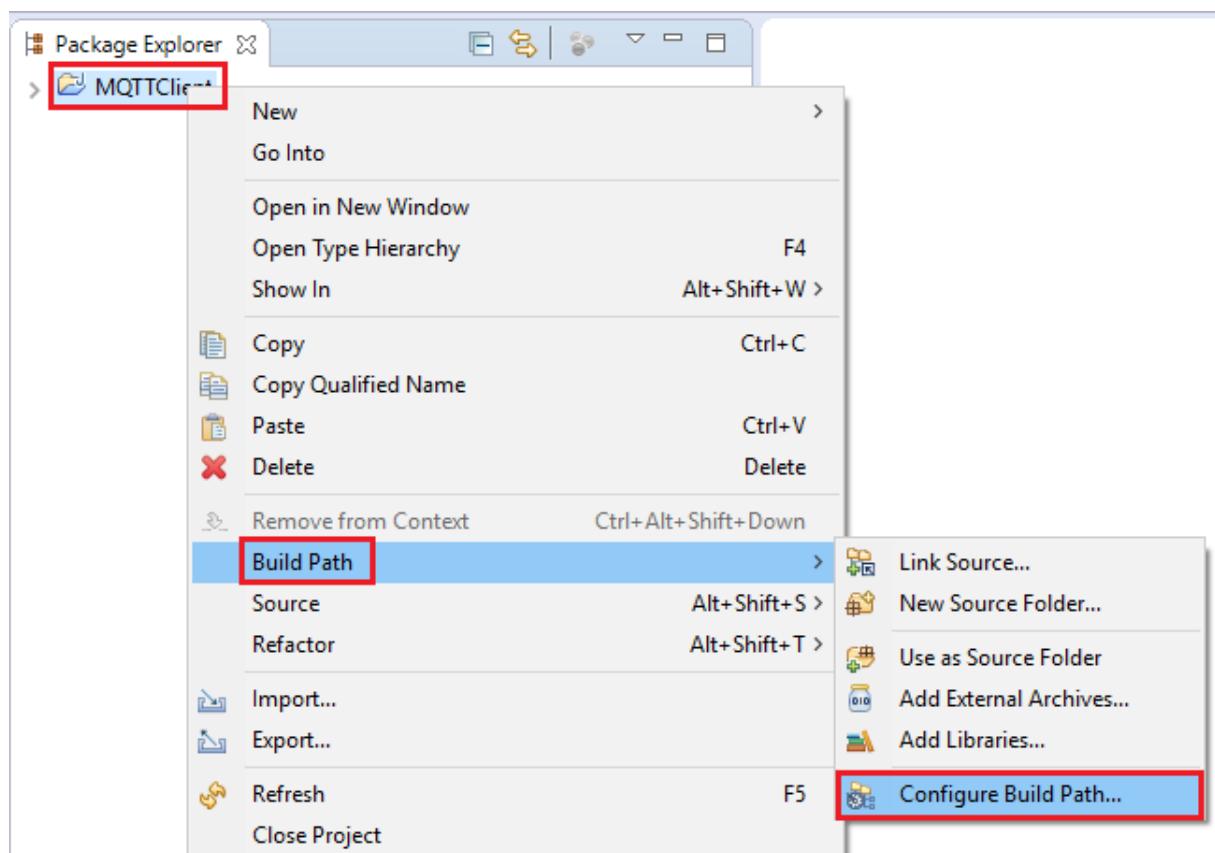


Part 1: The 35 euro IoT project

Project Name: MQTTClient → Finish

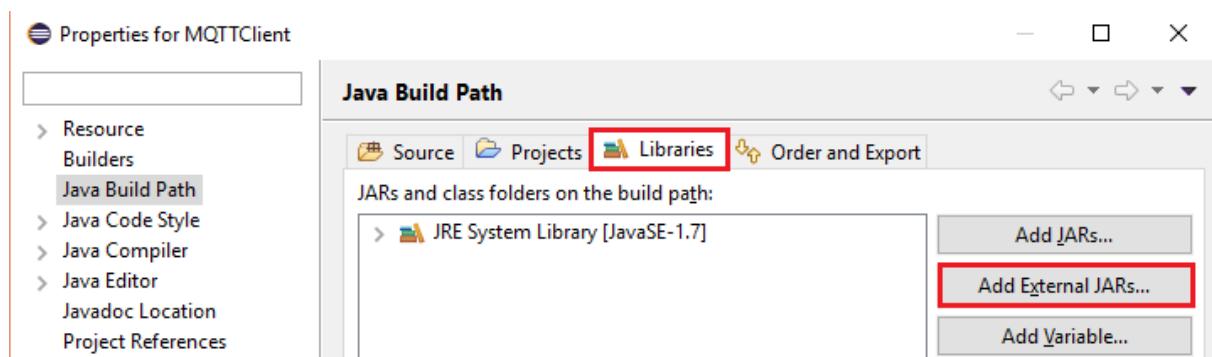


Project → rechter muisknop → Build Path → Configure Build Path...

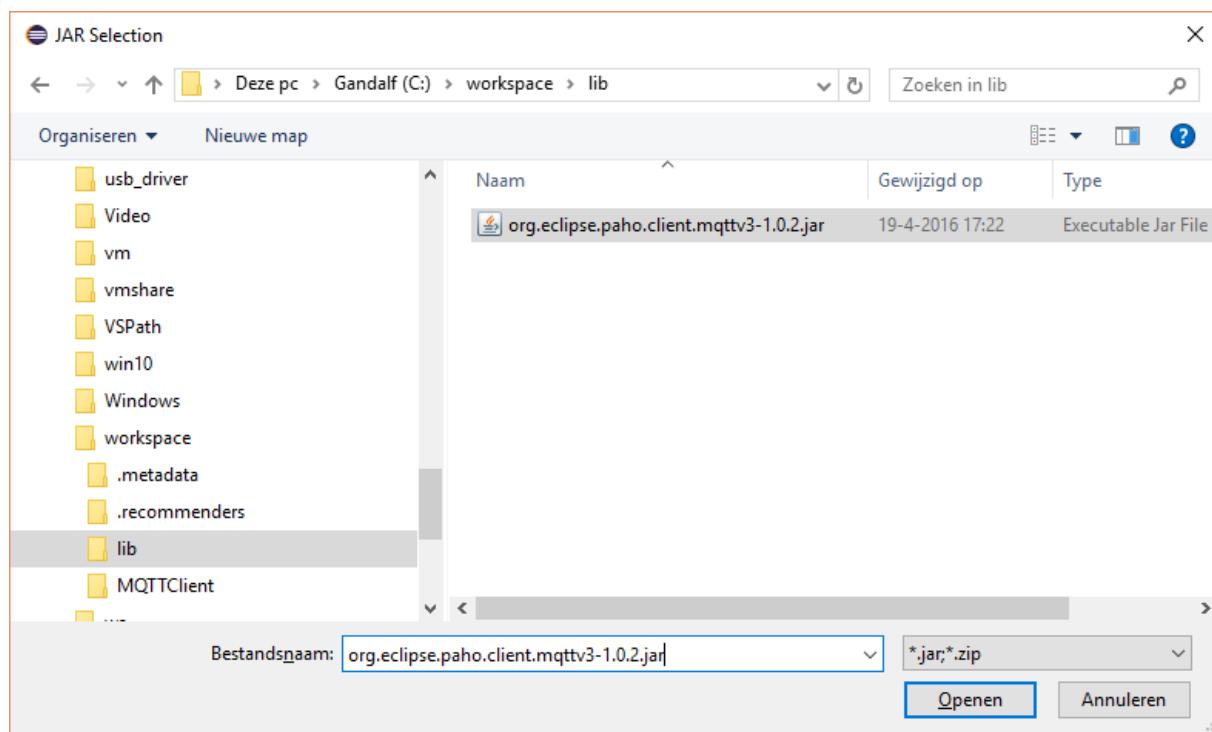


Part 1: The 35 euro IoT project

Tab: Libraries → Add External JARS...

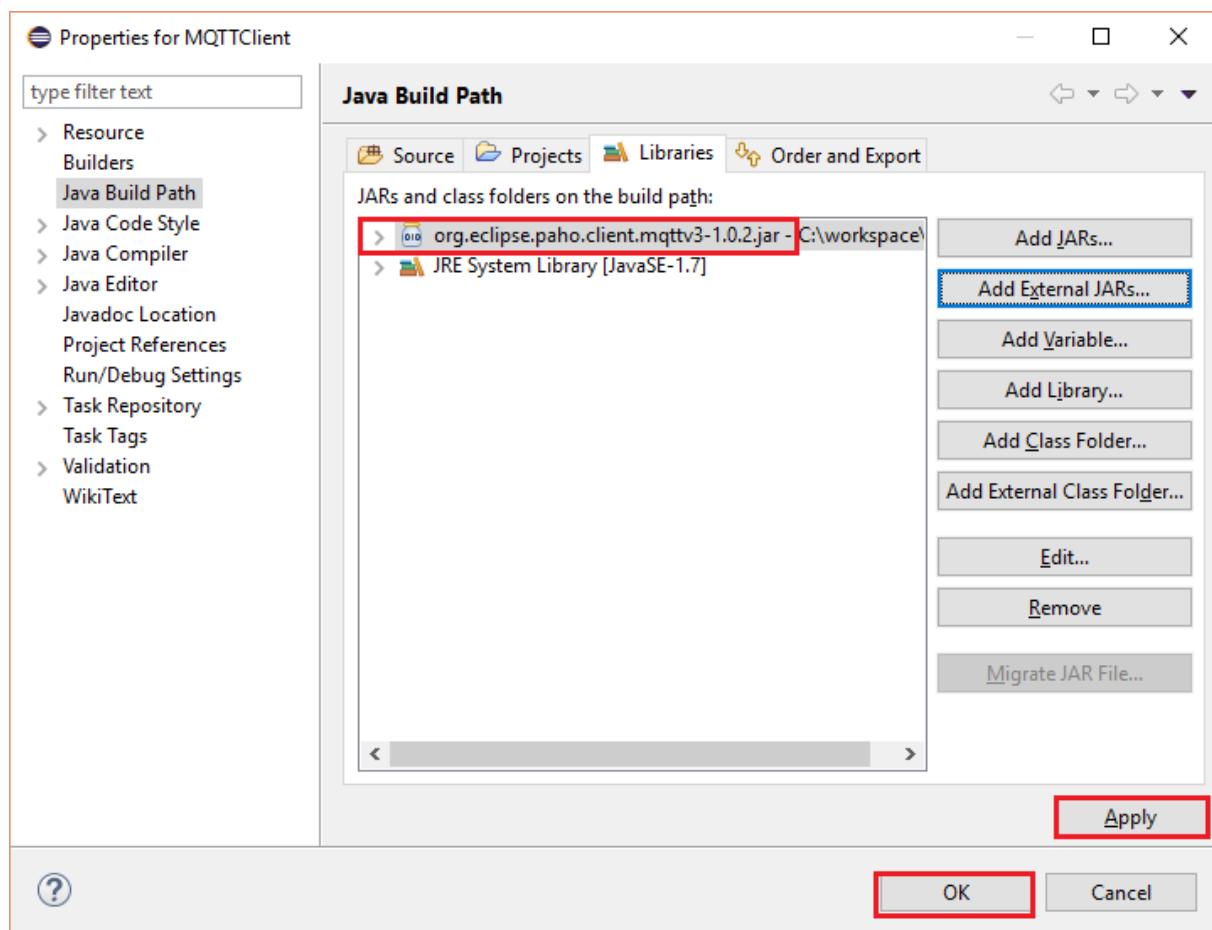


Selecteer org.eclipse.paho.client.mqttv3-1.0.2.jar in c:\workspace\lib → Openen

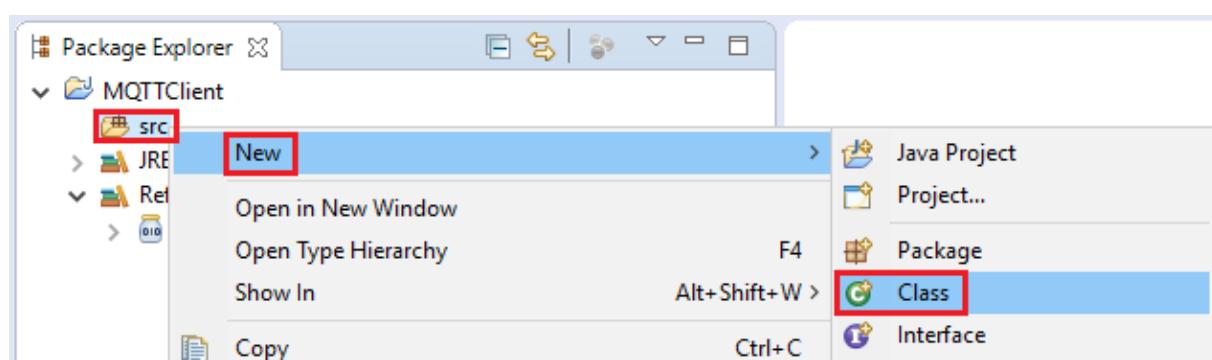


Part 1: The 35 euro IoT project

→ Apply → OK

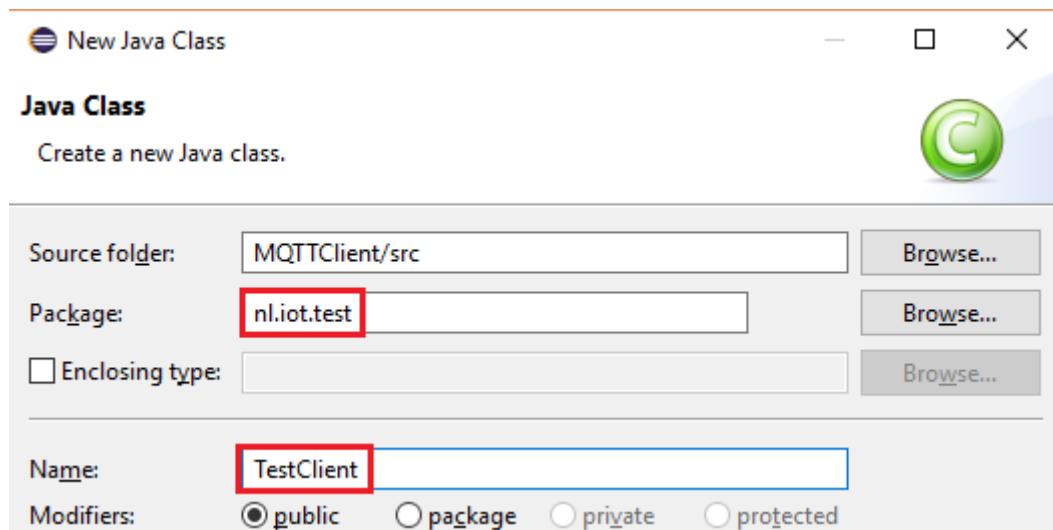


Klap het project uit → src → rechter muisknop → New → Class



Part 1: The 35 euro IoT project

Package: nl.iot.test
Name: TestClient
→ Finish



Zie: Sources → Java → MQTTClient → src → nl → iot → test → TestClient.java

TestClient.java

```
package nl.iot.test;

import java.text.DateFormat;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.concurrent.ThreadLocalRandom;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public class TestClient {
    // IP Address
    private String ipAddress = "127.0.0.1";

    // Default port
    private int port = 1883;

    MqttClient client;

    public TestClient() {
    }

    public static void main(String[] args) {
        new TestClient().doTests();
    }
}
```

Part 1: The 35 euro IoT project

```
public void doTests() {
    String clientIP = "tcp://" + ipAddress + ":" + port;
    System.out.println("Start client");
    try {
        client = new MqttClient(clientIP, "mqttDHT22");
        client.connect();
        while (true) {
            String jsonMessage = this.getJSONString();
            MqttMessage message = new MqttMessage();
            message.setPayload(jsonMessage.getBytes());
            System.out.println("Publish");
            client.publish("mqttDHT22/payload", message);
            Thread.sleep(5000);
        }
    } catch (InterruptedException e) {
        System.err.println("InterruptedException!");
        System.err.println(e.getMessage());
    } catch (MqttException e) {
        System.err.println("MqttException!");
        e.printStackTrace();
    } finally {
        try {
            client.disconnect();
        } catch (MqttException e) {
            System.err.println("MqttException!");
            e.printStackTrace();
        }
    }
}

public String getJSONString() {
    String jsonMessage = "{\"dht22\":{\"datetime\":\"" +
        this.getCurrentDate();
    jsonMessage += "\",\"temperature\":\"", +
        this.getRandomTemperature(15.0, 32.0);
    jsonMessage += "\",\"humidity\":\"", +
        this.getRandomHumidity(30, 70);
    jsonMessage += "\"}}";
    System.out.println(jsonMessage);
    return jsonMessage;
}

public String getCurrentDate() {
    DateFormat df = new SimpleDateFormat("yyyyMMddHHmmss");
    Date date = new Date();
    return df.format(date);
}

public String getRandomTemperature(double min, double max) {
    DecimalFormat df = new DecimalFormat("#.#");
    double temp = ThreadLocalRandom.current().nextDouble(min, max);
    return df.format(temp).replace(',', '.');
}

public int getRandomHumidity(int min, int max) {
    return ThreadLocalRandom.current().nextInt(min, max);
}
}
```

Part 1: The 35 euro IoT project

In deze cliënt wordt een bericht gepubliceerd op de topic mqttDHT22/payload. Het bericht wordt opgeleverd in het formaat dat straks ook zal worden gebruikt als bericht voor de AngularJS en Android app.

De source begint met het importeren van de benodigde Java libraries. Naast de cliënt library van MQTT gaat het ook om libraries voor het formateren van de datum en tijd en het ophalen van een random waarde.

In het main entrypoint wordt de test cliënt aangeroepen. De test cliënt begint met het maken van een connectie naar de MQTT-broker. Daarna wordt er om de 5 seconden een bericht gepubliceerd naar de broker.

Voor het bericht worden random waarden gebruikt voor temperatuur en luchtvochtigheid.

Het bericht wordt gepubliceerd in het zogenaamde JSON-formaat. Het bericht heeft als formaat:

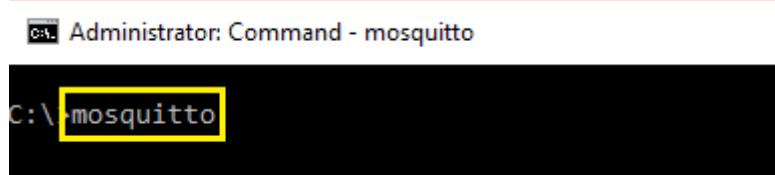
```
{"dht22":  
  {  
    "datetime": "20160328180535",  
    "temperature": "20.4",  
    "humidity": "42"  
  }  
}
```

De inhoud spreekt min of meer voor zichzelf. Het bevat een datum/tijdstip, de temperatuur in graden Celcius en het luchtvochtigheidspercentage. De begin tag dht22 slaat op de gebruikte sensor.

4.3.8.4 Testen cliënt.

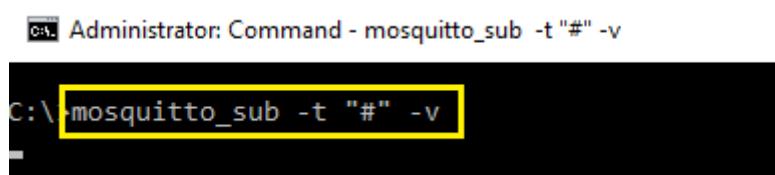
Start de mosquitto broker en subscriber (zoals al eerder beschreven).

De broker: mosquitto



```
C:\>mosquitto
```

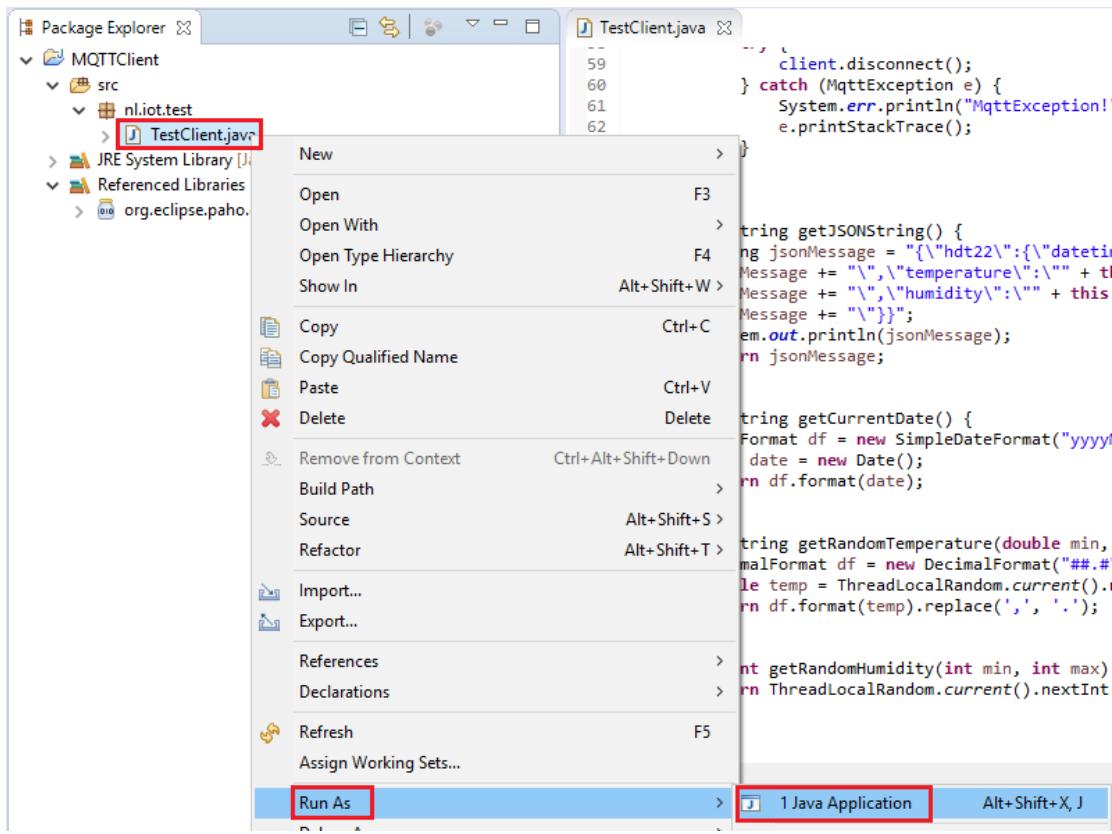
De subscriber: mosquitto_sub -t "#" -v



```
C:\>mosquitto_sub -t "#" -v
```

Part 1: The 35 euro IoT project

Ga terug naar het Eclipse project en voer het project uit met TestClient.java → rechter muisknop → Run As → Java Application



De volgende uitvoer komt op de Console:

```
<terminated> TestClient [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (8 jun. 2016 12:07:17)
Start client
{"dht22":{"datetime":"20160608120718","temperature":"28.9","humidity":"50"}}
Publish
{"dht22":{"datetime":"20160608120723","temperature":"27.1","humidity":"62"}}
Publish
{"dht22":{"datetime":"20160608120728","temperature":"30.9","humidity":"53"}}
```

De gepubliceerde berichten worden door de subscriber ontvangen en getoond:

```
C:\>mosquitto_sub -t "#" -v
C:\>mosquitto_sub -t "#" -v
mqttDHT22/payload {"dht22":{"datetime":"20160608120718","temperature":"28.9","humidity":"50"}}
mqttDHT22/payload {"dht22":{"datetime":"20160608120723","temperature":"27.1","humidity":"62"}}
mqttDHT22/payload {"dht22":{"datetime":"20160608120728","temperature":"30.9","humidity":"53"}}
mqttDHT22/payload {"dht22":{"datetime":"20160608120733","temperature":"19.5","humidity":"64"}}
```

Stop het publiceren door op het stop icoontje in de Console te klikken.



Part 1: The 35 euro IoT project

4.3.9 Android Studio 2.x

4.3.9.1 Inleiding

Het installeren van Android Studio 2.x bestaat uit twee delen:

- Installeren van de Java Development Kit 1.8
- Installeren van Android Studio 2.x

De huidige versie (mei 2016) van Android Studio is 2.1.

4.3.9.2 Java JDK 1.8

Open een browser en ga naar:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Download de laatste versie van de JDK:

Vink "Accept Licence Agreement" aan.



Selecteer de juiste versie (32 of 64 bits):

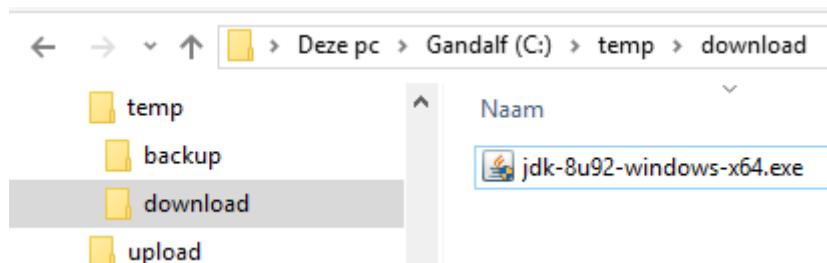
The screenshot shows the "Java SE Development Kit 8u92" download page. It features a table with columns for Product / File Description, File Size, and Download. The table lists various operating system and architecture combinations. A red box highlights the "Windows x64" row, specifically the "Download" link which is "jdk-8u92-windows-x64.exe".

| Product / File Description | File Size | Download |
|-------------------------------------|-----------|---|
| Linux x86 | 160.26 MB | jdk-8u92-linux-i586.rpm |
| Linux x86 | 174.94 MB | jdk-8u92-linux-i586.tar.gz |
| Linux x64 | 158.27 MB | jdk-8u92-linux-x64.rpm |
| Linux x64 | 172.99 MB | jdk-8u92-linux-x64.tar.gz |
| Mac OS X | 227.32 MB | jdk-8u92-macosx-x64.dmg |
| Solaris SPARC 64-bit (SVR4 package) | 139.47 MB | jdk-8u92-solaris-sparcv9.tar.Z |
| Solaris SPARC 64-bit | 98.93 MB | jdk-8u92-solaris-sparcv9.tar.gz |
| Solaris x64 (SVR4 package) | 140.35 MB | jdk-8u92-solaris-x64.tar.Z |
| Solaris x64 | 96.76 MB | jdk-8u92-solaris-x64.tar.gz |
| Windows x86 | 188.43 MB | jdk-8u92-windows-i586.exe |
| Windows x64 | 193.66 MB | jdk-8u92-windows-x64.exe |

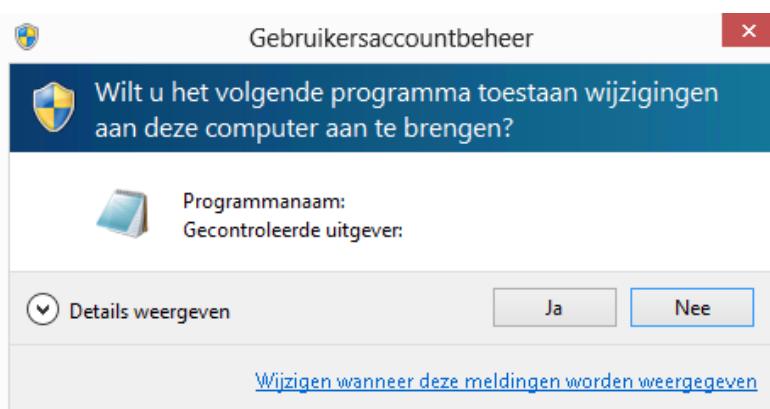
Klik op de juiste versie en wacht totdat het bestand gedownload is.

Part 1: The 35 euro IoT project

Dubbelklik op de download om de installatie te starten.



→ Ja

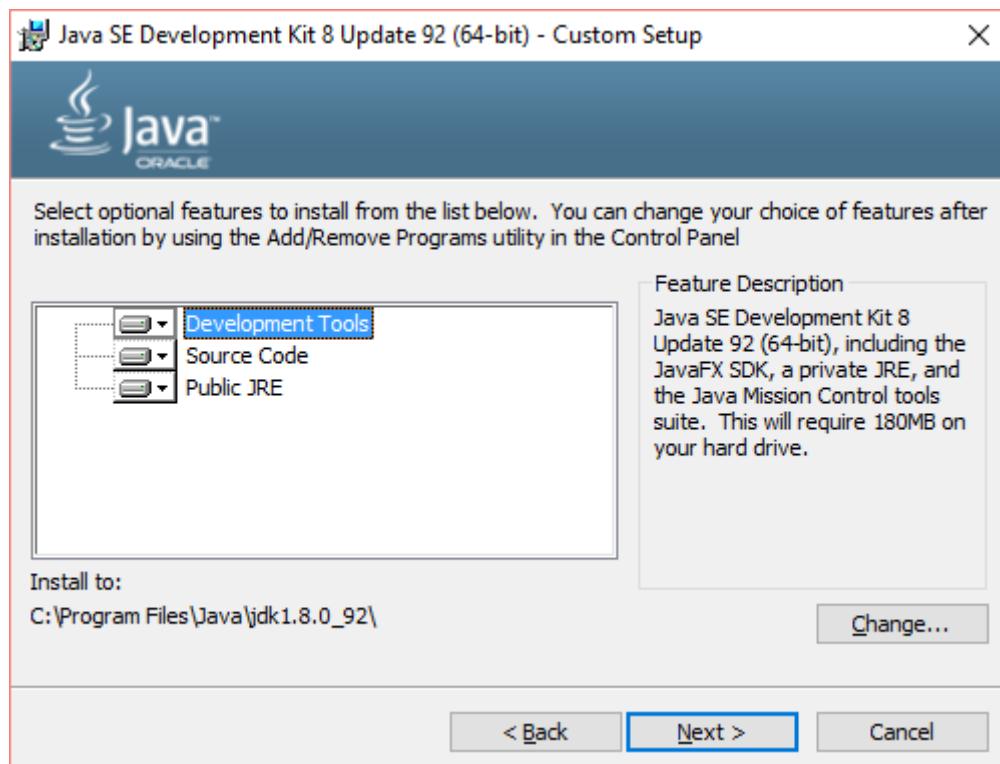


→ Next

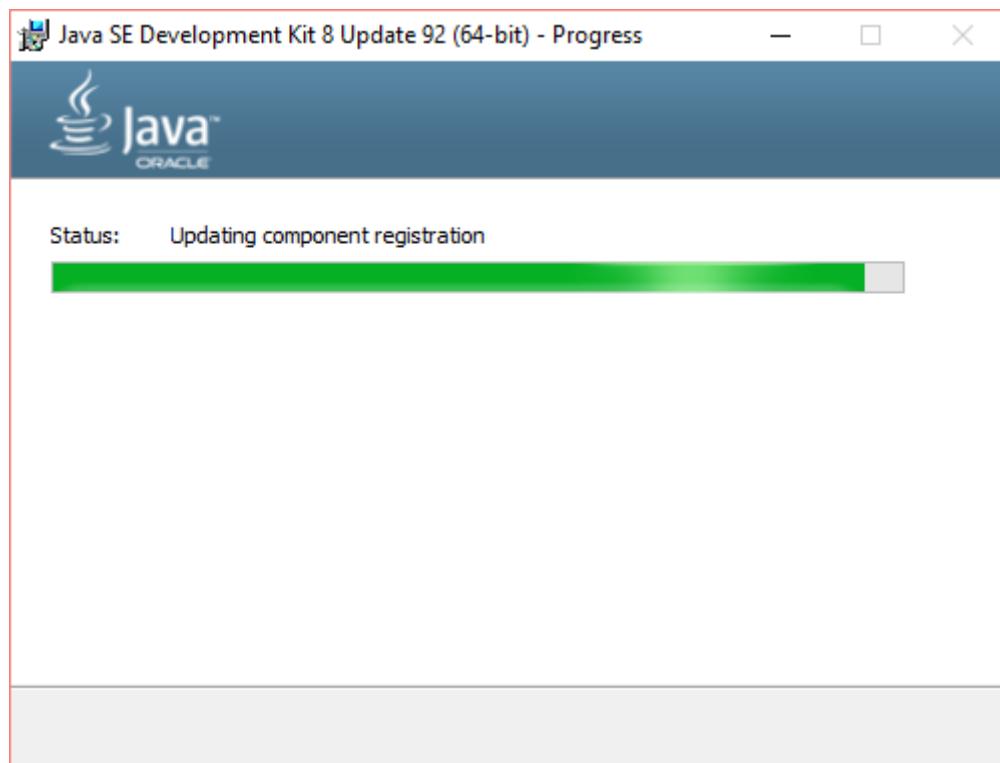


Part 1: The 35 euro IoT project

→ Next

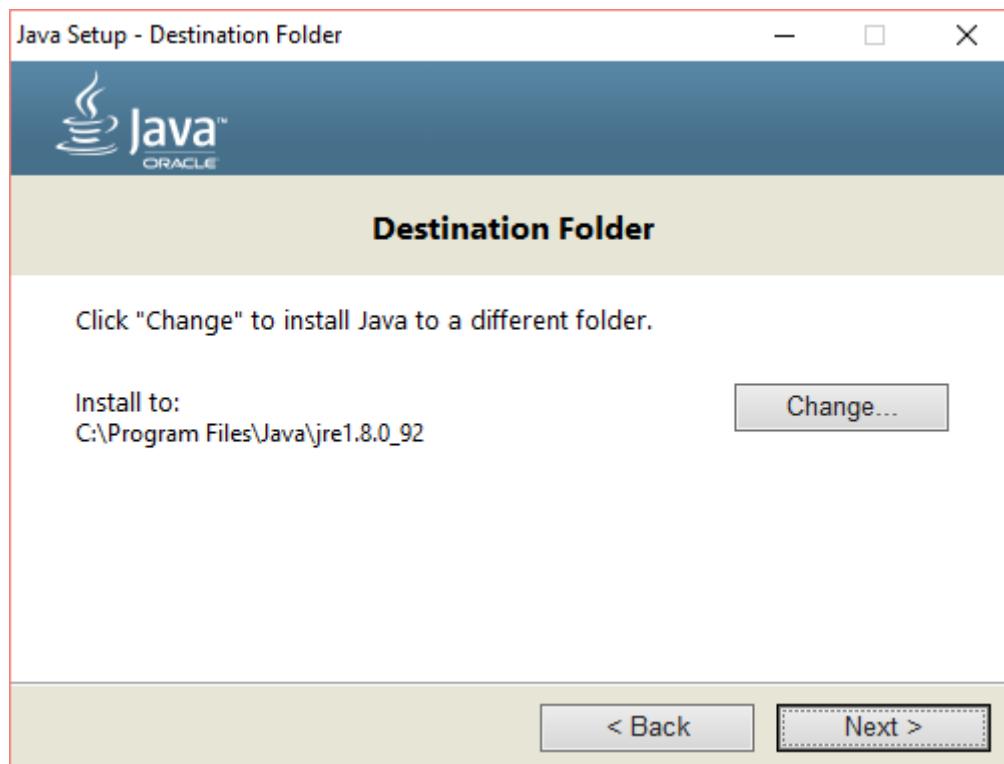


Even geduld...



Part 1: The 35 euro IoT project

→ Next

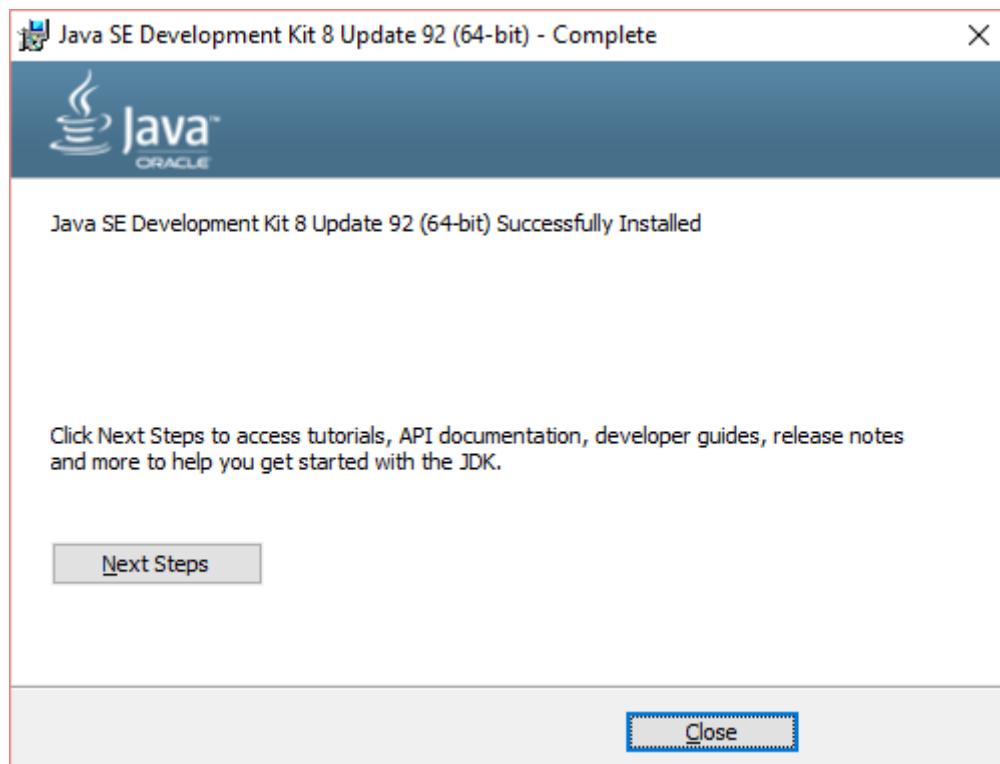


Even geduld...

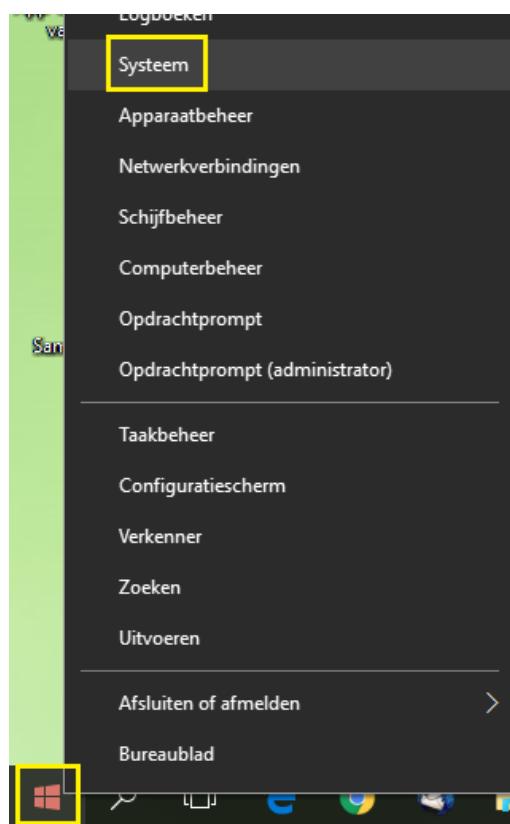


Part 1: The 35 euro IoT project

→ Close

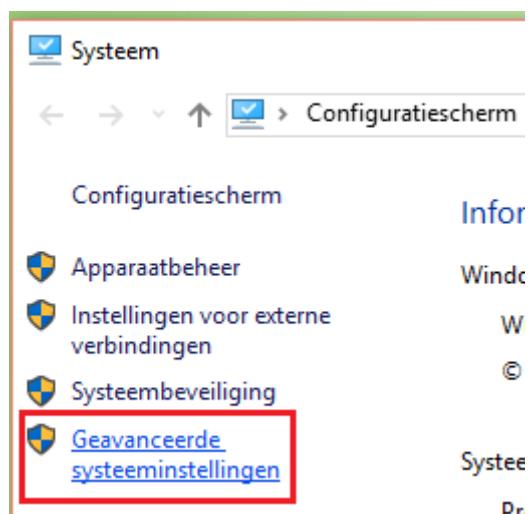


Start → rechtermuisknop → Systeem

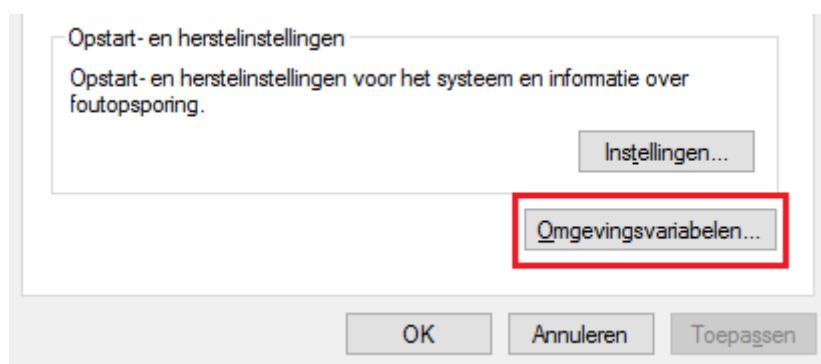


Part 1: The 35 euro IoT project

Geavanceerde systeeminstellingen



→ Omgevingsvariabelen

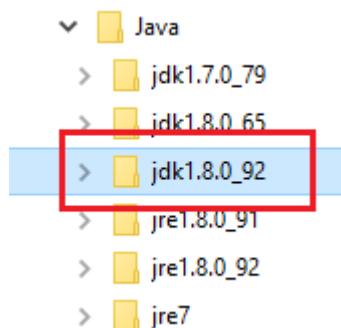


→ Nieuw

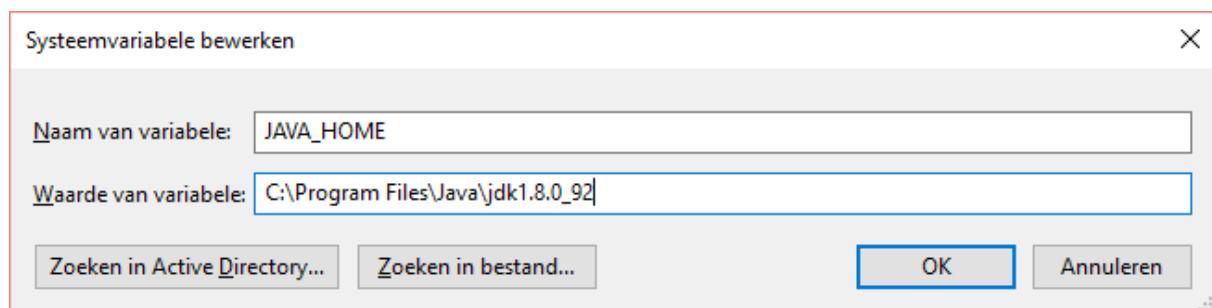
| Systeemvariabelen | |
|------------------------|--|
| Variabele | Waarde |
| OPENSSL_CONF | C:\Applicaties\OpenSSL-Win32\bin\openssl.cfg |
| OS | Windows_NT |
| Path | C:\Applicaties\maven\bin;C:\Program Files (x86)\Intel\iCLS Client\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\WindowsPowerShell\v1.0\Microsoft.PowerShell_profile.ps1;.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC |
| PATHEXT | .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC |
| PROCESSOR_ARCHITECTURE | AMD64 |
| PROCESSOR_IDENTIFIER | Intel64 Family 6 Model 58 Stepping 9, GenuineIntel |
| PROCESSOR_LEVEL | 6 |

Part 1: The 35 euro IoT project

Maak een nieuwe systeemvariabele JAVA_HOME aan met als waarde de geïnstalleerde versie van de JDK. Zie c:\program files\java:



Nieuwe variabele wordt dan:



→ OK (3 keer).

4.3.9.3 Android Studio

Open een browser en ga naar:
<http://developer.android.com/sdk/index.html>

The screenshot shows the official Android Studio website. It features a large image of a laptop displaying the Android Studio interface, which includes a code editor, a navigation bar, and a preview window showing a mobile application. Below the image, the text 'Android Studio' and 'The Official IDE for Android' is displayed. A paragraph explains that Android Studio provides tools for building apps on every type of Android device. Another paragraph highlights world-class code editing, debugging, performance tooling, and a flexible build system. At the bottom left is a green button with the text 'DOWNLOAD ANDROID STUDIO 2.1 FOR WINDOWS (1181 MB)'.

Part 1: The 35 euro IoT project

Download Android Studio.

Terms and Conditions

This is the Android Software Development Kit License Agreement

1. Introduction

1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: <http://source.android.com/>, as updated from time to time.

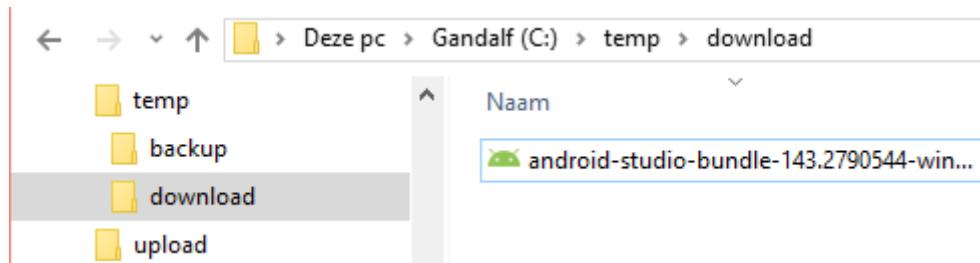
I have read and agree with the above terms and conditions

DOWNLOAD ANDROID STUDIO 2.1 FOR WINDOWS (1181 MB)

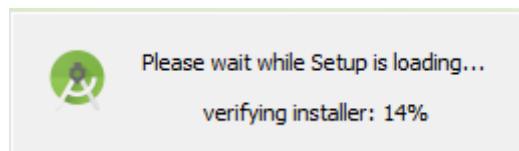
Vink "I have read and and agree..." aan en klik op Download Android Studio...

Het is een vrij groot bestand (> 1 GB) dus het downloaden kan even duren.

Dubbelklik met de muis op het gedownloade bestand om de installatie te starten.

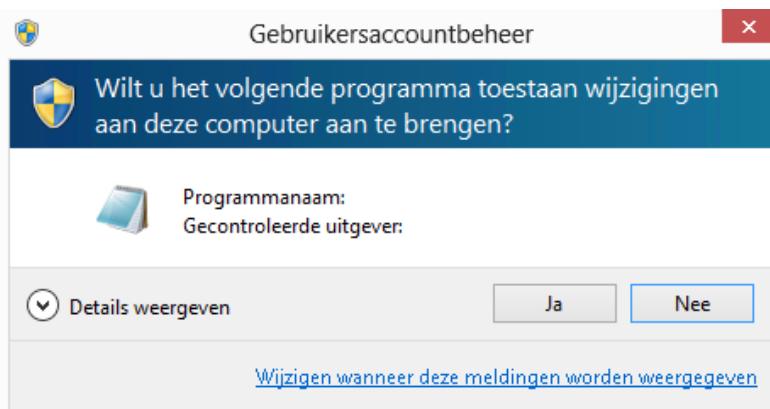


Even geduld...



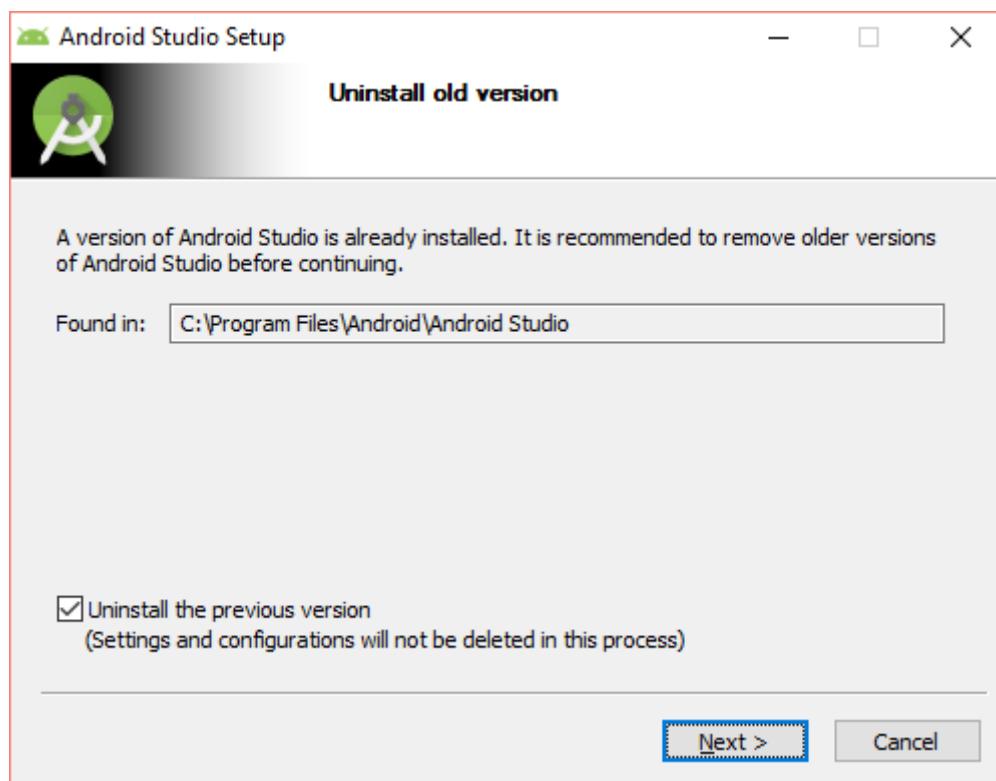
Part 1: The 35 euro IoT project

→ Ja



Optioneel:

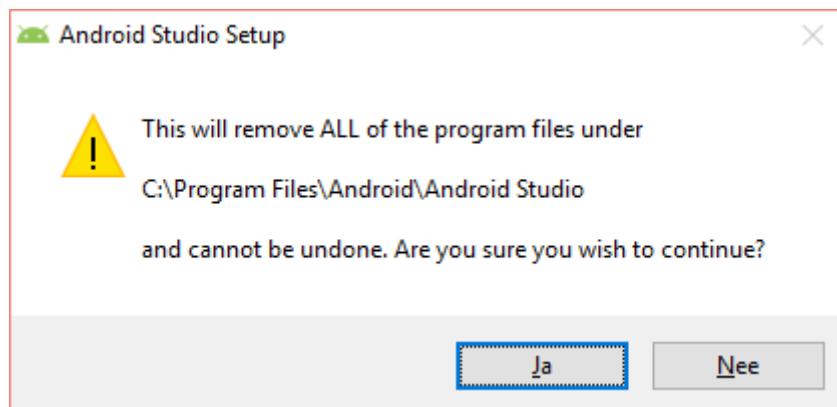
Als er al een oudere versie van Android Studio aanwezig is dan deze eerst verwijderen.



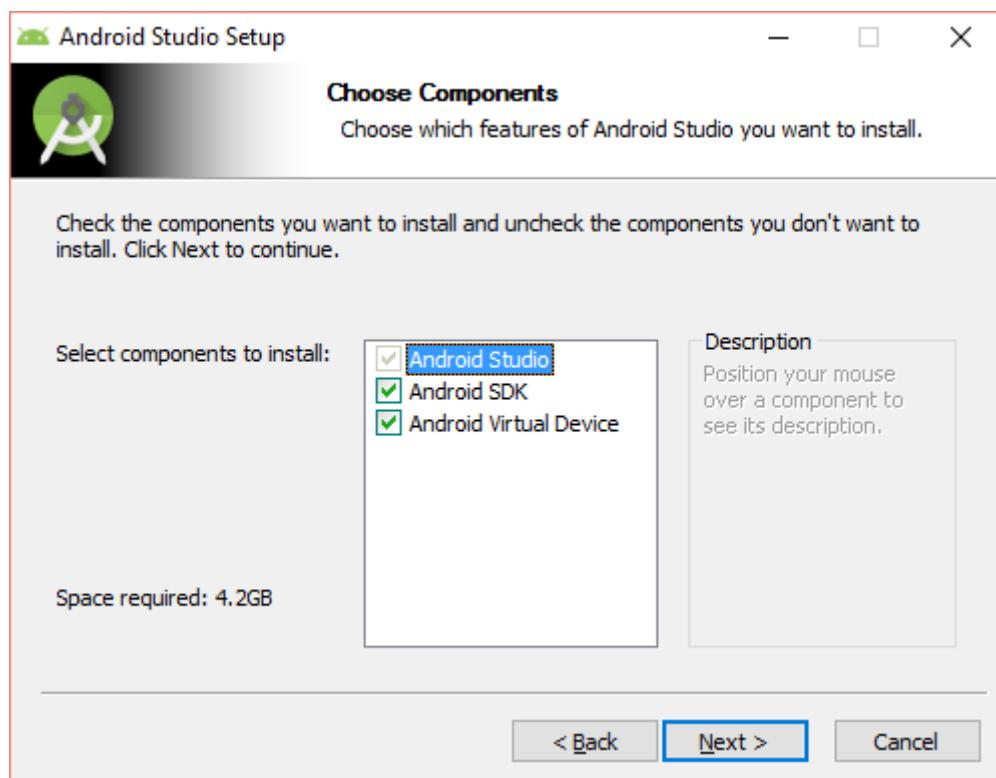
→ Next

Part 1: The 35 euro IoT project

→ Ja

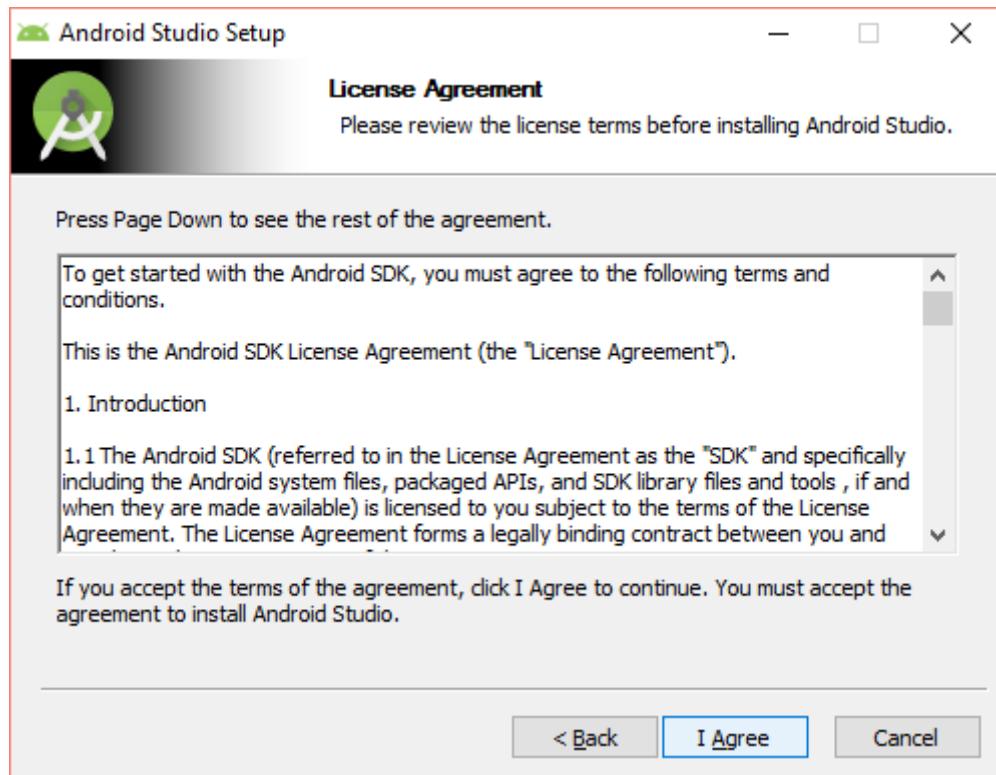


→ Next

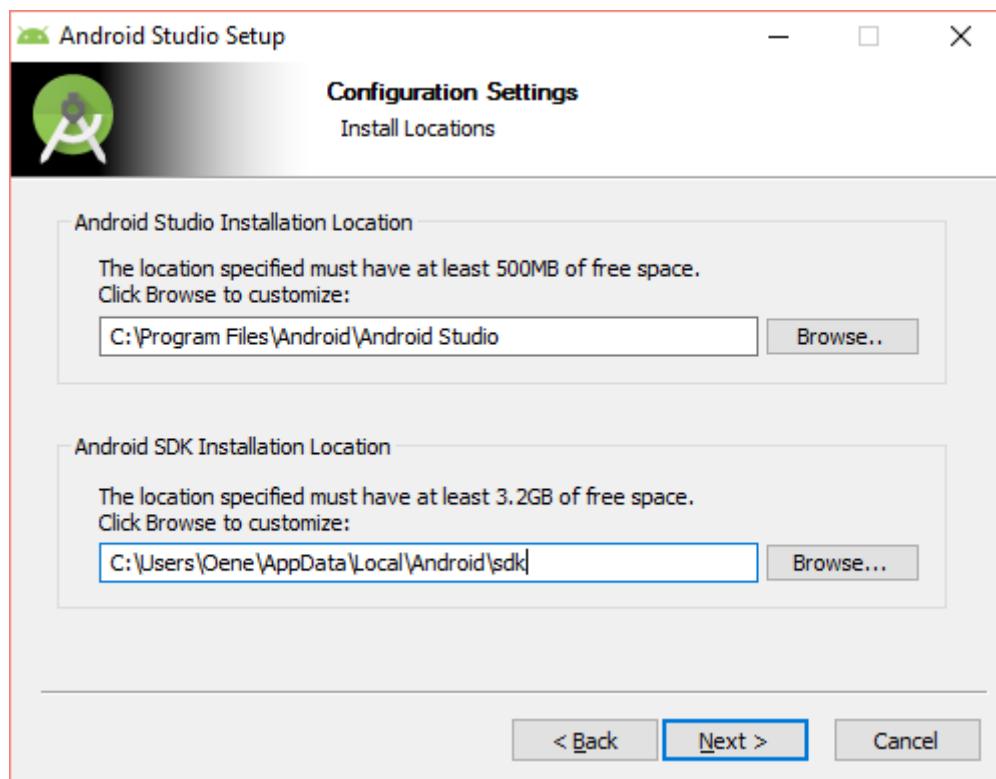


Part 1: The 35 euro IoT project

→ I Agree

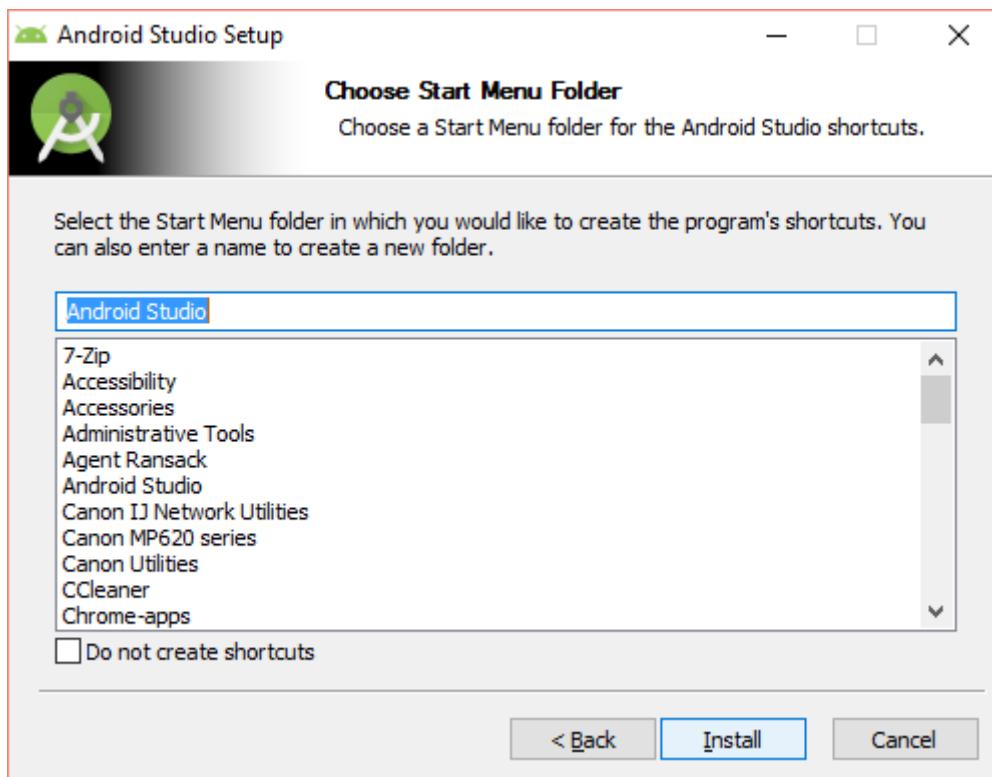


→ Next

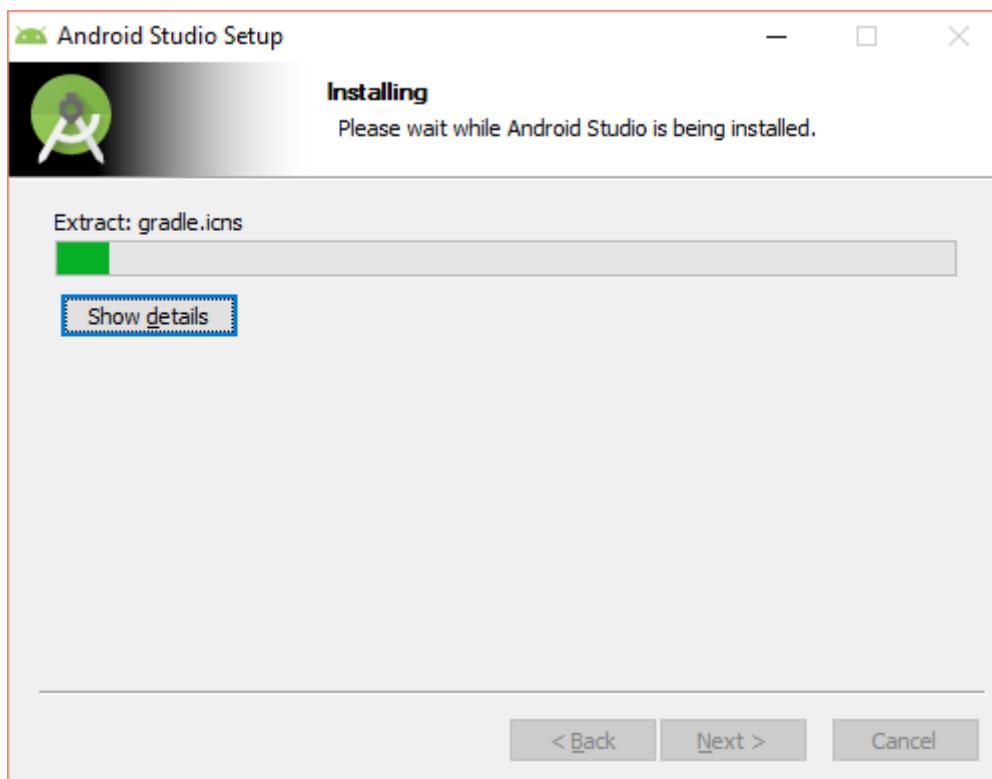


Part 1: The 35 euro IoT project

→ Install

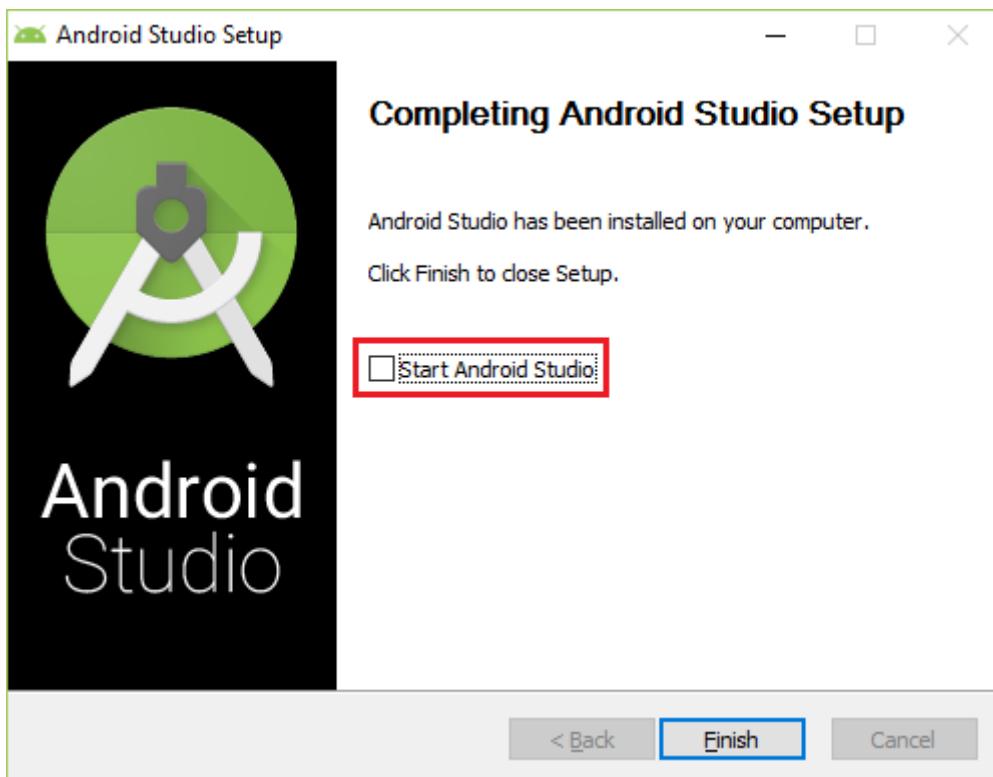


Even geduld...



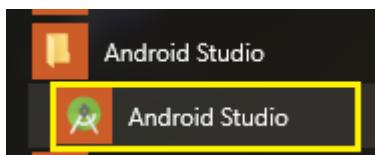
Part 1: The 35 euro IoT project

Vink Start Android Studio uit → Finish



4.3.9.4 Android Hello World App

Start Android Studio via Start → Android Studio → Android Studio

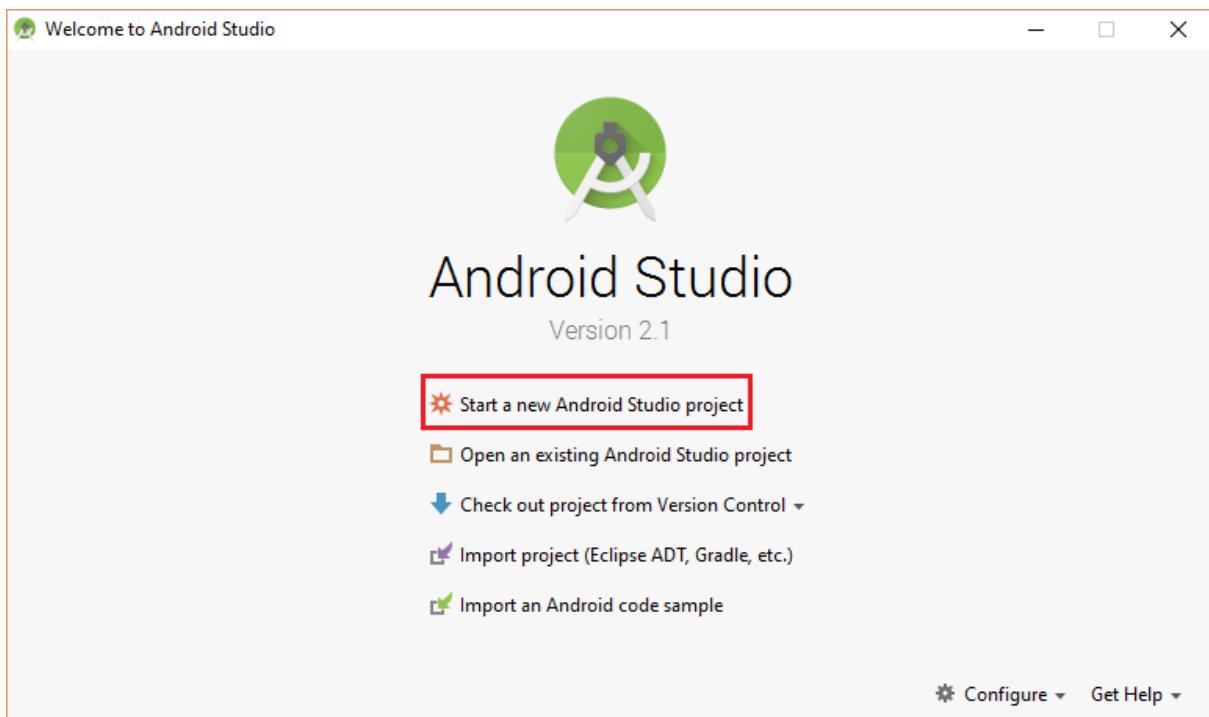


Even geduld...



Part 1: The 35 euro IoT project

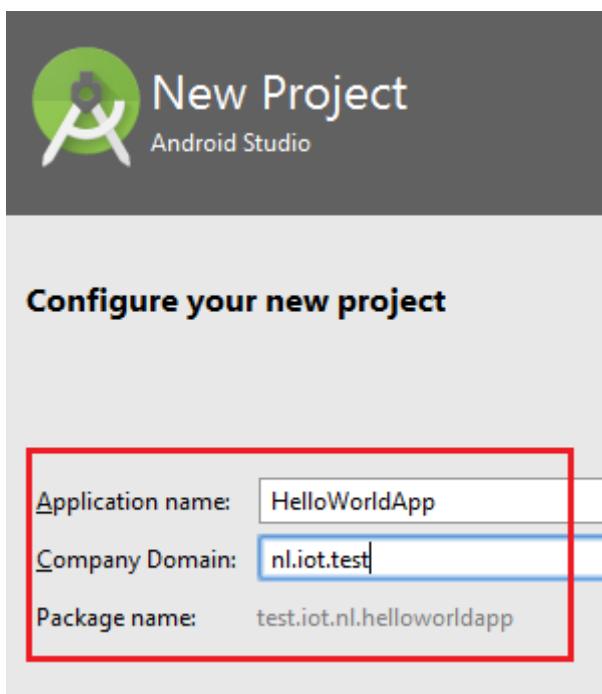
→ Start a new Android Studio project



Application name: HelloWorldApp

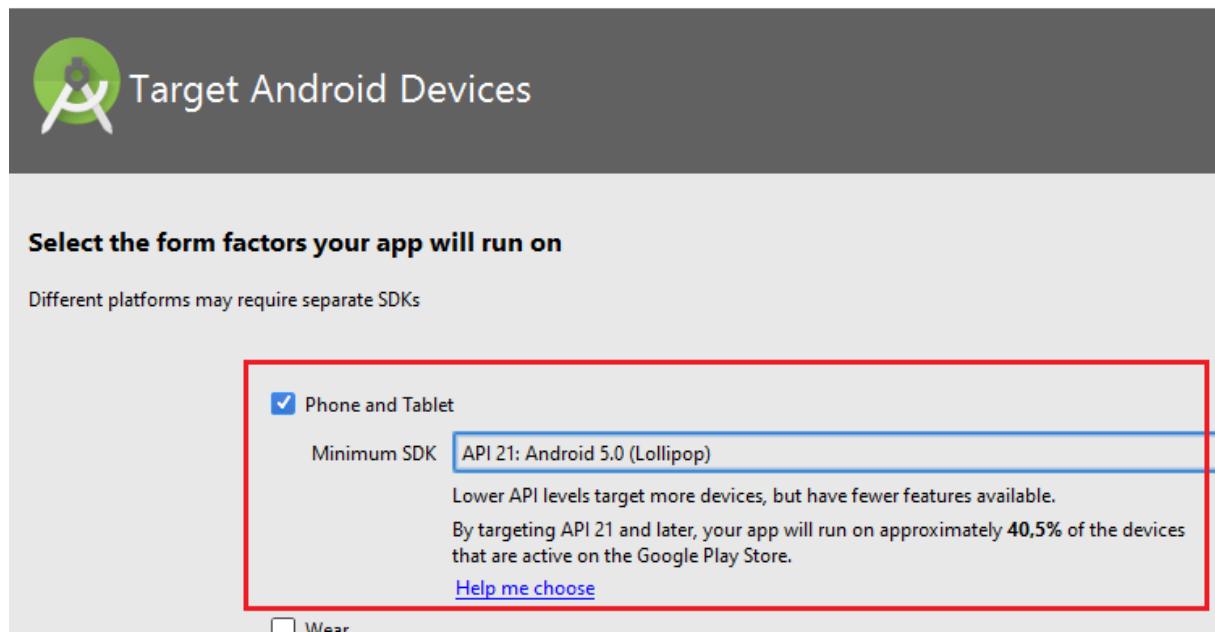
Company Domain: nl.iot.test

→ Next

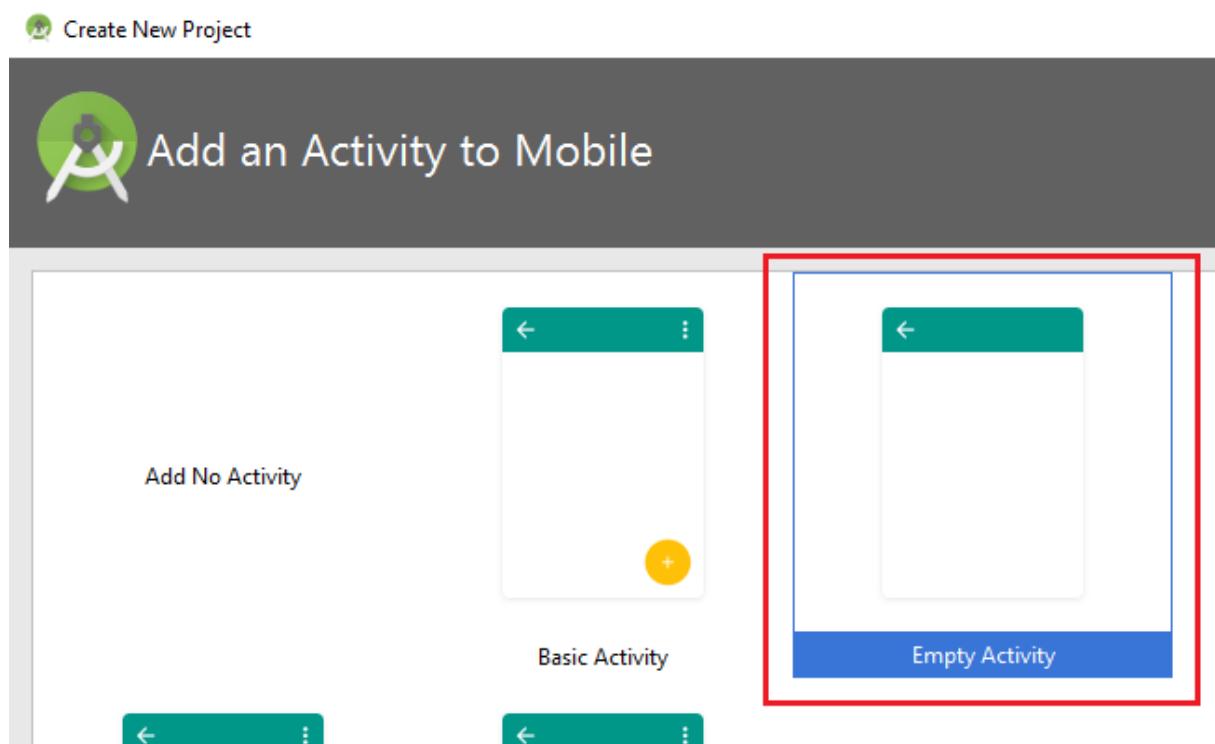


Part 1: The 35 euro IoT project

Selecteer een Minimum SDK, in dit voorbeeld API 21: Android 5.0 (Lollipop) voor telefoon en tablet → Next

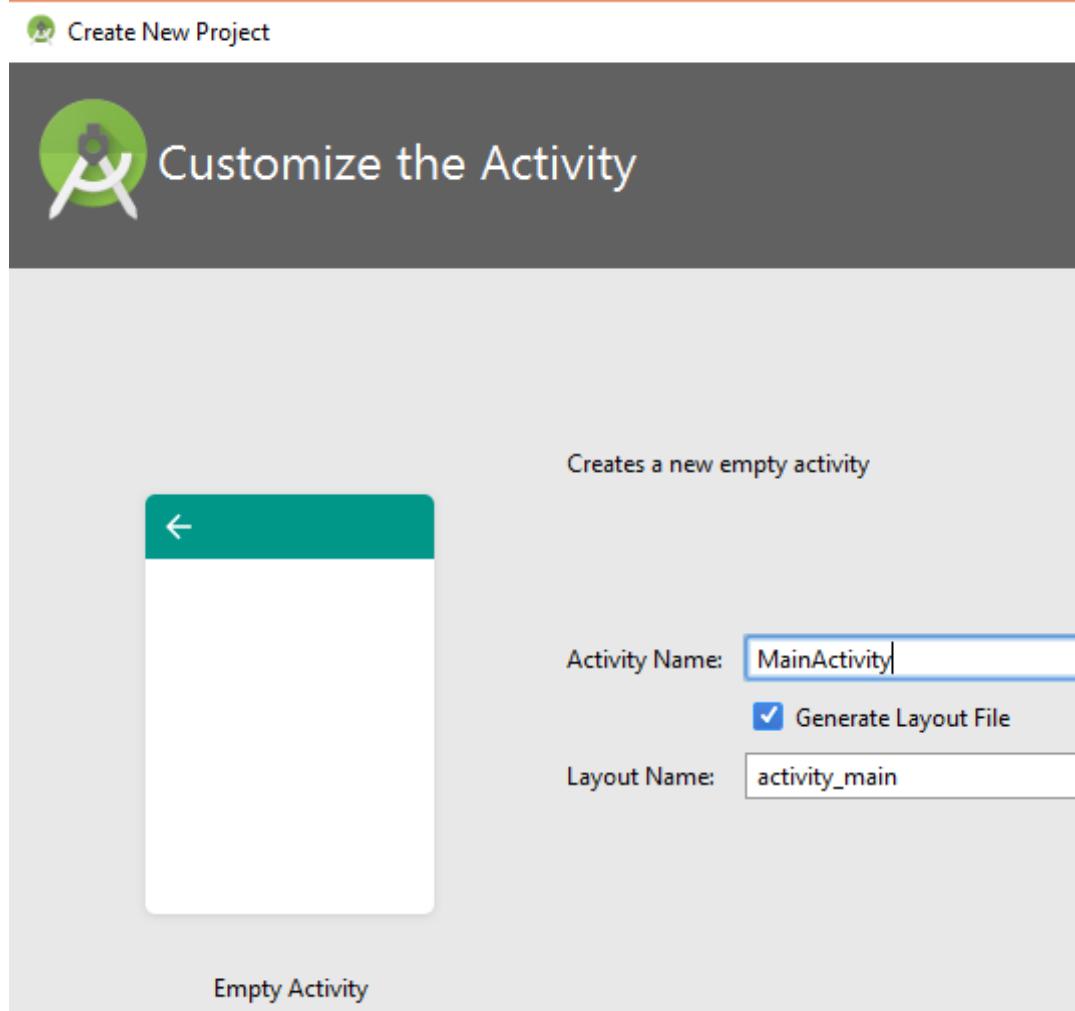


Empty Activity → Next

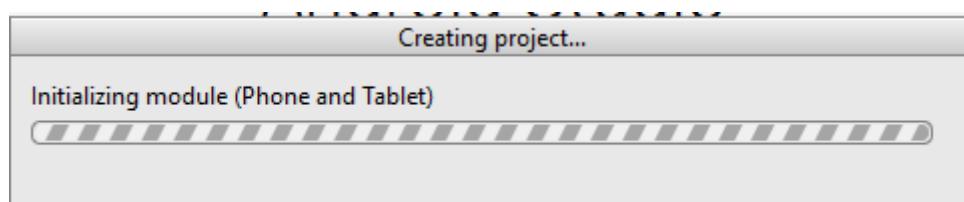


Part 1: The 35 euro IoT project

→ Finish

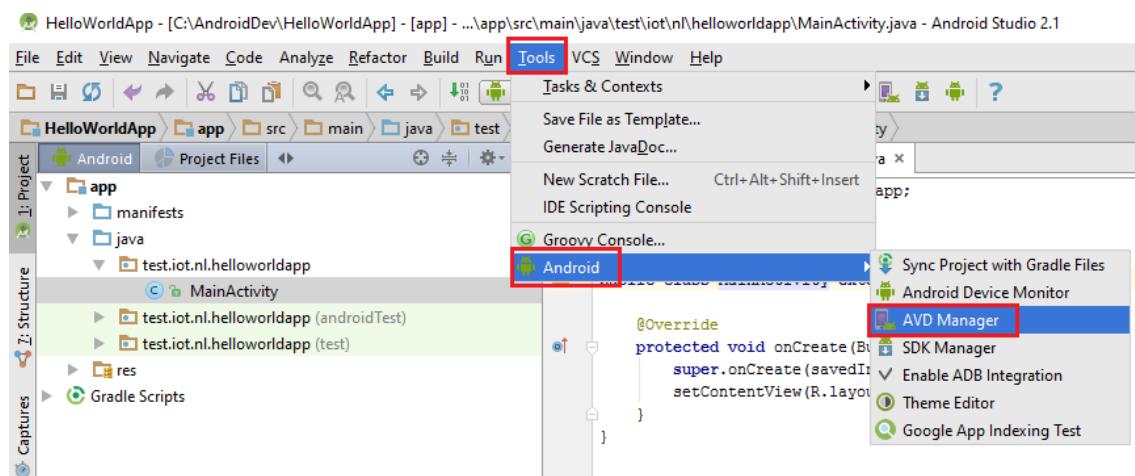


Even geduld...

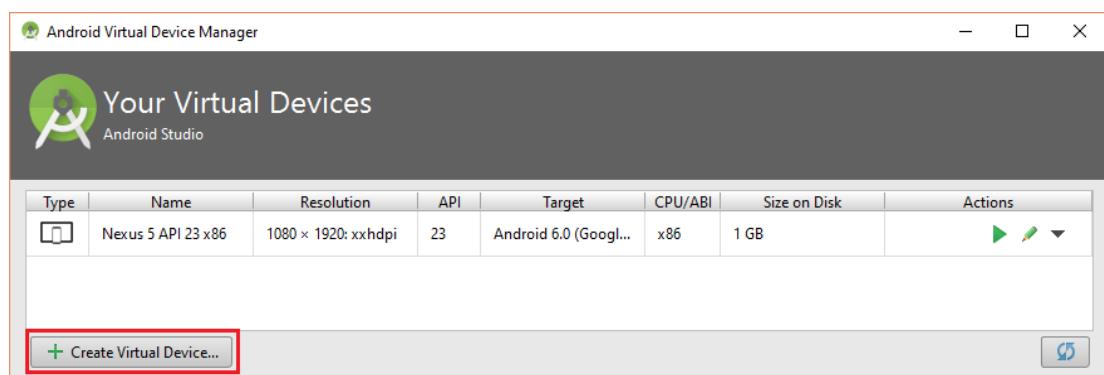


Part 1: The 35 euro IoT project

Menu → Tools → Android → AVD Manager

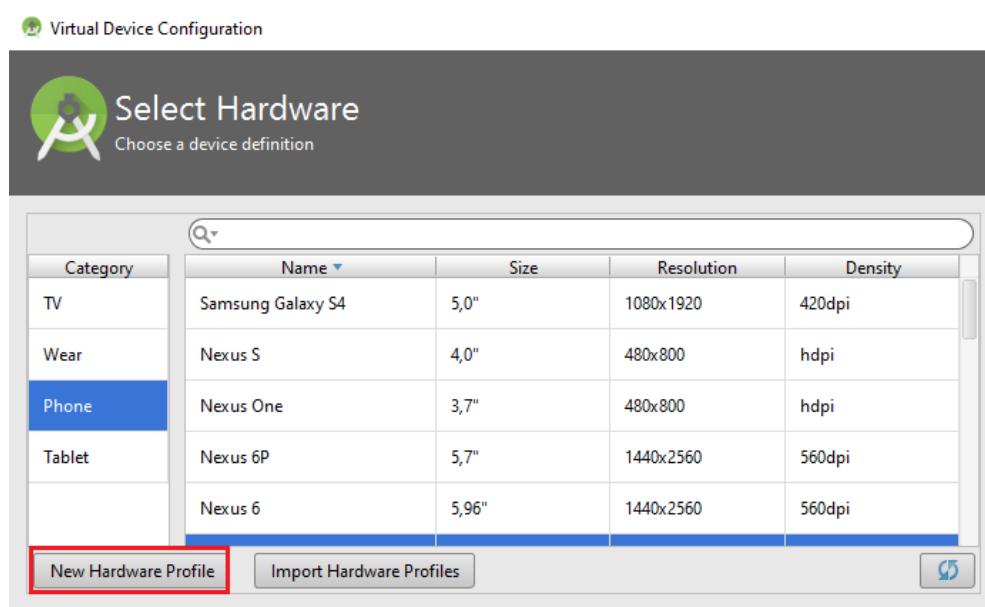


→ Create Virtual Device...



Maak een telefoon of tablet dat aan de geselecteerde API (in dit voorbeeld API 21) voldoet.

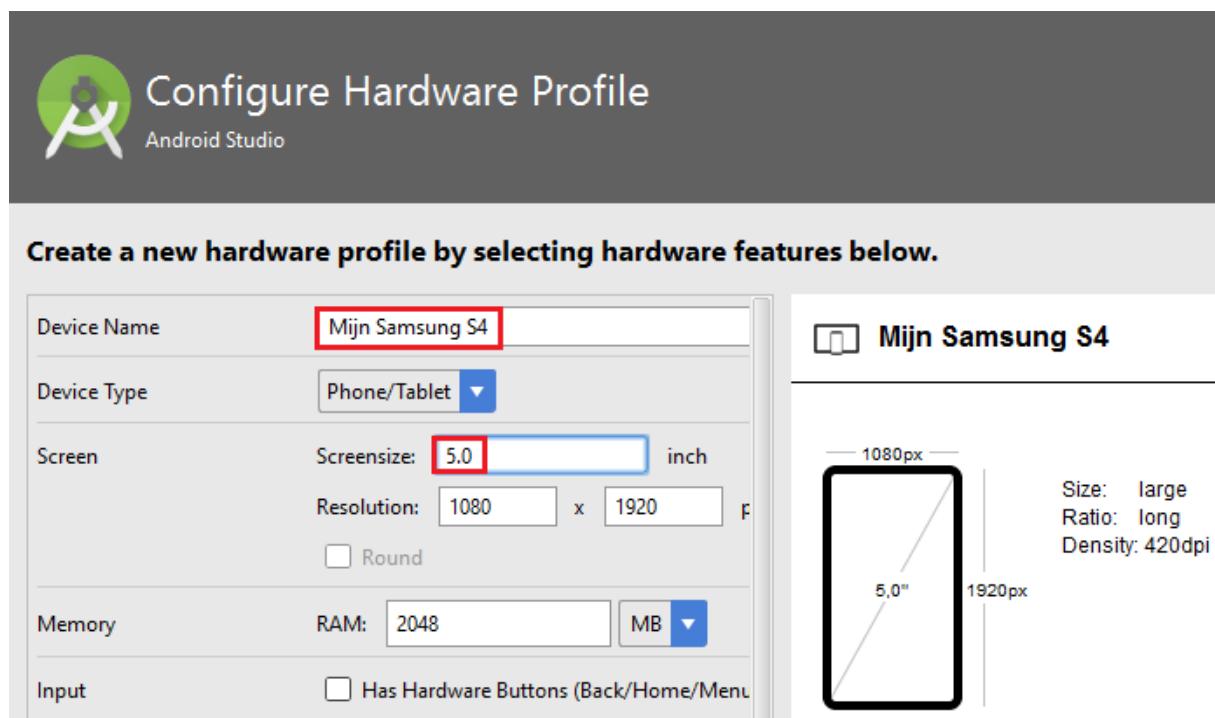
→ New Hardware Profile



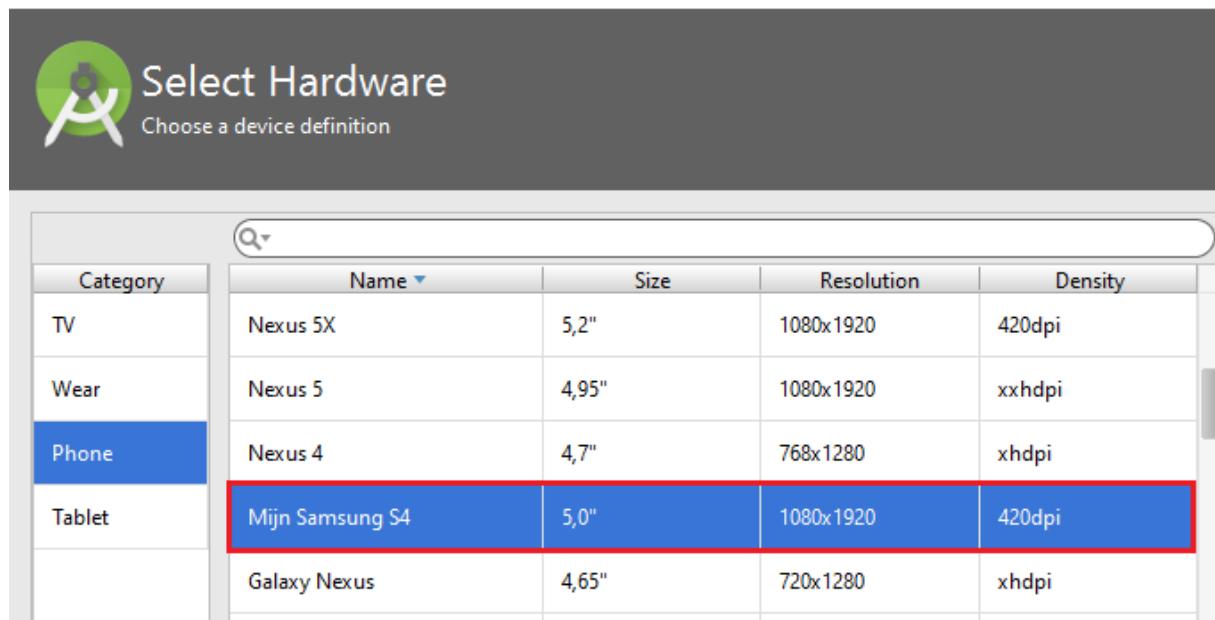
Part 1: The 35 euro IoT project

Device Name: Mijn Samsung S4

Screensize: 5.0 → Finish



→ Next

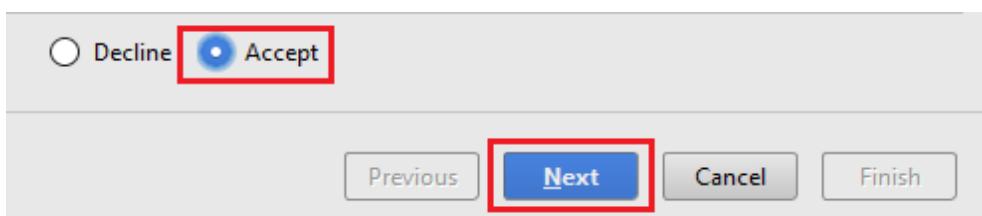


Part 1: The 35 euro IoT project

Selecteer Lollipop API Level 21 x86_64 → Download

| Release Name | API Level | ABI | Target |
|-----------------------------------|-----------|--------|--------------------------------|
| N | N | x86_64 | Android 6.X |
| N | N | x86 | Android 6.X |
| Marshmallow | 23 | x86_64 | Android 6.0 |
| Marshmallow | 23 | x86 | Android 6.0 |
| Lollipop Download | 22 | x86 | Android 5.1 |
| Lollipop Download | 22 | x86_64 | Android 5.1 |
| Lollipop Download | 21 | x86_64 | Android 5.0 (with Google APIs) |
| Lollipop Download | 21 | x86 | Android 5.0 (with Google APIs) |

Licence Agreement → Accept → Next



Even geduld... → Finish

Component Installer
Android Studio

Installing Requested Components

SDK Path: C:\Users\Oene\AppData\Local\Android\sdk

```
To install:  
- Google APIs Intel x86 Atom_64 System Image (system-images;android-21;google_apis;x86_64)  
  
Installing Google APIs Intel x86 Atom_64 System Image  
Downloading https://dl.google.com/android/repository/sys-img/google_apis/sysimg_x86_64-21_r11.zip
```

Part 1: The 35 euro IoT project

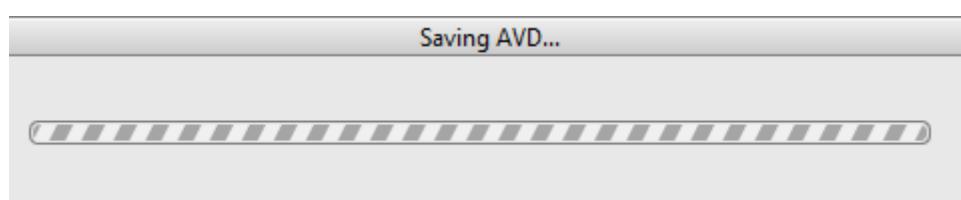
→ Next

| Release Name | API Level | ABI | Target |
|-------------------|-----------|--------|--------------------------------|
| N | N | x86_64 | Android 6.X |
| N | N | x86 | Android 6.X |
| Marshmallow | 23 | x86_64 | Android 6.0 |
| Marshmallow | 23 | x86 | Android 6.0 |
| Lollipop Download | 22 | x86 | Android 5.1 |
| Lollipop Download | 22 | x86_64 | Android 5.1 |
| Lollipop | 21 | x86_64 | Android 5.0 (with Google APIs) |
| Lollipop Download | 21 | x86 | Android 5.0 (with Google APIs) |

→ Finish

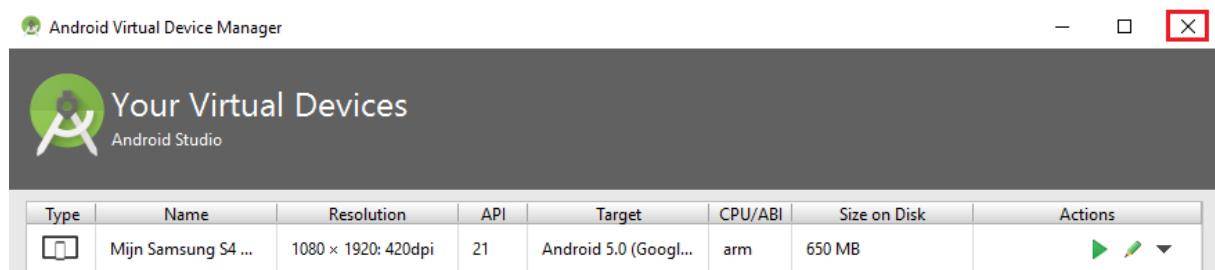


Even geduld...

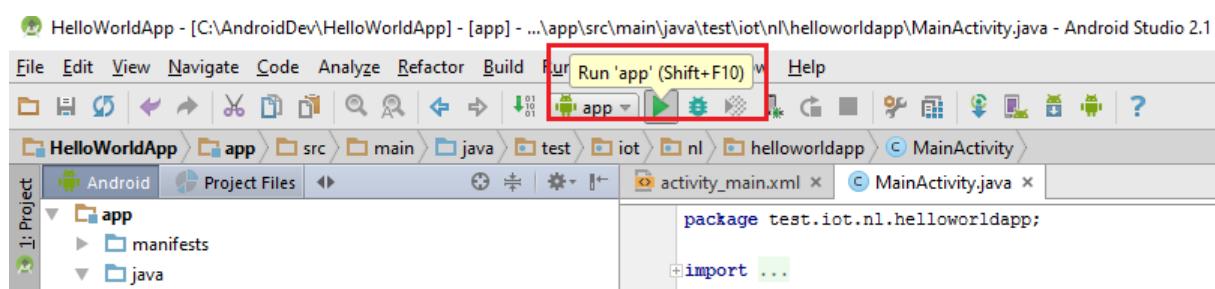


Part 1: The 35 euro IoT project

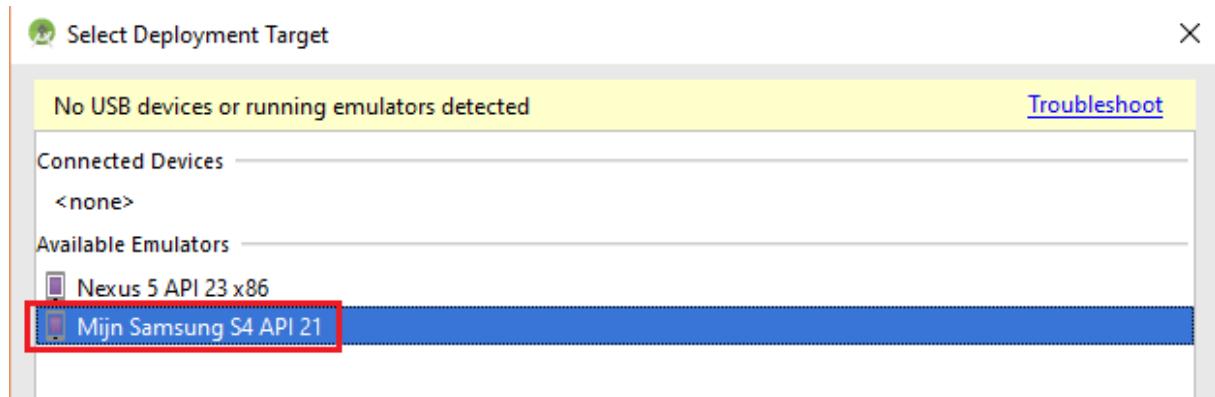
Sluit AVD Manager af met 



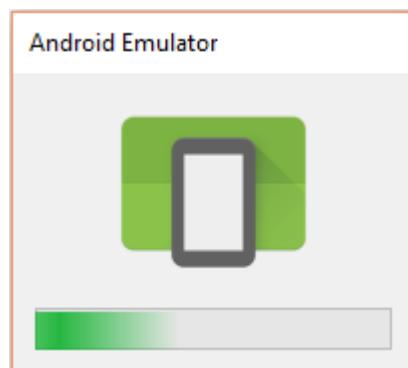
Voer de app uit met 



Selecteer het aangemaakte device (in dit voorbeeld Mijn Samsung S4 API 21) → OK



Even geduld...



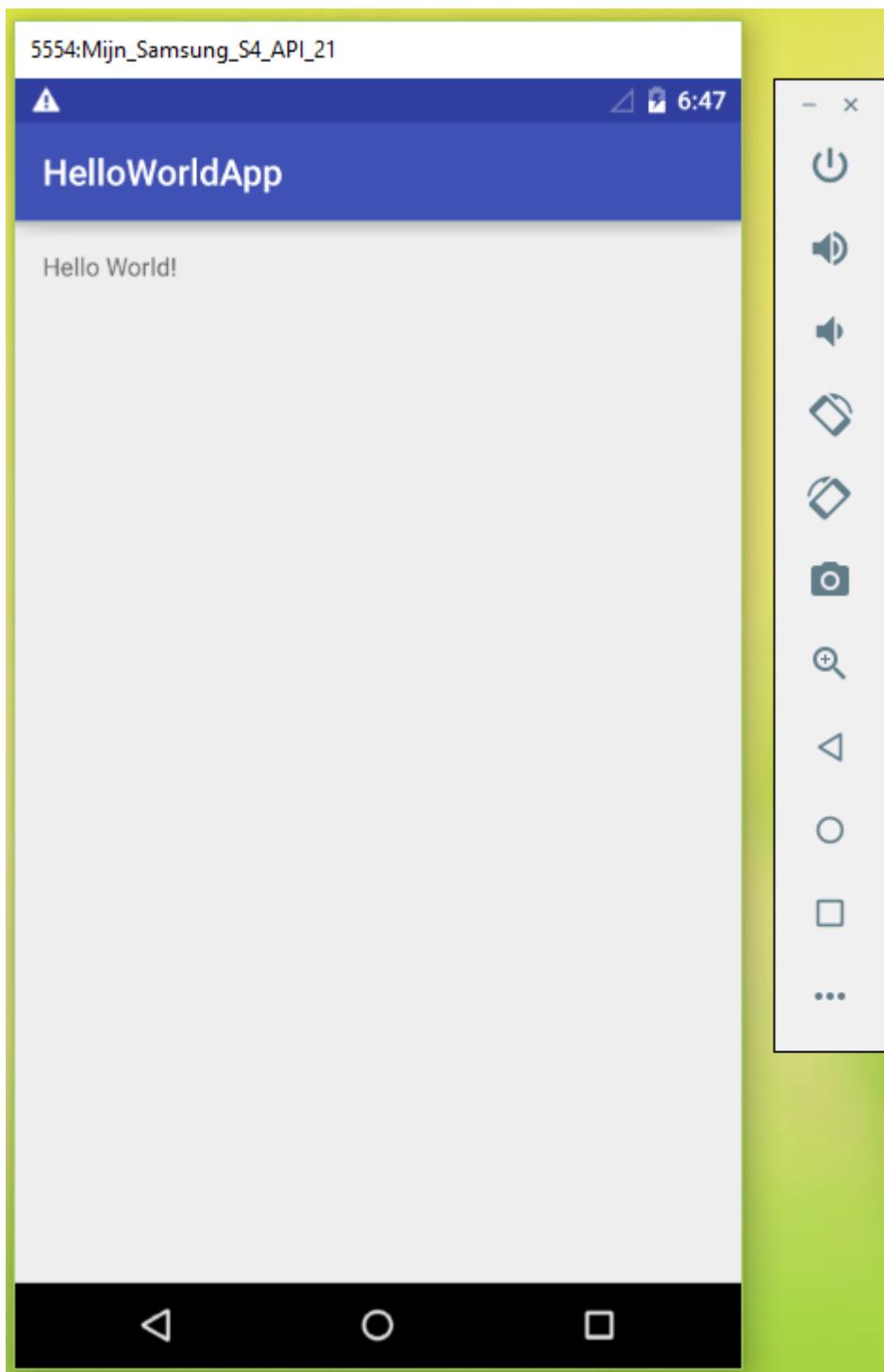
Part 1: The 35 euro IoT project

Even geduld...



Part 1: The 35 euro IoT project

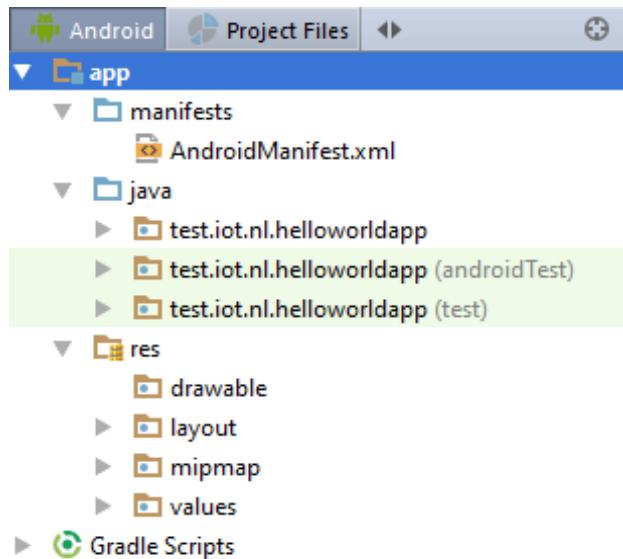
De app wordt gestart en Hello World wordt getoond.



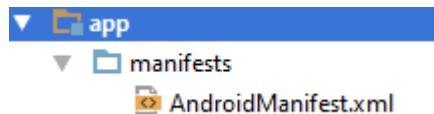
Part 1: The 35 euro IoT project

Een Android App heeft een standaard opmaak. De belangrijkste kenmerken komen hier aan de orde.

Structuur:

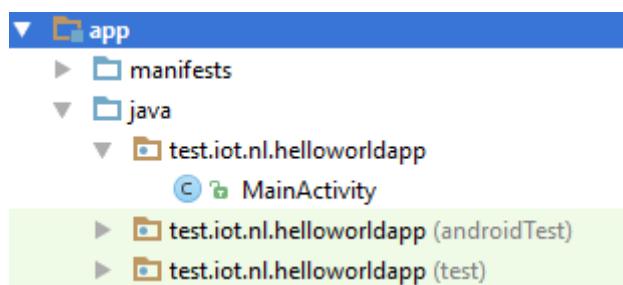


Onder de folder manifest staat AndroidManifest.xml.



Hierin wordt opgenomen waar het startpunt van de applicatie is maar ook welke autorisatie de app nodig heeft. Bijvoorbeeld dat de app toegang krijgt tot internet, de camera en gps.

Onder de folder Java worden de sources opgenomen.



Hierin staat o.a. de hoofdactiviteit (MainActivity.java) die gestart moet worden.

Part 1: The 35 euro IoT project

Zie: Sources → Android → HelloWorldApp → app → src → main → java → test → iot → nl → helloworldapp → MainActivity.java

MainActivity.java

```
package test.iot.nl.helloworldapp;

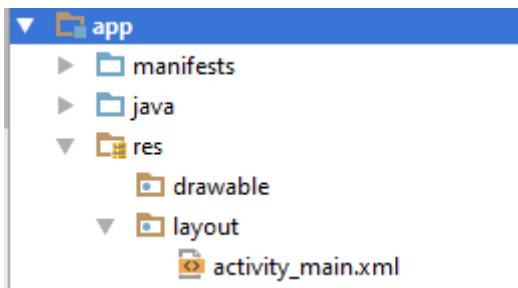
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

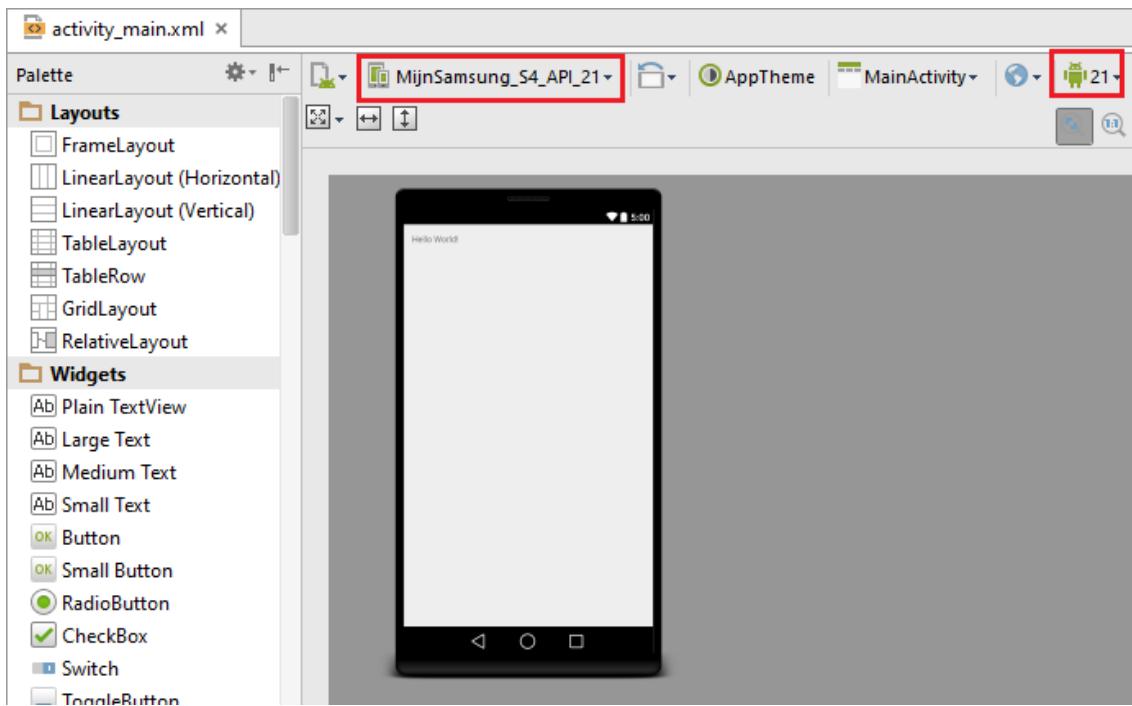
De hoofdactiviteit van onze HelloWorld app bestaat alleen uit het maken van de te tonen view. Dit gebeurt in de functie OnCreate.

Onder de folder res → layout staat het bestand activity_main.xml.



Hierin wordt de lay-out beschreven van het te tonen scherm op de mobiel. Via dropdowns kunnen de te gebruiken mobiel en Android API gezet worden (in ons voorbeeld API 21).

Part 1: The 35 euro IoT project



4.3.10 MongoDB installeren

4.3.10.1 Drivers

MongoDB kent verschillende drivers voor o.a. C++, Java en ook NodeJS. In dit boek gaan we de NodeJS driver gebruiken.

4.3.10.2 Installeren

De installatie bestaat uit twee delen:

- MongoDB (database),
- MongoDB NodeJS driver

4.3.10.3 MongoDB

Open een browser en ga naar:

<https://www.mongodb.com/download-center#community>

Selecteer Windows Server 2008 R2 and later... → DOWNLOAD (msi).

Part 1: The 35 euro IoT project

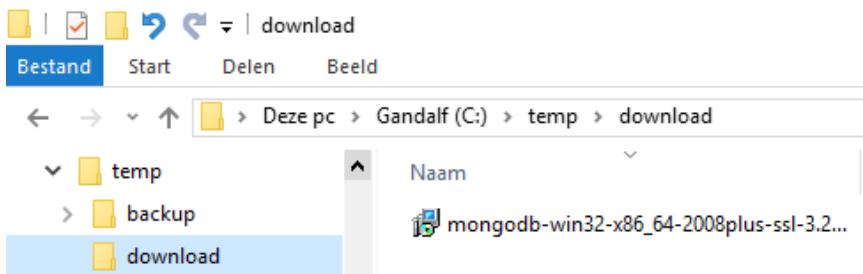
Current Stable Release (3.2.6)
04/27/2016: [Release Notes](#) | [Changelog](#)
Download Source: [tgz](#) | [zip](#)

Version:
Windows Server 2008 R2 and later, with SSL support ▾

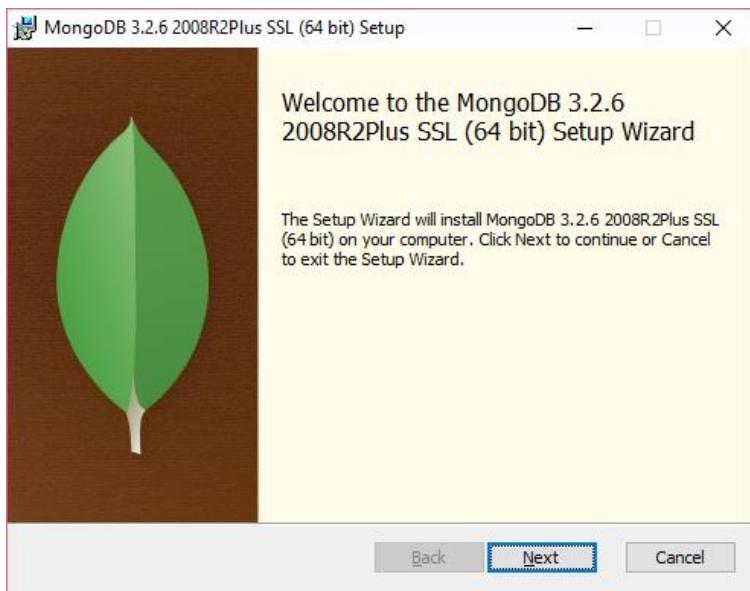
Installation Package:
[DOWNLOAD \(msi\)](#)

Deze versie bevat zowel de 32 als de 64 bits installatie.

Dubbelklik met de muis op `mongodb-win32-x86_64-2008plus-ssl-3.2.6-signed.msi`



→ Next

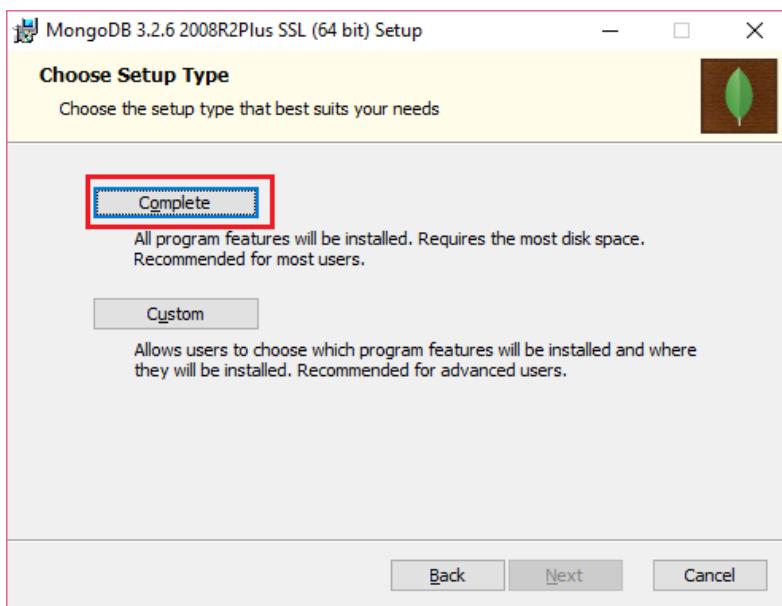


Vink I accept... aan → Next

Part 1: The 35 euro IoT project

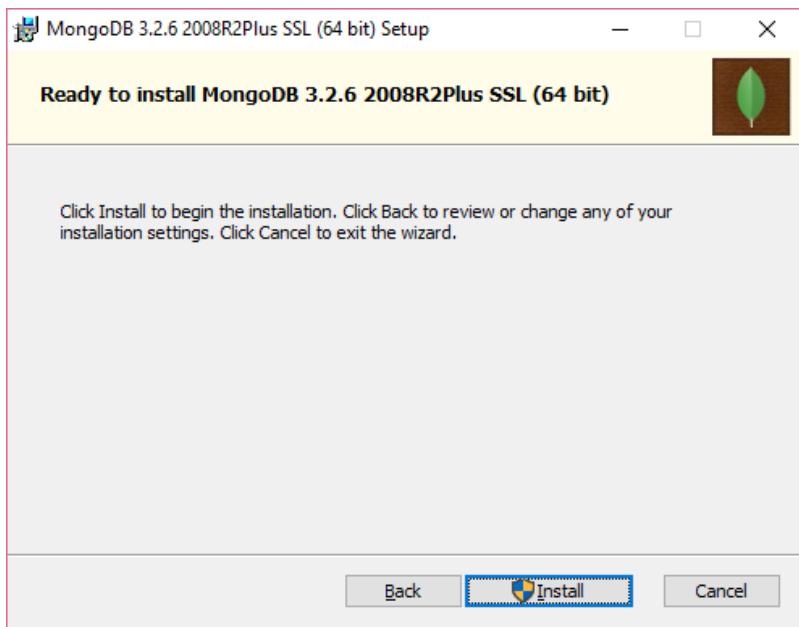


→ Complete



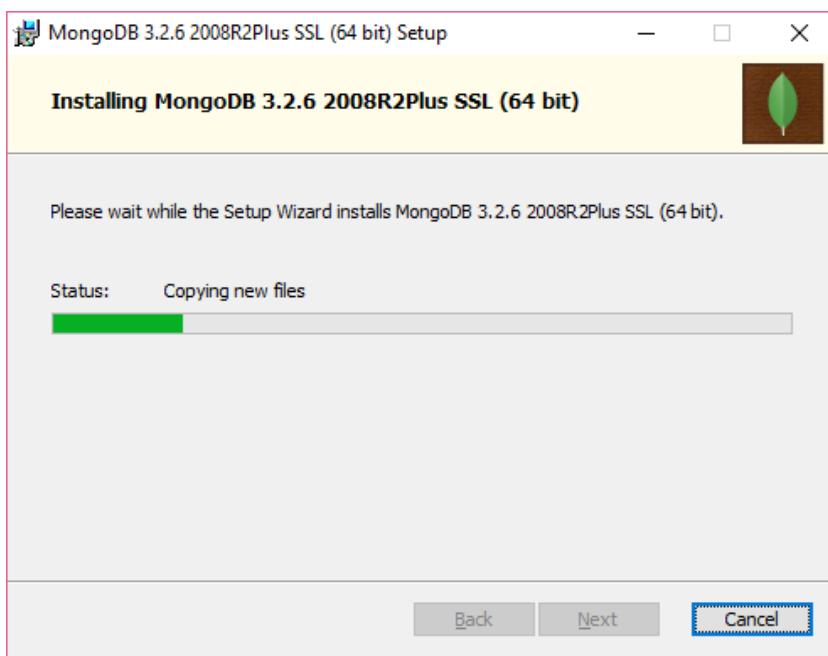
→ Install

Part 1: The 35 euro IoT project



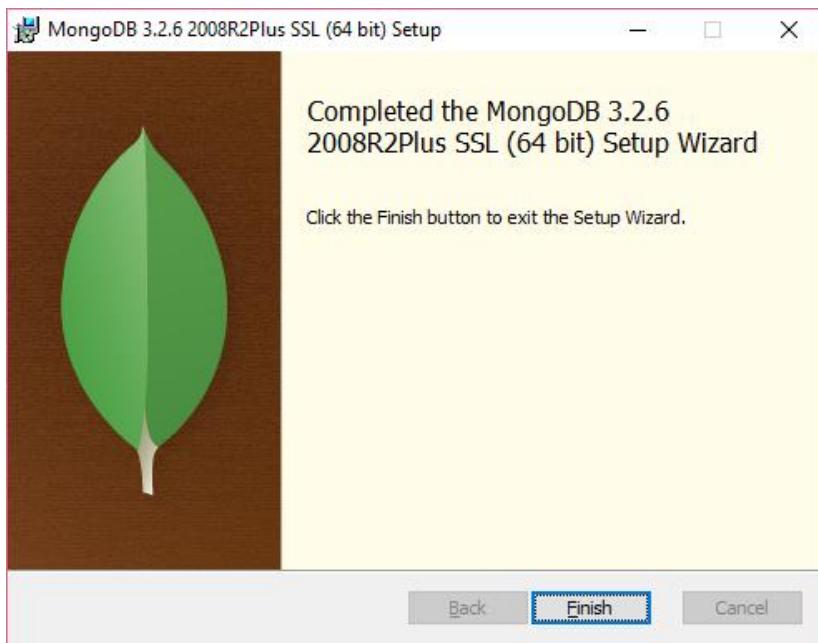
Gebruikers Accountbeheer → Ja

Even geduld...



→ Finish

Part 1: The 35 euro IoT project



4.3.10.4 MongoDB NodeJS driver

Installeren van de NodeJS MongoDB driver is simpel:

```
Open een command prompt  
C:  
cd\  
cd program files\nodejs  
npm install mongodb
```

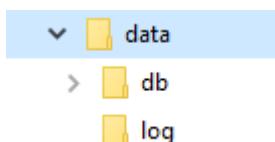
Administrator: Command

```
C:\Program Files\nodejs>npm install mongodb  
mongodb@2.1.20 node_modules\mongodb  
|-- es6-promise@3.0.2  
|-- readable-stream@1.0.31 (string_decoder@0.10.
```

4.3.10.5 MongoDB starten

Open een command prompt en maak eenmalig een directory aan voor de data.

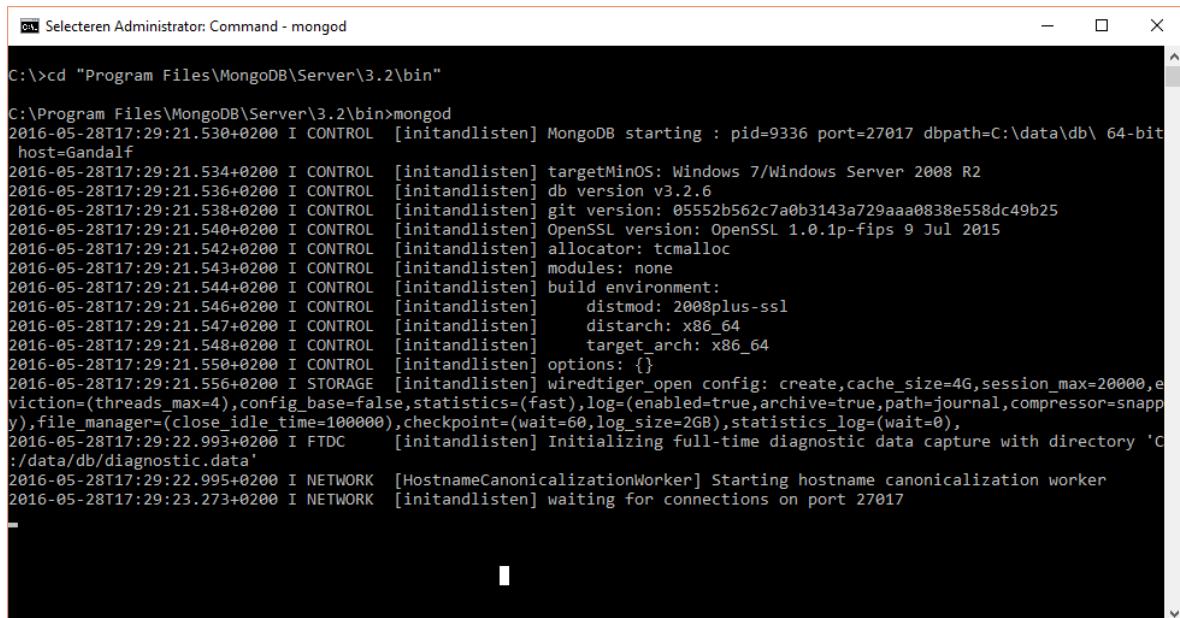
```
C:  
cd \  
md data\db  
cd data\db  
cd ..  
md log
```



Part 1: The 35 euro IoT project

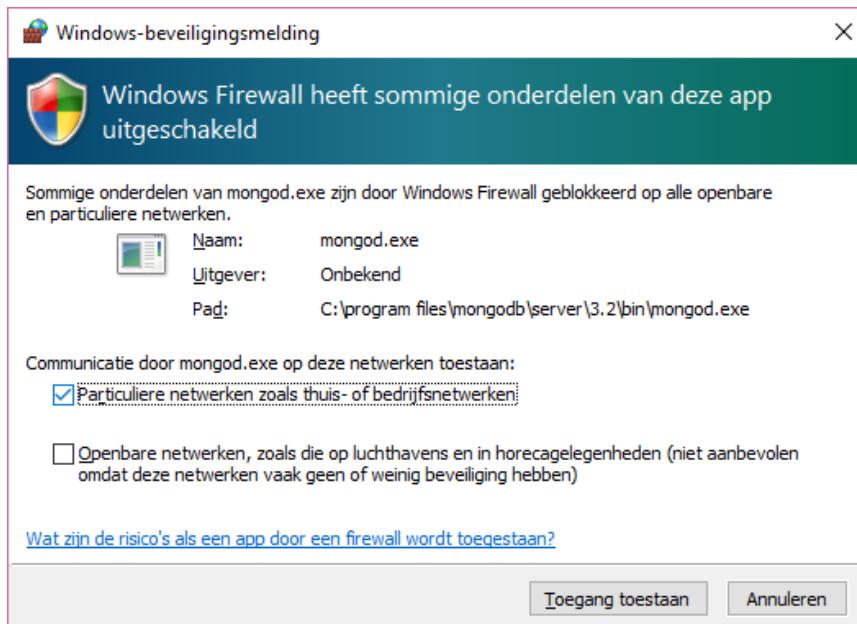
Start de MongoDB server met de commando's:

```
cd \
cd Program Files\MongoDB\Server\3.2\bin
mongod
```



```
C:\>cd "Program Files\MongoDB\Server\3.2\bin"
C:\Program Files\MongoDB\Server\3.2\bin>mongod
2016-05-28T17:29:21.530+0200 I CONTROL [initandlisten] MongoDB starting : pid=9336 port=27017 dbpath=C:\data\db\ 64-bit
host=Gandalf
2016-05-28T17:29:21.534+0200 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2016-05-28T17:29:21.536+0200 I CONTROL [initandlisten] db version v3.2.6
2016-05-28T17:29:21.538+0200 I CONTROL [initandlisten] git version: 05552b562c7a0b3143a729aaa0838e558dc49b25
2016-05-28T17:29:21.540+0200 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1p-fips 9 Jul 2015
2016-05-28T17:29:21.542+0200 I CONTROL [initandlisten] allocator: tcmalloc
2016-05-28T17:29:21.543+0200 I CONTROL [initandlisten] modules: none
2016-05-28T17:29:21.544+0200 I CONTROL [initandlisten] build environment:
2016-05-28T17:29:21.546+0200 I CONTROL [initandlisten]   distmod: 2008plus-ssl
2016-05-28T17:29:21.547+0200 I CONTROL [initandlisten]   distarch: x86_64
2016-05-28T17:29:21.548+0200 I CONTROL [initandlisten]   target_arch: x86_64
2016-05-28T17:29:21.550+0200 I CONTROL [initandlisten] options: {}
2016-05-28T17:29:21.556+0200 I STORAGE [initandlisten] wldredtiger_open config: create,cache_size=4G,session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2016-05-28T17:29:22.993+0200 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2016-05-28T17:29:22.995+0200 I NETWORK [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2016-05-28T17:29:23.273+0200 I NETWORK [initandlisten] waiting for connections on port 27017
```

Particuliere netwerken aanvinken indien nodig → Toegang toestaan



Part 1: The 35 euro IoT project

4.3.10.6 JSON importeren

In deze stap gaan we een voorbeeld JSON-bestand importeren in MongoDB. Hierbij worden zowel de database als ook het schema aangemaakt.

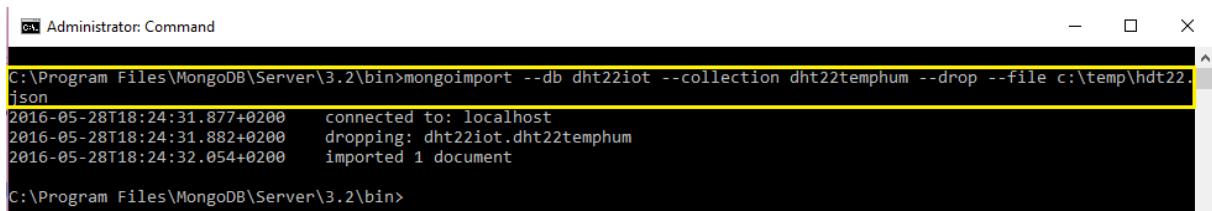
Voorbeeld JSON-bestand:

```
{  
    "dht22":  
    {  
        "datetime": "20160328180535",  
        "temperature": "20.4",  
        "humidity": "42"  
    }  
}
```

Plaats dit bestand als volgt op c:\temp onder de naam dht22.json.

Open een command prompt en voer de volgende commando's uit:

```
cd \  
cd Program Files\MongoDB\Server\3.2\bin  
mongoimport --db dht22iot --collection dht22temphum --drop --file  
c:\temp\dht22.json
```



```
C:\Program Files\MongoDB\Server\3.2\bin>mongoimport --db dht22iot --collection dht22temphum --drop --file c:\temp\dht22.json  
2016-05-28T18:24:31.877+0200      connected to: localhost  
2016-05-28T18:24:31.882+0200      dropping: dht22iot.dht22temphum  
2016-05-28T18:24:32.054+0200      imported 1 document  
C:\Program Files\MongoDB\Server\3.2\bin>
```

Controleer of data is opgeslagen door het openen een command prompt en het uitvoeren van de volgende commando's:

```
cd \  
cd Program Files\MongoDB\Server\3.2\bin  
mongo  
use dht22iot  
db.dht22temphum.find()
```



```
C:\Program Files\MongoDB\Server\3.2\bin>mongo  
MongoDB shell version: 3.2.6  
connecting to: test  
> use dht22iot  
switched to db dht22iot  
> db.dht22temphum.find()  
{ "_id" : ObjectId("5749c63fe14650267a2e69b2"), "hdt22" : { "datetime" : "20160328180535", "temperature" : "20.4", "humidity" : "42" } }  
> -
```

MongoDB heeft het schema uitgebreid met een uniek id (`_id`).

Part 1: The 35 euro IoT project

4.3.10.7 NodeJS script

Maak een test script aan op bijvoorbeeld c:\temp en kopieer deze naar c:\Program Files\nodejs (rechtstreek aanmaken van bestanden op c:\Program Files is onder Windows 10 niet zomaar mogelijk).

testdb.js

```
// 1
var MongoClient = require('mongodb').MongoClient;

// 2
var url = 'mongodb://localhost:27017/dht22iot';

// 3
var insertDocument = function(db, callback) {
    db.collection('dht22temphum').insertOne(
    {
        "dht22": {
            "datetime": "20160328190535",
            "temperature": "20.4",
            "humidity": "42"
        }
    }, function(err, result) {
        console.log("Inserted a document into the restaurants collection.");
        callback();
    });
};

// 4
MongoClient.connect(url, function(err, db) {
    insertDocument(db, function() {
        db.close();
    });
});
```

Zie: Sources → NodeJS → testdb.js

Het script bestaat uit de volgende delen:

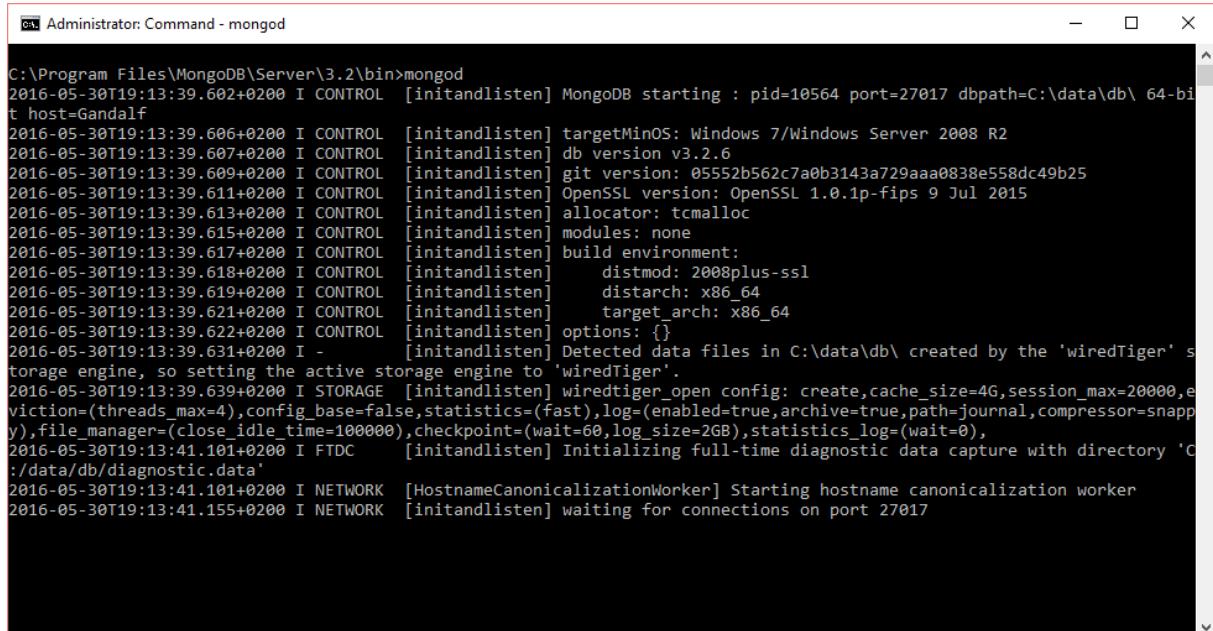
1. Er wordt een MongoClient aangemaakt.
2. Er wordt een URL naar de database gedefinieerd.
3. Er worden een functie insertDocument gemaakt waarin een enkele rij wordt geschreven in de database.
4. Zodra er een connectie met de database wordt gemaakt wordt de rij weggeschreven en de connectie met de database wordt weer gesloten.

Part 1: The 35 euro IoT project

Open drie command prompts.

Command prompt 1:

```
C:  
cd\  
cd Program Files\MongoDB\Server\3.2\bin  
mongod
```



```
C:\Program Files\MongoDB\Server\3.2\bin>mongod  
2016-05-30T19:13:39.602+0200 I CONTROL [initandlisten] MongoDB starting : pid=10564 port=27017 dbpath=C:\data\db\ 64-bit host=Gandalf  
2016-05-30T19:13:39.607+0200 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2  
2016-05-30T19:13:39.609+0200 I CONTROL [initandlisten] db version v3.2.6  
2016-05-30T19:13:39.611+0200 I CONTROL [initandlisten] git version: 05552b562c7a0b3143a729aaa0838e558dc49b25  
2016-05-30T19:13:39.611+0200 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1p-fips 9 Jul 2015  
2016-05-30T19:13:39.613+0200 I CONTROL [initandlisten] allocator: tcmalloc  
2016-05-30T19:13:39.615+0200 I CONTROL [initandlisten] modules: none  
2016-05-30T19:13:39.617+0200 I CONTROL [initandlisten] build environment:  
2016-05-30T19:13:39.618+0200 I CONTROL [initandlisten] distmod: 2008plus-ssl  
2016-05-30T19:13:39.619+0200 I CONTROL [initandlisten] distarch: x86_64  
2016-05-30T19:13:39.621+0200 I CONTROL [initandlisten] target_arch: x86_64  
2016-05-30T19:13:39.622+0200 I CONTROL [initandlisten] options: {}  
2016-05-30T19:13:39.631+0200 I - [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.  
2016-05-30T19:13:39.639+0200 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=4G/session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),  
2016-05-30T19:13:41.101+0200 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'  
2016-05-30T19:13:41.101+0200 I NETWORK [HostnameCanonicalizationWorker] Starting hostname canonicalization worker  
2016-05-30T19:13:41.155+0200 I NETWORK [initandlisten] Waiting for connections on port 27017
```

Command prompt 2:

```
C:  
cd\  
cd Program Files\MongoDB\Server\3.2\bin  
mongo  
use dht22iot  
db.dht22tempum.find()
```



```
C:\Program Files\MongoDB\Server\3.2\bin>mongo  
MongoDB shell version: 3.2.6  
connecting to: test  
> use dht22iot  
switched to db dht22iot  
> db.dht22tempum.find()  
{ "_id" : ObjectId("5749c63fe14650267a2e69b2"), "hdt22" : { "datetime" : "20160328180535", "temperature" : "20.4", "humidity" : "42" } }  
>
```

Part 1: The 35 euro IoT project

Command prompt 3:

```
C:  
cd\  
cd Program Files\nodejs  
node testdb.js
```

```
Administrator: Command  
C:\Program Files\nodejs>node testdb.js  
Inserted a document into the restaurants collection.  
C:\Program Files\nodejs>
```

Command prompt 2:

```
C:  
cd\  
cd Program Files\MongoDB\Server\3.2\bin  
mongo  
use dht22iot  
db.dht22tempum.find()
```

```
Administrator: Command - mongo  
C:\Program Files\MongoDB\Server\3.2\bin>mongo  
MongoDB shell version: 3.2.6  
connecting to: test  
> use dht22iot  
switched to db dht22iot  
> db.dht22tempum.find()  
{ "_id" : ObjectId("5749c63fe14650267a2e69b2"), "hdt22" : { "datetime" : "20160328180535", "temperature" : "20.4", "humidity" : "42" } }  
{ "_id" : ObjectId("5749d40503e124682b8699be"), "hdt22" : { "datetime" : "20160328190535", "temperature" : "20.4", "humidity" : "42" } }  
> -
```

Er is een tweede rij aangemaakt.

Om met een schone database te beginnen worden alle rijen verwijderd.

Command prompt 2:

```
C:  
cd\  
cd Program Files\MongoDB\Server\3.2\bin  
mongo  
use dht22iot  
db.dht22tempum.remove({})  
db.dht22tempum.find()
```

```
Administrator: Command - mongo  
C:\Program Files\MongoDB\Server\3.2\bin>mongo  
MongoDB shell version: 3.2.6  
connecting to: test  
> use dht22iot  
switched to db dht22iot  
> db.dht22tempum.remove({})  
writeResult({ "nRemoved" : 2 })  
> db.dht22tempum.find()  
>
```

Sluit de command prompts 1 en 2 af met Ctrl+C.

Part 1: The 35 euro IoT project

4.3.11 Virtual Router

4.3.11.1 Geen Wi-Fi en nu?

Voor het geval er geen Wi-Fi beschikbaar is kan er gebruik worden gemaakt van een virtuele Wi-Fi router. Voorwaarde is dat de Windows PC, laptop of tablet beschikt over een Wi-Fi adapter.

4.3.11.2 Installatie

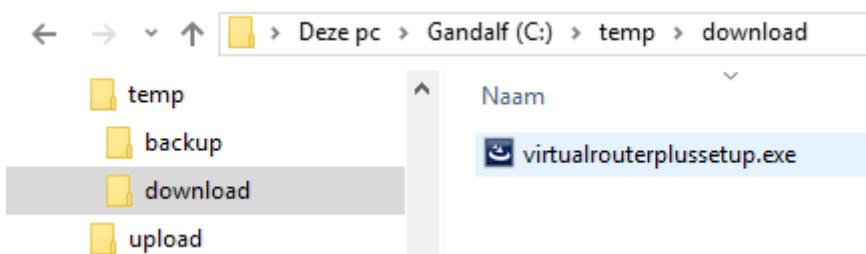
Open een browser en ga naar:

<http://virtualrouter-plus.nl.softonic.com/>

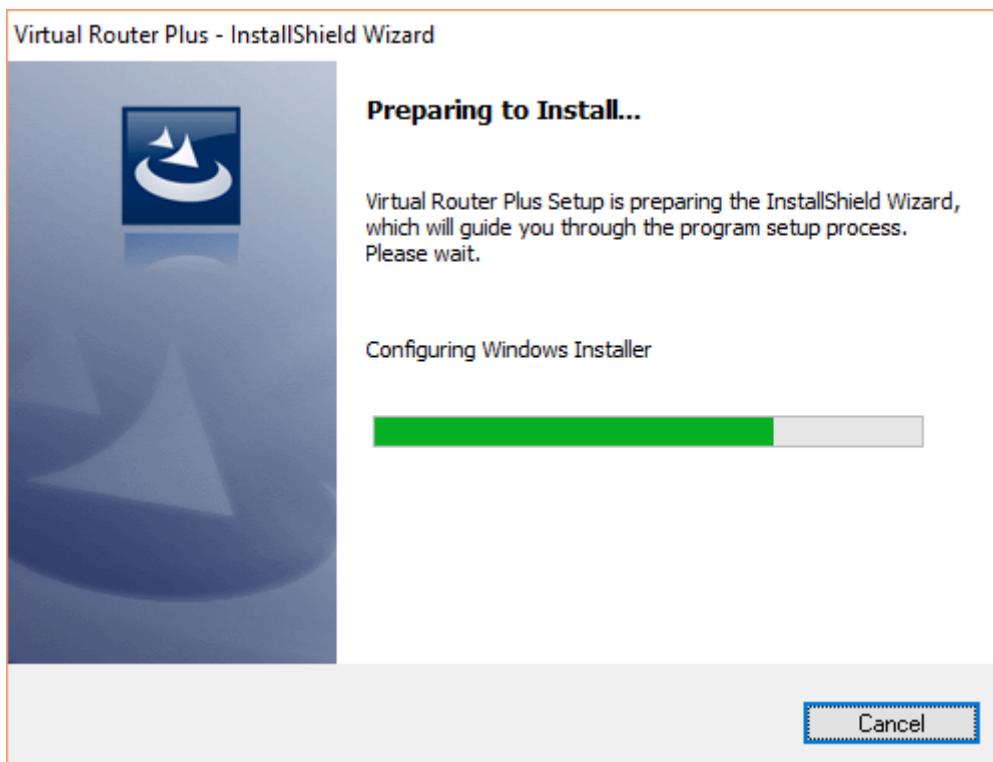
The screenshot shows the Softonic website interface. At the top, there's a navigation bar with the Softonic logo, followed by links for APPLICATIES, GAMES, EDITORIAL, ANSWERS, and VIDEO'S. Below the navigation bar, there's a search bar labeled 'Zoeken...'. A sidebar on the left contains the text 'EXPERT ADVICE & SAFE DOWNLOADS.' and a list of three steps: '1. Klik op Gratis download', '2. Start de snelle scan', and '3. Repareer de fouten'. To the right of this list is a large blue button labeled 'Gratis Download' with a downward arrow icon. The main content area features a large image of a router and the text 'VirtualRouter Plus'. Below this, there's a green button labeled 'Gratis Download' with a white download icon, and the text 'Veilige Download' underneath it. Further down, there's another green button labeled 'Alternatieve download' with a small download icon. At the bottom of the page, there's a green box containing the text 'HET PROGRAMMA WORDT NU OPGEHAALD.' and 'Als de download niet automatisch start, klik dan hier'.

Part 1: The 35 euro IoT project

Dubbelklik met de muis op het gedownloade bestand om de installatie te starten.

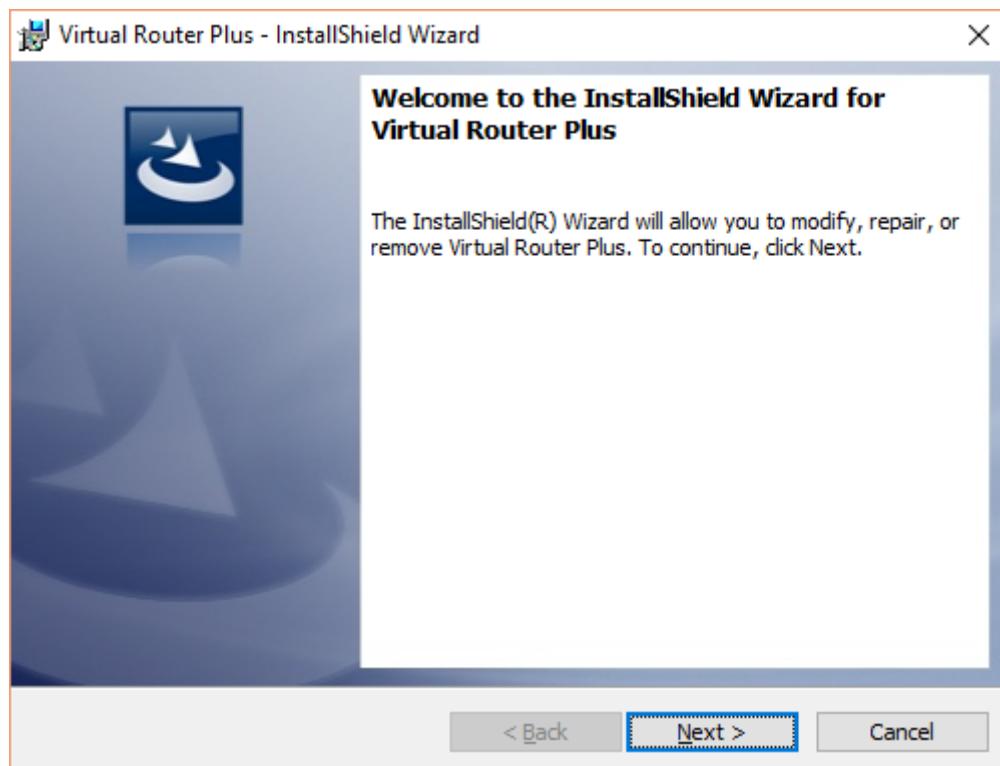


Even geduld...

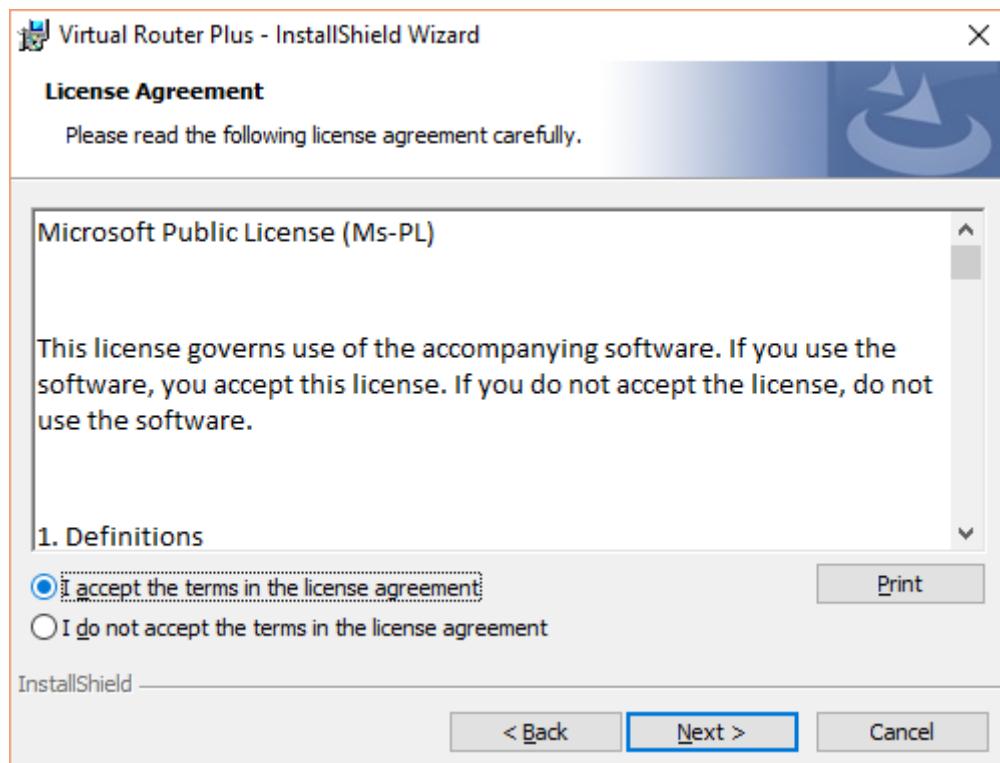


Part 1: The 35 euro IoT project

→ Next

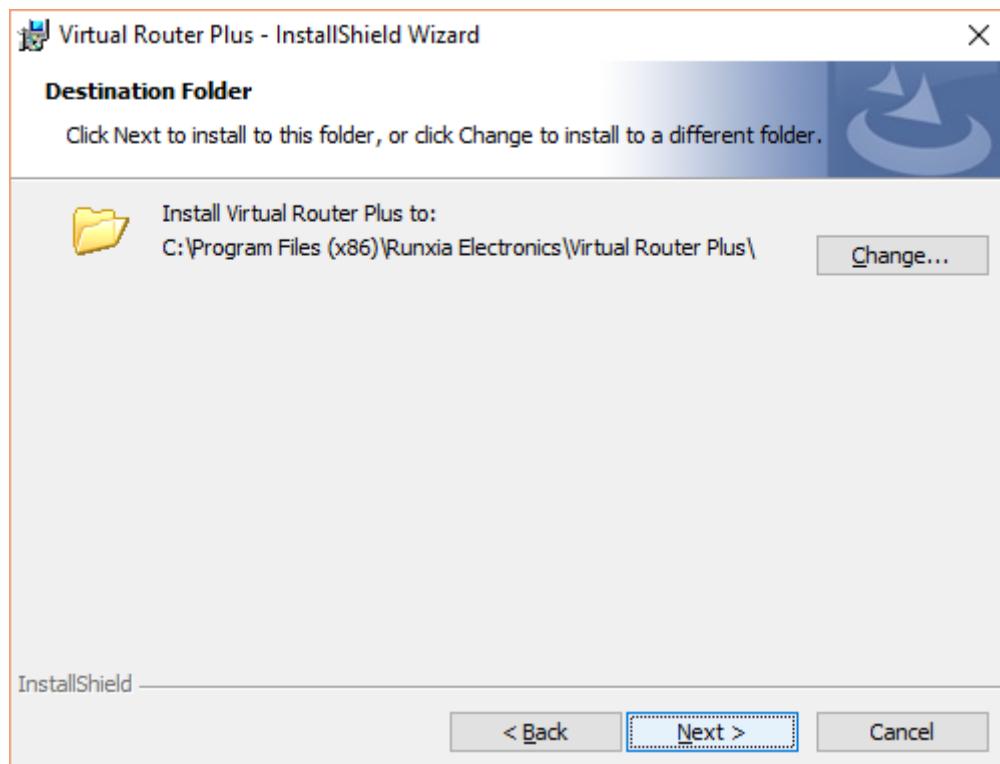


Vink "I accept the terms..." → Next

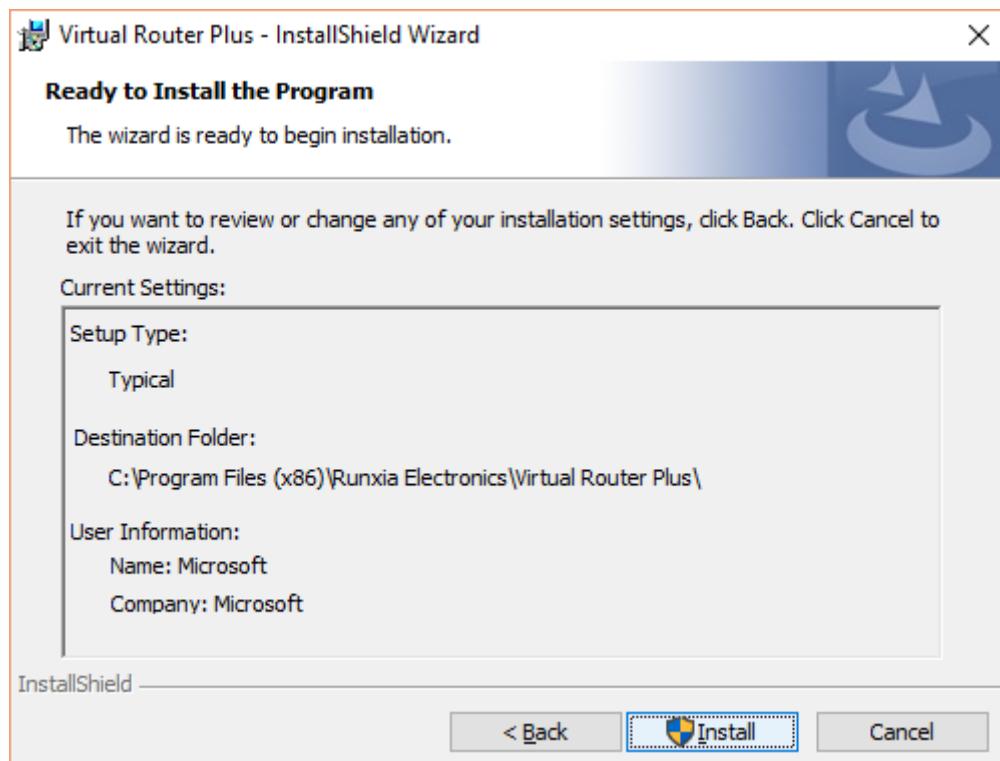


Part 1: The 35 euro IoT project

→ Next

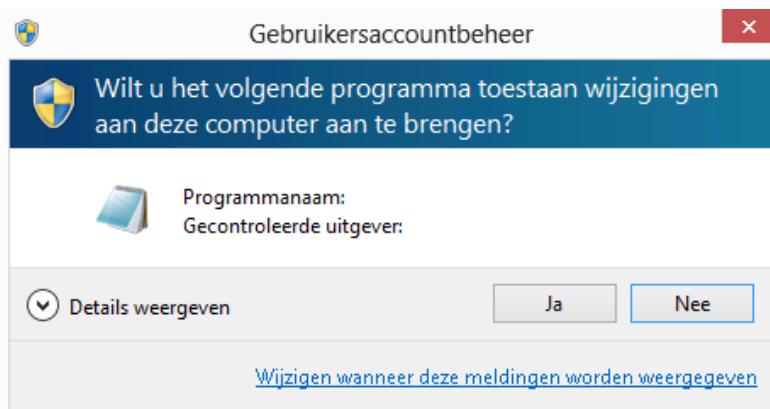


→ Install

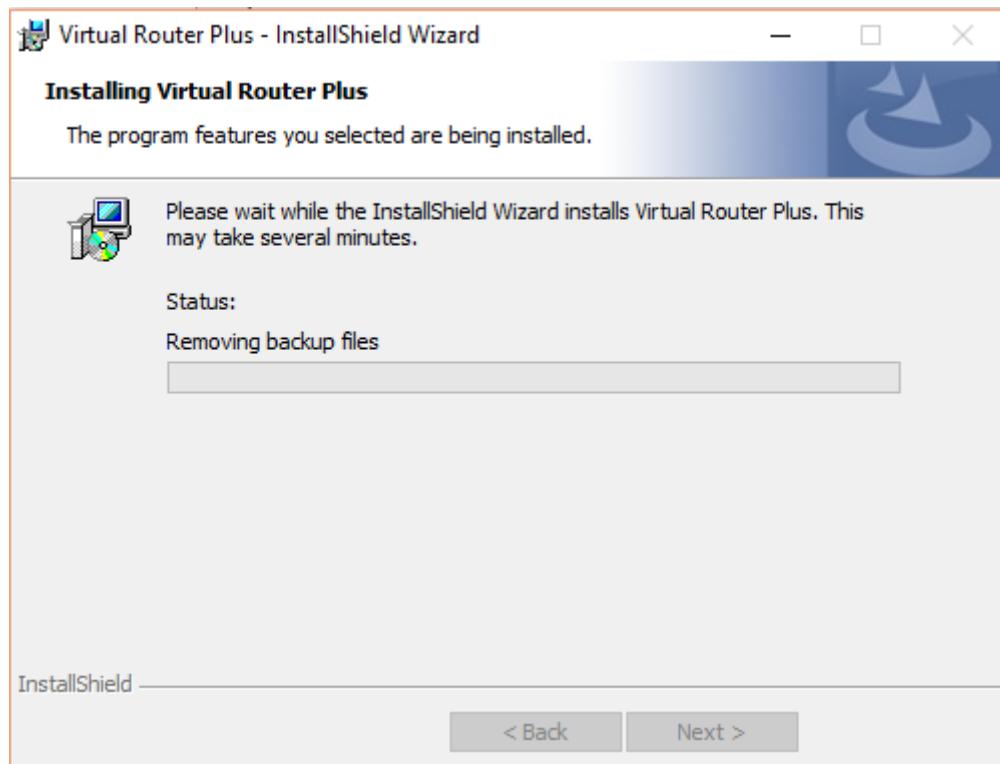


Part 1: The 35 euro IoT project

→ Ja

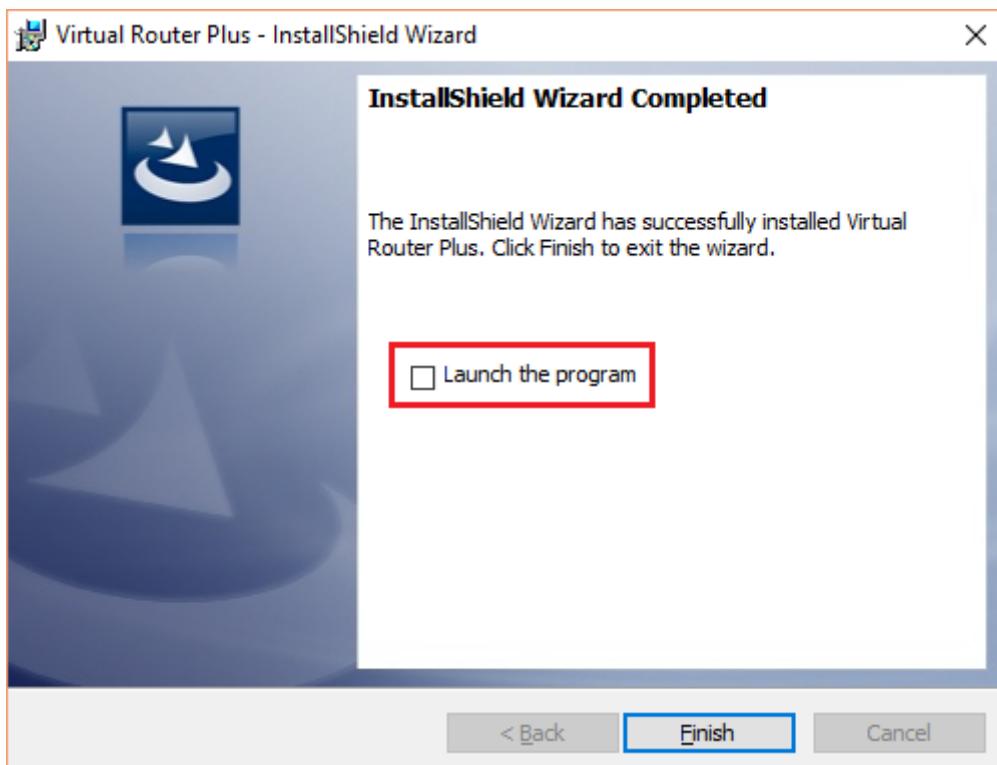


Even geduld...



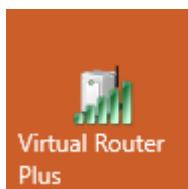
Part 1: The 35 euro IoT project

Vink "Launch the program" uit → Finish

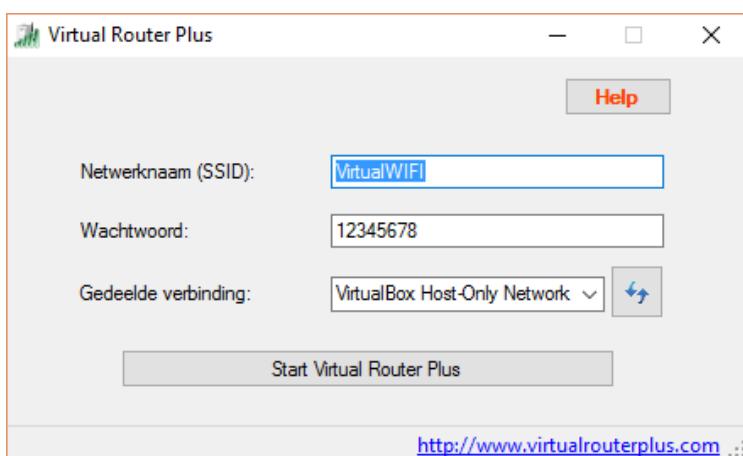


4.3.11.3 Starten Virtual Router

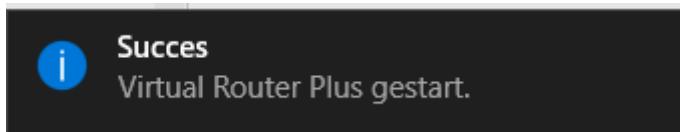
Start Virtual Router.



Wijzig eventueel de SSID en het wachtwoord → Start Virtual Router Plus



Part 1: The 35 euro IoT project



Indien het de eerste keer niet lukt het starten nogmaals proberen.

Open een command prompt → ipconfig /all

Zoek naar: Microsoft Hosted Network Virtual Adapter

```
Wireless LAN adapter LAN-verbinding* 3:

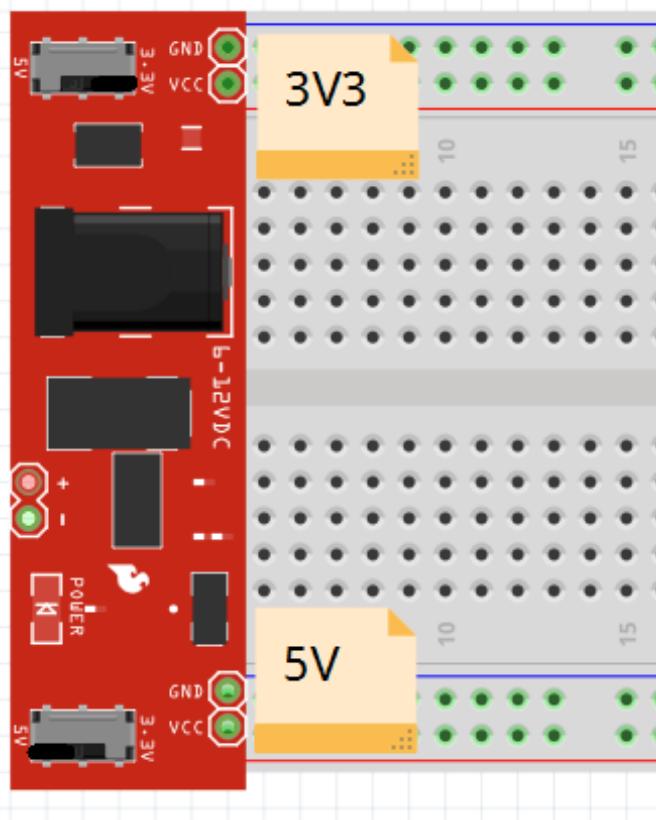
Connection-specific DNS Suffix . :
Description . . . . . : Microsoft Hosted Network Virtual Adapter
Physical Address. . . . . : 02-30-DD-03-18-12
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fc00::f0cc:7abf:3921:1f05%12(PREFERRED)
IPv4 Address. . . . . : 192.168.137.1(PREFERRED)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
DHCPv6 IAID . . . . . : 560084701
DHCPv6 Client DUID. . . . . : 00-01-00-01-1D-FA-45-65-70-54-D2-85-DB-61
DNS Servers . . . . . . . . . : fec0:0:0:ffff::1%1
                               fec0:0:0:ffff::2%1
                               fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled
```

Het getoonde IP-adres kan vervolgens gebruikt worden als alternatief voor het “normale” Wi-Fi IP-adres.

5 Bouwen

5.1 Hardware

5.1.1 Breadboard en voeding



Sluit de breadboardvoeding aan op het breadboard. Zorg ervoor dat VCC, Out of + op de rode lijn wordt aangesloten en GND of – op de zwarte of blauwe lijn.

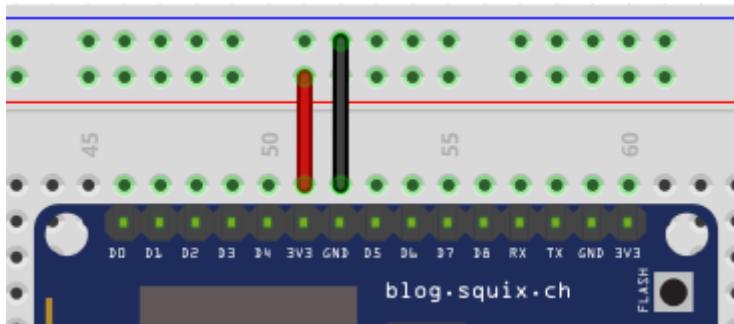
Zet de ene schakelaar van de dipswitch op 3V3 en de andere op 5V.

Gebruik een 9V-12V (1.0A) gestabiliseerde voedingsadapter.

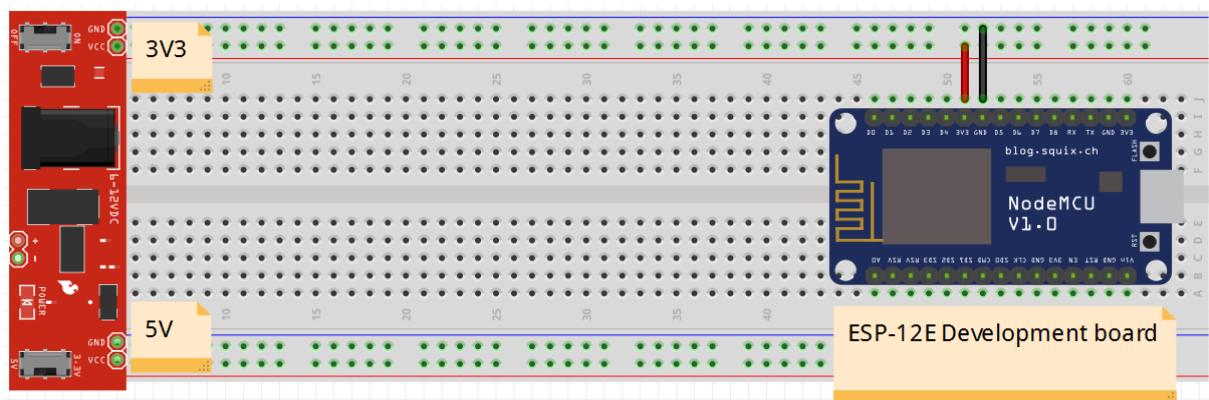
Part 1: The 35 euro IoT project

5.1.2 ESP-12E DEVKIT

Plaat de ESP-12E op het breadboard en sluit één van de 3V3 aansluiten aan op 3V3 van het breadboard (rode lijn). En één van de GND-aansluitingen op de GND-aansluiting van het breadboard (zwarte of blauwe lijn).

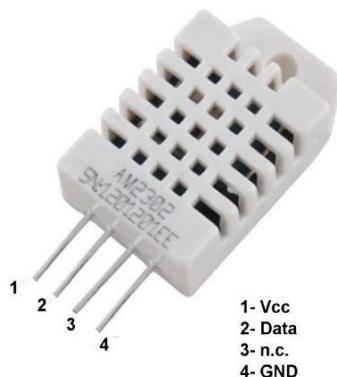


Resultaat:



Part 1: The 35 euro IoT project

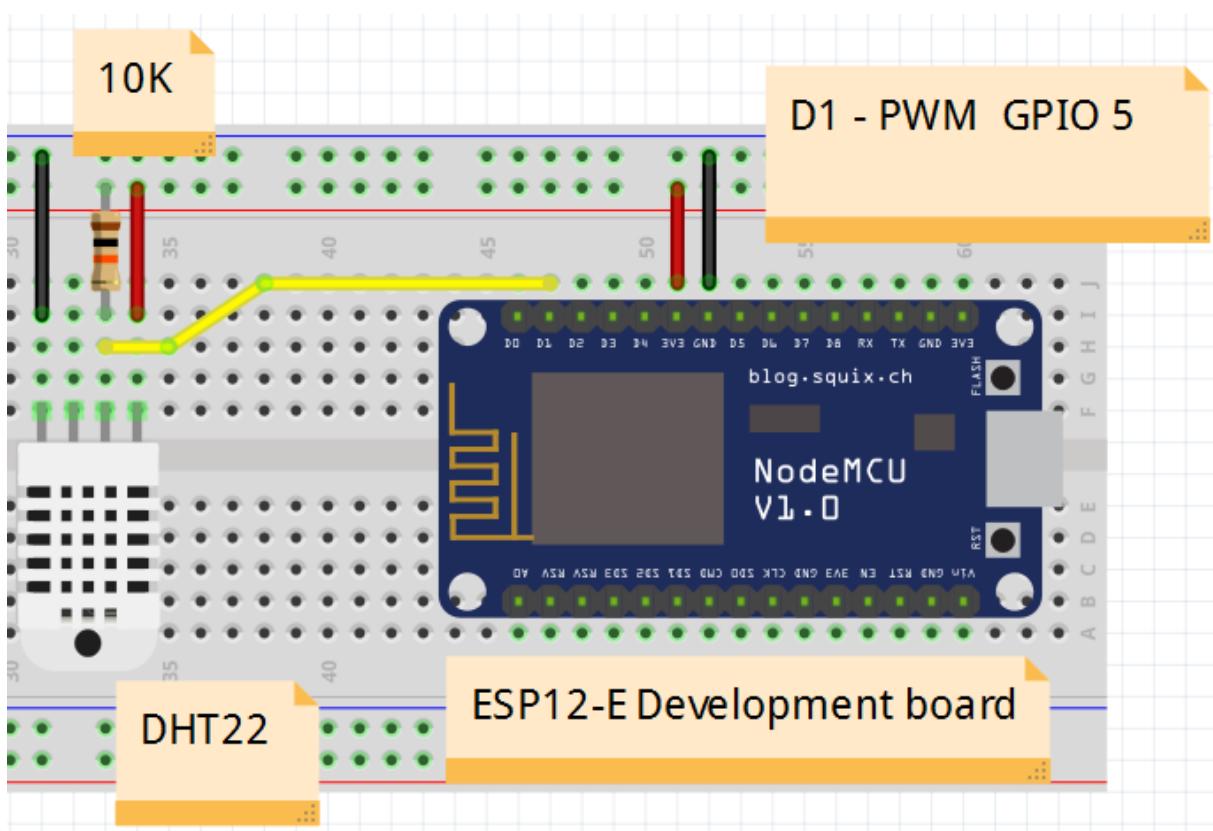
5.1.3 DHT22



De DHT22 heeft 4 pinnen, van links naar rechts VCC, Data, niet aangesloten en GND.

Verbind VCC met 3V3 op het breadboard (rode lijn) en GND met GND op het breadboard (zwarte of blauwe lijn).

Verbind Data met D1 op de ESP-12E. Plaats vervolgens een weerstand van 10K tussen de Data aansluiting en de 3V3 op het breadboard (rode lijn).

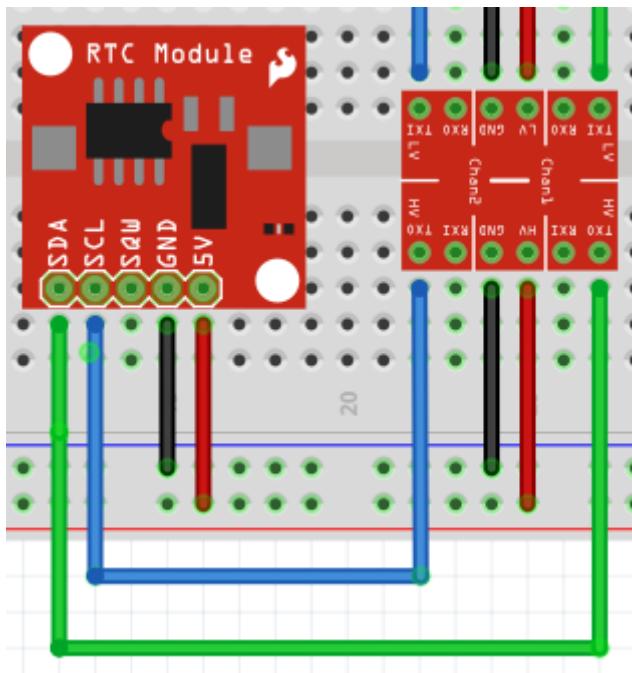


Part 1: The 35 euro IoT project

5.1.4 DS1307 RTC

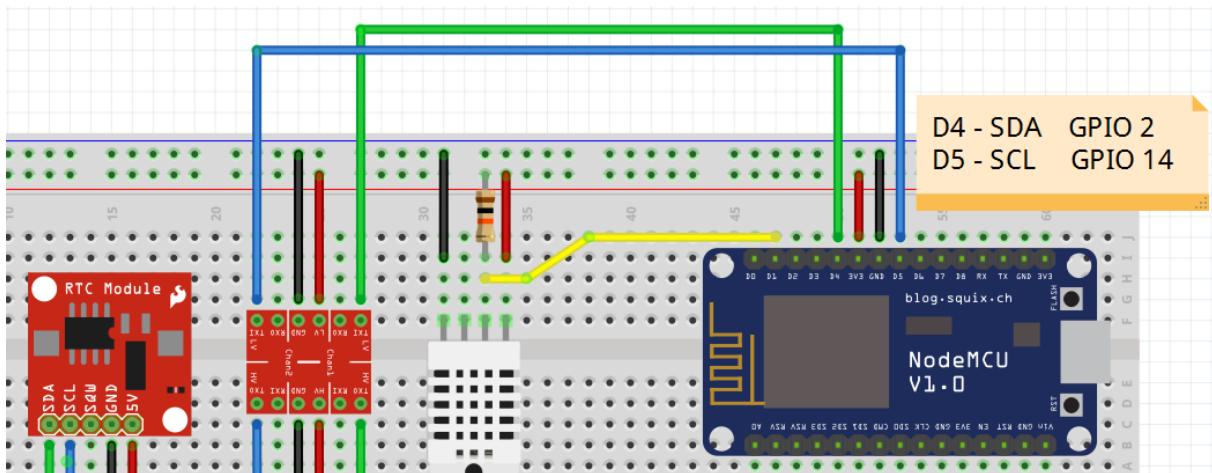
De DS1307 is een 5V component en deze dient derhalve via een level shifter te worden aangesloten op de ESP-12E.

Sluit de 5V aansluiting van de DS1307 aan op de 5V van het breadboard (rode lijn) en de GND op de GND van het breadboard (zwarte of blauwe lijn). Sluit SDA van de DS1307 aan op de ene TXO en SCL op de andere.



Zorg ervoor dat het HV (High Voltage) deel van de level shifter op het 5V deel van het breadboard wordt aangesloten en het LV (Low Voltage) deel op het 3V3 deel van het breadboard. Zorg er ook voor dat de GND-aansluitingen worden aangesloten.

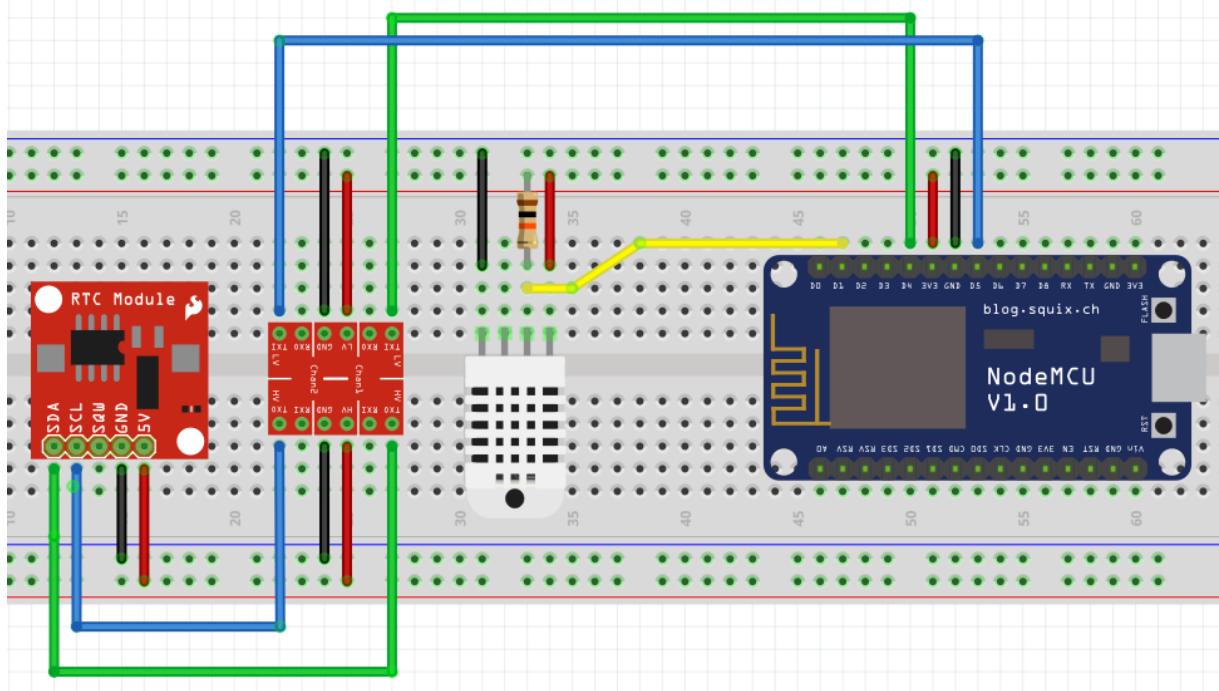
Verbind de TXI van het SDA-deel met D4 op de ESP-12E (groene draad). En verbind TXI van het SCL-deel met D5 op de ESP-12E (blauwe draad).



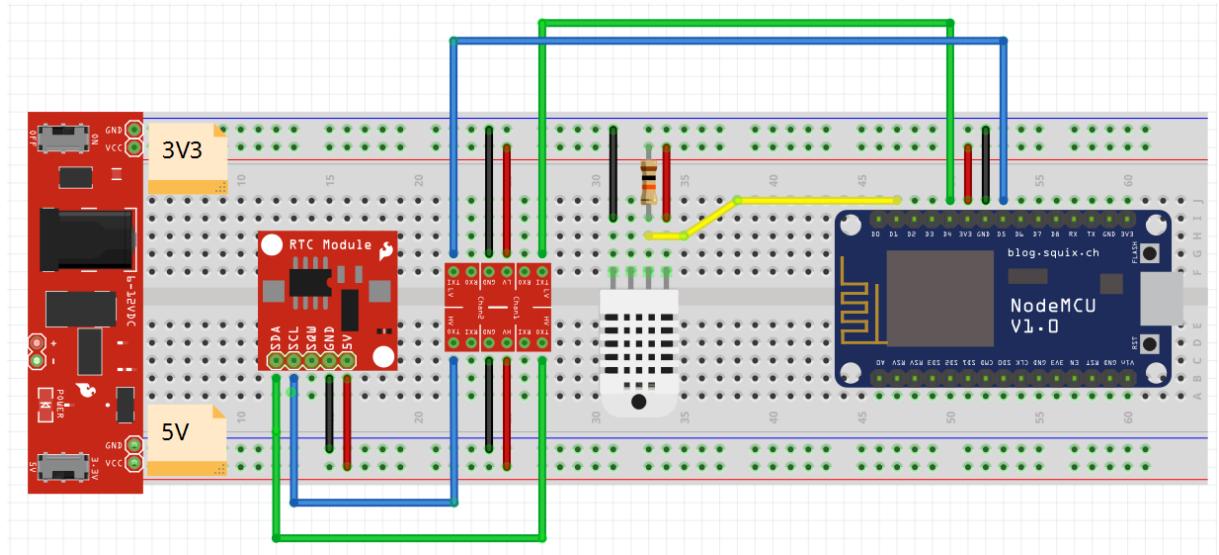
Part 1: The 35 euro IoT project

5.1.5 Hardware schema

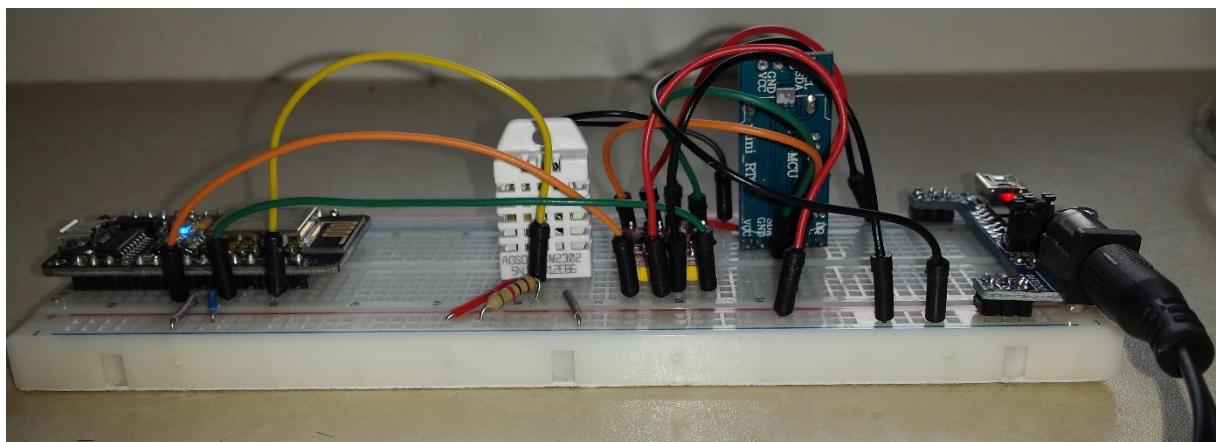
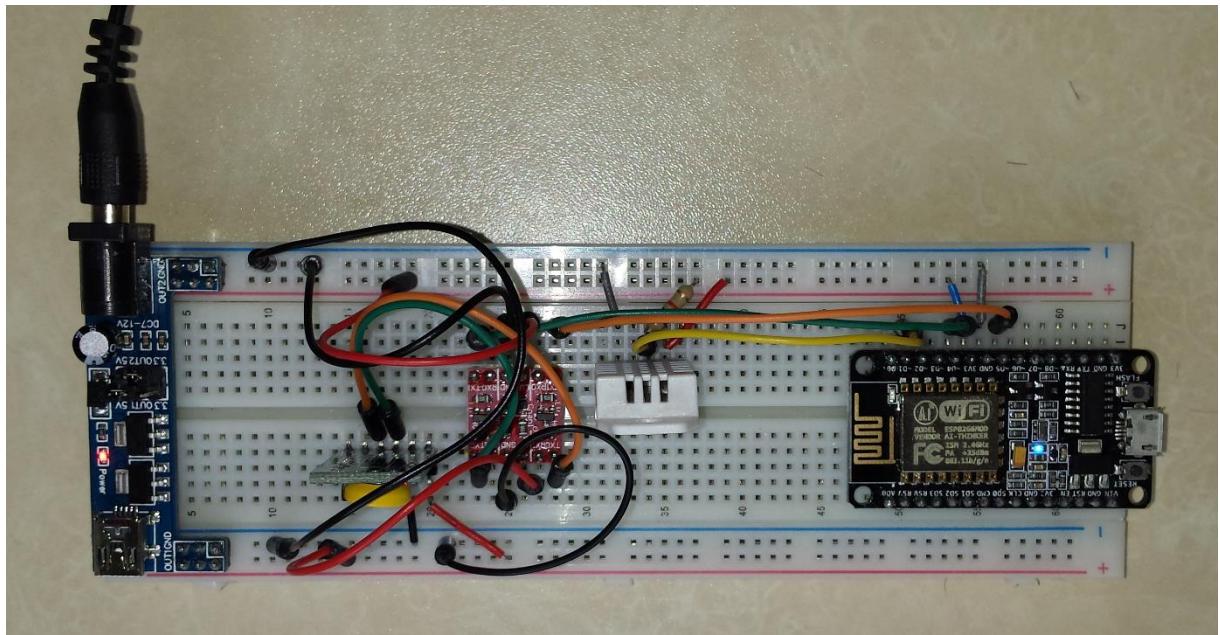
Het complete plaatje:



Het complete plaatje:



Part 1: The 35 euro IoT project



Part 1: The 35 euro IoT project

5.2 Software

5.2.1 NodeJS, Mosca en MongoDB

5.2.1.1 Inleiding

Voor het AngularJS deel wordt een NodeJS server gestart. Voor het communicatie deel wordt een MQTT-broker gestart via de Mosca implementatie. De data wordt opgeslagen in een MongoDB database.

5.2.1.2 Javascript source

Het starten van de beide servers gebeurd via een stukje javascript. Deze wordt geplaatst in de directory waar NodeJS is geïnstalleerd. In dit voorbeeld:

C:\Program Files\nodejs

Het javascript om beide servers te starten bestaat uit de volgende delen:

1. Declaratie van variabelen en gebruikt NodeJS packages. Het ipAddress dient op het eerder bepaalde IP-adres gezet te worden (Wi-Fi, mobiele hotspot of Virtual Router).
2. Daarna wordt een NodeJS Express server gestart. De server gebruikt een static implementatie. D.w.z. dat de applicaties op een vast plaats staan, in ons voorbeeld: C:\AngularJS (via ..\..\AngularJS).
3. Er wordt vervolgens een connectie gemaakt met de MongoDB database.
4. De Mosca MQTT-broker implementatie wordt gestart.
5. Er worden een zestal events (publish, subscribe, unsubscribe, connect, disconnecting en disconnect) afgevangen waarbij één of meer boodschappen op de console worden getoond.
6. Bij het publish event wordt het bericht naar de AngularJS applicatie gestuurd (end). Tevens wordt er een entry weggeschreven in de MongoDB database.

mijnserver.js

```
// 1
var express = require('express'),
serveStatic = require('serve-static');
bodyParser = require('body-parser');
mosca = require('mosca')
mongoClient = require('mongodb').MongoClient;
mongoDb = "";
mongoDbUrl = "";
dht22String = "";
dht22object = "";
accept = 0;
// Op het juiste IP-adres zetten:
ipAddress = '192.168.0.108';
nodeSettings = {
    port : 5000
};
moscaSettings = {
    host : ipAddress,
    port : 1883
};
```

Part 1: The 35 euro IoT project

```
mongoDbSettings = {
    host : '127.0.0.1',
    port : 27017
};

// 2
// Here we start NodeJS
app = express();

app.use( serveStatic('..../AngularJS') );
app.use( bodyParser.json() );

app.get('/',function(req, res, next) {
    if( accept == 1) {
        res.write( dht22String );
        res.send(JSON.stringify(res.body));
    }
});

app.listen(nodeSettings.port);
console.log('Node server is up and running')

// 3
// Here we start mongoDb
mongoDbUrl = 'mongodb://' + mongoDbSettings.host + '\:' + mongoDbSettings.port +
'/dht22iot';

mongoClient.connect(mongoDbUrl, function(err, db) {
    console.log("MongoDB connected");
    mongoDb = db;
});

// 4
// Here we start mosca
var server = new mosca.Server(moscaSettings);
server.on('ready', setup);

// 5
// Fired when the mqtt server is ready
function setup() {
    console.log('Mosca MQTT broker is up and running')
}

// Fired when a client is connected
server.on('clientConnected', function(client) {
    console.log('client connected', client.id);
});
```

Part 1: The 35 euro IoT project

```
// 6
// Fired when a message is received
server.on('published', function(packet, client) {
    var jsonString = packet.payload.toString();
    var index = jsonString.indexOf("{\"dht22\":");
    if(index > -1) {
        accept = 1;

        dht22String = jsonString;
        console.log('dht22String: ', dht22String);
        var dht22object = JSON.parse(jsonString);
        console.log('Date/time : ', dht22object.dht22.datetime);
        console.log('Temperature: ', dht22object.dht22.temperature);
        console.log('Humidity : ', dht22object.dht22.humidity);
        mongoDb.collection('dht22tempum').insertOne(
        {
            "dht22":
            {
                "datetime":dht22object.dht22.datetime,
                "temperature":dht22object.dht22.temperature,
                "humidity":dht22object.dht22.humidity
            }
        }, function(err, result) {
            if( err ) {
                console.log("Error: " + err.message);
                throw err;
            } else {
                console.log(
                    "Inserted a document into the dht22tempum collection");
            }
        });
    }
});

// Fired when a client subscribes to a topic
server.on('subscribed', function(topic, client) {
    console.log('subscribed : ', topic);
});

// Fired when a client subscribes to a topic
server.on('unsubscribed', function(topic, client) {
    console.log('unsubscribed : ', topic);
});

// Fired when a client is disconnecting
server.on('clientDisconnecting', function(client) {
    console.log('clientDisconnecting : ', client.id);
});

// Fired when a client is disconnected
server.on('clientDisconnected', function(client) {
    console.log('clientDisconnected : ', client.id);
});
```

Zie: Sources → NodeJS → mijnserver.js

Part 1: The 35 euro IoT project

5.2.2 AngularJS dashboard

5.2.2.1 Inleiding

De AngularJS app maakt gebruik van de eerder gedownloade javascript gauge.js en angular-canvas-gauge.js. Er wordt gebruik gemaakt van de volgende structuur:

Directories:

```
C:  
  +-+ AngularJS  
      +-+ dht  
          +-+ controllers  
          +-+ fonts  
          +-+ js
```

Files:

```
C:  
  +-+ AngularJS  
      +-+ dht  
          - index.html  
          +-+ controllers - controller.js  
          +-+ fonts       - digital-7-mono.ttf      *)  
          +-+ js          - digital-7-mono.eot     *)  
                          - angular.min.js        *)  
                          - angular-canvas-gauge.js  *)  
                          - angular-resource.min.js *)  
                          - gauge.js             *)
```

*) Deze bestanden zijn al eerder geïnstalleerd (zie paragraaf 4.3.4).

5.2.2.2 AngularJS HTML en JavaScript sources

index.html

Dit bestand bestaat uit de volgende delen:

In het <head> deel worden de benodigde javascripts geïmporteerd en wordt de te gebruiken style gezet. Voor de gauges wordt een viewport gedefinieerd.

In het <body> deel worden de gauges gedefinieerd. Er zijn diverse settings betreffende kleurgebruik en reeksen. De meeste settings spreken voor zichzelf.

De temperatuur meter heeft een bereik van -10 t/m 50 graden Celsius. De vochtigheidsmeter heeft een bereik van 0 t/m 100%.

Ook worden in <body> via een zogenaamde ng-directive verwijzingen naar de AngularJS app en controller opgenomen.

Part 1: The 35 euro IoT project

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>ESP12E IOT DHT12</title>
    <script type="text/javascript" src="js/angular.min.js"></script>
    <script type="text/javascript" src="js/angular-resource.min.js"></script>
    <script type="text/javascript" src="js/gauge.js"></script>
    <script type="text/javascript" src="js/angular-canvas-gauge.js"></script>
    <script type="text/javascript" src="controllers/controller.js"></script>

    <style>
        body {background-color:lightgrey;}
        h2 {color:darkblue;}
        h4 {color:darkgreen;font-family:monospace;}
    </style>
</head>

<body ng-app="dhtApp" ng-controller="dhtAppController">

    <table>
        <tr>
            <td align="center" colspan="2">
                <h2>IOT ESP12E DHT22 Temperature and Humidity</h2>
            </td>
        </tr>
        <tr>
            <td>
                <canvas canvas-gauge
                    id="myCanvas1"
                    width="{{gauge1.width}}"
                    height="{{gauge1.height}}"
                    data-value="{{gauge1.value}}"
                    data-title="Temperature"
                    data-min-value="-10"
                    data-max-value="50"
                    data-major-ticks="-10 0 10 20 30 40 50"
                    data-minor-ticks="10"
                    data-stroke-ticks="false"
                    data-units="Celcius"
                    data-value-format="1.1"
                    data-glow="true"
                    data-animation-delay="10"
                    data-animation-duration="200"
                    data-animation-fn="linear"
                    data-colors-needle="#ff0 #00f"
                    data-highlights="-10 0 #03F, 0 10 #09F, 10 25 #0F0,
                                    25 30 #F00, 30 50 #F00"
                    data-circles-outervisible="true"
                    data-circles-middlevisible="true"
                    data-circles-innervisible="true"
                    data-valuebox-visible="true"
                    data-valuetext-visible="true"
                    data-colors-plate="#cfc"
                    data-colors-title="#003"
                    data-colors-units="#030"
                    data-colors-numbers="#009"
                    data-colors-needle-start="rgba(255, 0, 0, 1)"
                    data-colors-needle-end="rgba(255, 102, 0, .9)"
                    data-colors-needle-shadowup="rgba(2, 255, 255, 0.2)"
                    data-colors-needle-shadowdown="rgba(188, 143, 143, 0.45)"
                    data-colors-needle-circle-outerstart="#f0f0f0"
                    data-colors-needle-circle-outerend="#ccc"
                    data-colors-needle-circle-innerstart="#e8e8e8"
                    data-colors-needle-circle-innerend="#f5f5f5"
                >
                </td>
            </td>
        </tr>
    </table>
</body>
```

Part 1: The 35 euro IoT project

```
data-colors-valuebox-rectstart="#888"
data-colors-valuebox-rectend="#666"
data-colors-valuebox-background="#7a0"
data-colors-valuebox-shadow="rgba(0, 0, 0, 1)"
data-colors-valuetext-foreground="#cf0"
data-colors-valuetext-shadow="rgba(0, 0, 0, 0.3)"
data-colors-circle-shadow="rgba(0, 0, 0, 0.3)"
data-colors-circle-outerstart="#ddd"
data-colors-circle-outerend="#aaa"
data-colors-circle-middlestart="#eee"
data-colors-circle-middleend="#f0f0f0"
data-colors-circle-innerstart="#fafafa"
data-colors-circle-innerend="#ccc" />
</td>
<td>
<canvas canvas-gauge
id="myCanvas2"
width="{{gauge2.width}}"
height="{{gauge2.height}}"
data-value="{{gauge2.value}}"
data-title="Humidity"
data-min-value="0"
data-max-value="100"
data-major-ticks="0 10 20 30 40 50 60 70 80 90 100"
data-minor-ticks="10"
data-stroke-ticks="false"
data-units="%"
data-value-format="1"
data-glow="true"
data-animation-delay="10"
data-animation-duration="200"
data-animation-fn="linear"
data-colors-needle="#ff0 #00f"
data-highlights="0 35 #F00, 35 40 #FF0, 40 60 #0F0,
60 65 #FF0, 65 100 #03f"
data-circles-outervisible="true"
data-circles-middlevisible="true"
data-circles-innervisible="true"
data-valuebox-visible="true"
data-valuetext-visible="true"
data-colors-plate="#cfc"
data-colors-title="#003"
data-colors-units="#030"
data-colors-numbers="#009"
data-colors-needle-start="rgba(255, 0, 0, 1)"
data-colors-needle-end="rgba(255, 102, 0, .9)"
data-colors-needle-shadowup="rgba(2, 255, 255, 0.2)"
data-colors-needle-shadowdown="rgba(188, 143, 143, 0.45)"
data-colors-needle-circle-outerstart="#f0f0f0"
data-colors-needle-circle-outerend="#ccc"
data-colors-needle-circle-innerstart="#e8e8e8"
data-colors-needle-circle-innerend="#f5f5f5"
data-colors-valuebox-rectstart="#888"
data-colors-valuebox-rectend="#666"
data-colors-valuebox-background="#7a0"
data-colors-valuebox-shadow="rgba(0, 0, 0, 1)"
data-colors-valuetext-foreground="#cf0"
data-colors-valuetext-shadow="rgba(0, 0, 0, 0.3)"
data-colors-circle-shadow="rgba(0, 0, 0, 0.3)"
data-colors-circle-outerstart="#ddd"
data-colors-circle-outerend="#aaa"
data-colors-circle-middlestart="#eee"
data-colors-circle-middleend="#f0f0f0"
data-colors-circle-innerstart="#fafafa"
data-colors-circle-innerend="#ccc" />
</td>
</tr>
```

Part 1: The 35 euro IoT project

```
<td align="center" colspan="2">
    <h4>{{datetime}}</h4>
</td>
<tr>
</tr>
</table>

</body>
</html>
```

Zie: Sources → AngularJS → dht → index.html

controller.js

In de controller is de AngularJS module dhtApp gedefinieerd. Deze maakt gebruik van o.a. de angular-canvas-gauge resource. Ook wordt de controller hier gedefinieerd. Deze heeft o.a. een http-scope. Hiermee kan de controller reageren op eventuele binnenkomende MQTT boodschappen.

De controller heeft een refresh functie die elke 2.5 seconden wordt uitgevoerd. Op basis van de nieuwe (of eerdere) invoer worden de wijzers in de gauges gezet.

Als er een http-get binnenkomt wordt de data uitgelezen en vindt er een update plaats.

Ook worden de afmetingen van de beide gauges gezet.

```
angular.module('dhtApp', ['ngResource','angular-canvas-gauge'])
.controller('dhtAppController', ['$scope', '$interval', '$http',
function($scope, $interval, $http) {
    $scope.datetime = "";

    $scope.setDate = function(date) {
        console.log('***');
        console.log('date ' + date);
        var ret =
            date.substring(6, 8) + "-" +
            date.substring(4, 6) + "-" +
            date.substring(0, 4) + " " +
            date.substring(8, 10) + ":" +
            date.substring(10, 12) + ":" +
            date.substring(12, 14);
        console.log('ret: ' + ret);
        return ret;
    };

    $scope.refresh = function() {
        $http.get('/').success(function(data) {
            console.log('***');
            console.log(data);
            mydata = data;
            console.log('***');
            console.log(mydata);
            console.log('***');
            var update = "Last update: " + $scope.setDate(
                mydata.dht22.datetime );
            $scope.datetime = update;
            $scope.gauge1.value = mydata.dht22.temperature;
            $scope.gauge2.value = mydata.dht22.humidity;
        });
    }
}];
```

Part 1: The 35 euro IoT project

```
$scope.gauge1 = {  
    value: 20,  
    width: 300,  
    height: 300  
};  
  
$scope.gauge2 = {  
    value: 20,  
    width: 300,  
    height: 300  
};  
  
$scope.intervalPromise = $interval(function() {  
    $scope.refresh();  
}, 2500);  
  
$scope.refresh();  
});  
});
```

Zie: Sources → AngularJS → dht → controllers → controller.js

5.2.3 ESP-12E MQTT programma

5.2.3.1 Inleiding

De ESP-12E wordt middels een programma gemaakt in de Arduino IDE geprogrammeerd. Sluit de ESP-12E middels een USB-kabel aan en selecteer het juiste board (NodeMCU v1.0 (ESP-12E module)) en COM-poort.

5.2.3.2 Source

De source bestaat uit de volgende delen:

1. Benodigde include statements voor de libraries die worden gebruikt door deze schets.
2. Een aantal define statements o.a. voor de door de DHTxx en RTC DS1307 gebruikte poorten op de ESP-12E.
3. Definities voor de WiFi variabelen.
4. Definities voor de NTP variabelen.
5. Create van de diverse gebruikte clients.
6. Definitie van een aantal globaal gebruikte variabelen.
7. DHTxx functies:
 - Initialisatie van de DHTxx sensor.
 - Functies voor het ophalen van de temperatuur en relatieve vochtigheid.
8. NTP functie voor het ophalen van de huidige datum en tijd.
9. RTC (DS1307) functies:
 - Zetten huidige datum/tijd indien NTP beschikbaar.
 - Bepalen en eventueel zetten datum/tijd indien NTP niet beschikbaar.
 - Datum/tijd functies.
10. MQTT client functies:
 - Opzetten connectie met MQTT broker.
 - Callback functies.
 - Reconnect functie (voor het geval de connectie is verbroken).
11. WiFi functie:

Part 1: The 35 euro IoT project

- Opzetten WiFi connectie.
12. Opmaken JSON publicatie string met datum/tijd, temperatuur en relatieve vochtigheid.
13. Arduino setup functie:
- Serial.
 - DHTxx.
 - WiFi.
 - MQTT.
 - NTP.
 - RTC.
14. Arduino loop functie:
- Check de client connectie en reconnect indien nodig.
 - Check elke 5 seconden de status van de DHTxx.
 - Bepaal temperatuur en relatieve luchtvochtigheid.
 - Publiceer JSON string richting MQTT broker.

IOT_Project.ino

Zie: Sources → Arduino → IOT-Project → IOT-Project.ino

```
// 1
// -----
// Includes
// -----
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <RTClib.h>
#include <NtpClientLib.h>
#include <ArduinoJson.h>

// 2
// -----
// Defines
// -----
// DHTxx Temperature and Humidity Sensor
// xx = 11 or 22
// GPIO 5 = D1
#define DHTPIN 5
#define DHTTYPE DHT22

// DS1307 Real Time Clock (RTC)
// GPIO 2 = D4 - SDA
// GPIO 14 = D5 - SCL
#define RTC_SDA 2
#define RTC_SCL 14
```

Part 1: The 35 euro IoT project

```
// 3
// -----
// Wifi / MQTT broker
// -----
// Update these with values suitable for your network

#define WIFI_HOME
//#define WIFI_MOBILE
//#define VIRTUAL_ROUTER

#ifdef WIFI_HOME
// WiFi home
const char * ssid = "?????????";
const char * password = "?????????";
const char * mqtt_broker = "192.168.0.108";
#else
#ifdef WIFI_MOBILE
// WiFi mobile
const char * ssid = "?????????";
const char * password = "?????????";
const char * mqtt_broker = "192.168.43.100";
#else
#ifdef VIRTUAL_ROUTER
// Virtual Router
const char * ssid = "?????????";
const char * password = "?????????";
const char * mqtt_broker = "192.168.137.1";
#endif
#endif
#endif

// MQTT broker portnumber
const int portNumber = 1883;

// 4
// -----
// NTP server
// -----
const char * ntp_server = "nl.pool.ntp.org";      // NTP pool server
const int timezone = 1;                            // GMT -/+ value
const bool isDST = true;                          // Daylight Saving Time (DST)

// 5
// -----
// Clients
// -----
// NTP client
ntpClient * ntp;

// WiFi Client
WiFiClient espClient;

// MQTT Client
PubSubClient client(espClient);

// DHTxx init (xx = 11 or 22)
DHT dht(DHTPIN, DHTTYPE);

// DS1307 init
RTC_DS1307 rtc;
```

Part 1: The 35 euro IoT project

```
// 6
// -----
// Globals
// -----
long millisPrevMsg = 0;
char sDateTime[32];
char sTemperature[8];
char sHumidity[8];

// 7
// -----
// DHTxx
// -----
// Setup DHT
void setup_DHT() {
    float temp = 0.0;

    // Start the communication with the DHT sensor by calling the begin method of
    // the dht object
    Serial.print("DHT setup ");
    dht.begin();

    // DHTxx is a slow sensor, wait for communication with the sensor
    do {
        digitalWrite(BUILTIN_LED, HIGH);
        delay(500);
        temp = dht.readTemperature();
        Serial.print(".");
        digitalWrite(BUILTIN_LED, LOW);
        delay(500);
    }
    while (isnan(temp));
    Serial.println();
    Serial.println("DHT ready");
    Serial.println();
}

// Get temperature reading from sensor
bool getTemperature() {
    char buffer[8];
    char sTemp[8];
    float temp = dht.readTemperature();
    if (isnan(temp)) {
        return false;
    }

    // Temperature: 99.99 to 9.9 (25.11 -> 25.1 / 7.65 --> 7.6 / 0.15 -> 0.1)
    dtostrf(temp, 1, 1, buffer);
    sprintf(sTemperature, "%s", buffer);
    return true;
}

// Get humidity reading from sensor
bool getHumidity() {
    char buffer[8];
    char sHum[8];
    float hum = dht.readHumidity();
    if (isnan(hum)) {
        return false;
    }

    // Humidity: 99.99 to 9 (25.11 -> 25 / 7.65 --> 7 / 0.15 -> 0)
    dtostrf(hum, 1, 0, buffer);
    sprintf(sHumidity, "%s", buffer);
    return true;
}
```

Part 1: The 35 euro IoT project

```
// 8
// -----
// NTP
// -----
// Setup NTP client
bool setup_NTP() {
    String date = "";
    bool isNotNTPInit = true;
    int counter = 0;

    // Setup NTP client
    ntp = ntpClient::getInstance(ntp_server, timezone, isDST);
    ntp->begin(); //Starts time synchronization

    // Try to connect to the NTP server
    delay(1000);
    Serial.print("Connecting to NTP ");
    while (isNotNTPInit) {
        digitalWrite(BUILTIN_LED, HIGH);
        Serial.print(".");
        date = ntp->getDateStr();

        // When date is in initial state (01/01/1970) connection is not succesfull
        if (date != "01/01/1970") {
            // Succesfull connected
            isNotNTPInit = false;
        } else {
            // NTP servers are very busy so try again...
            delay(500);
            digitalWrite(BUILTIN_LED, LOW);
            delay(500);
            ++counter;
            // Limit number of retries
            if (counter > 20) {
                Serial.println();
                Serial.println("NTP setup failed");
                return false;
            }
        }
    }
    Serial.println();

    Serial.print("NTP setup: ");
    Serial.print(ntp->getTimeStr());
    Serial.print(" ");
    Serial.println(ntp->getDateStr());
    Serial.println();
    return true;
}

// 9
// -----
// RTC DS1307
// -----
// Get date time in format yyyyMMddhhmmss from rtc
void getDateTIme() {
    DateTime now = rtc.now();
    snprintf(sDateTime, 32, "%04d%02d%02d%02d%02d", now.year(), now.month(),
             now.day(), now.hour(), now.minute(), now.second());
}
```

Part 1: The 35 euro IoT project

```
// Get date time in format yyyyMMddhhmmss from millis
void getDateTimeMillis() {
    DateTime now = DateTime(millis());
    sprintf(sDateTime, 32, "%04d%02d%02d%02d%02d", now.year(), now.month(),
           now.day(), now.hour(), now.minute(), now.second());
    Serial.print("Millis ");
    Serial.println(sDateTime);
}

// Show time and date in format hh:mm:ss dd-MM-yyyy
void showTimeDate() {
    char buffer[32];
    DateTime now = rtc.now();
    sprintf(buffer, 32, "%02d:%02d:%02d %02d-%02d-%04d", now.hour(), now.minute(),
            now.second(), now.day(), now.month(), now.year());
    Serial.print("Time/date: ");
    Serial.println(buffer);
}

// Setup RTC
bool setup_RTC(bool ntpStatus) {
    digitalWrite(BUILTIN_LED, HIGH);
    Wire.begin(RTC_SDA, RTC_SCL);
    Serial.println("RTC setup");

    // RTC begin
    if (! rtc.begin() ) {
        Serial.println("Couldn't find RTC");
        return false;
    }

    // Get date time from millis
    getDateTimeMillis();

    // If ntp available adjust rtc
    if ( ntpStatus ) {
        // Adjust ntp time
        String ntpTime = ntp->getTimeStr();
        String ntpDate = ntp->getDateStr();

        int hr = ntpTime.substring(0, 2).toInt();
        int mn = ntpTime.substring(3, 5).toInt();
        int sc = ntpTime.substring(6, 8).toInt();

        int dy = ntpDate.substring(0, 2).toInt();
        int mt = ntpDate.substring(3, 5).toInt();
        int yr = ntpDate.substring(6, 11).toInt();

        rtc.adjust(DateTime(yr, mt, dy, hr, mn, sc));
        Serial.print("NTP -> RTC adjust: ");
    } else {
        // Otherwise check RTC date time
        DateTime now = rtc.now();

        // If RTC time is invalid take builtin date and time
        if ( now.second() < 0 || now.second() > 59 ) {
            rtc.adjust(DateTime(F(__DATE__)), F(__TIME__));
        }

        // Check if time is valid
        if ( now.second() < 0 || now.second() > 59 ) {
            digitalWrite(BUILTIN_LED, LOW);
            Serial.println("RTC call failed!");
            return false;
        }
    }
}
```

Part 1: The 35 euro IoT project

```
// OK!
showTimeDate();
Serial.println("RTC OK");
Serial.println();
digitalWrite(BUILTIN_LED, LOW);
return true;
}

// 10
// -----
// MQTT
// -----
// Client callback function
void callback(char* topic, byte* payload, unsigned int length) {
    digitalWrite(BUILTIN_LED, HIGH);
    Serial.print("Message arrived: [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
    digitalWrite(BUILTIN_LED, LOW);
}

// Try to reconnect client
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        digitalWrite(BUILTIN_LED, HIGH);
        Serial.print("Attempting MQTT connection: ");
        // Attempt to connect
        if (client.connect("dht22publish")) {
            Serial.println("connected");
            // Once connected, publish an announcement...
            client.publish("outTopic", "ESP8266Client connected!");
            // ... and resubscribe
            client.subscribe("inTopic");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(": try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(1000);
            digitalWrite(BUILTIN_LED, LOW);
            delay(4000);
        }
    }
}

void setup_MQTTclient() {
    client.setServer(mqtt_broker, portNumber);
    client.setCallback(callback);
    Serial.print("MQTT client connected to: ");
    Serial.print(mqtt_broker);
    Serial.print(":");
    Serial.println(portNumber);
    Serial.println();
}
```

Part 1: The 35 euro IoT project

```
// 11
// -----
// WiFi
// -----
// Setup WiFi
void setup_WiFi() {
    // We start by connecting to a WiFi network
    Serial.print("Connecting to ");
    Serial.print(ssid);
    Serial.print(" ");

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        digitalWrite(BUILTIN_LED, HIGH);
        delay(250);
        Serial.print(".");
        digitalWrite(BUILTIN_LED, LOW);
        delay(250);
    }

    Serial.println();
    Serial.print("WiFi connected to: ");
    Serial.println(WiFi.localIP());
    Serial.println();
}

// 12
// -----
// JSON
// -----
const char * getJSONString() {
    // Setup JSON objects
    String jsonString;
    StaticJsonBuffer<128> jsonBuffer;
    JsonObject& root = jsonBuffer.createObject();
    JsonObject& data = jsonBuffer.createObject();
    data["datetime"] = sDateTime;
    data["temperature"] = sTemperature;
    data["humidity"] = sHumidity;
    root.set("dht22", data);
    root.printTo(jsonString);
    Serial.println("Message: ");
    root.prettyPrintTo(Serial);
    Serial.println();

    // Return JSON string
    return jsonString.c_str();
}
```

Part 1: The 35 euro IoT project

```
// 13
// -----
// Main setup
// -----
void setup() {
    pinMode(BUILTIN_LED, OUTPUT);
    Serial.begin(115200);
    delay(50);
    Serial.flush();

    // Setup DHT
    setup_DHT();

    // Connect to Wifi
    setup_WiFi();

    // Connect MQTT client to broker
    setup_MQTTclient();

    // NTP client setup
    bool ntpStatus = setup_NTP();

    // Setup RTC
    if (setup_RTC(ntpStatus) == false) {
        while (1) {
            // Setup error: bring esp in endless loop
            digitalWrite(BUILTIN_LED, HIGH);
            delay(150);
            digitalWrite(BUILTIN_LED, LOW);
            delay(150);
        }
    }

    // Give millis previous message initial value
    millisPrevMsg = millis();
    Serial.println("Setup completed");
    Serial.println();
}
```

Part 1: The 35 euro IoT project

```
// 14
// -----
// Main loop
// -----
void loop() {
    // Check if client is connected
    if (!client.connected()) {
        // Reconnect client
        reconnect();
    }
    // Make client active
    client.loop();

    // Send message every 5 seconds
    long now = millis();
    if (now - millisPrevMsg > 5000) {
        digitalWrite(BUILTIN_LED, HIGH);

        // Get data: datetime, temperature and humidity
        getDateTime();
        if (getTemperature() == false) {
            delay(1000);
            digitalWrite(BUILTIN_LED, LOW);
            Serial.println("Failed to read from DHT sensor [T]");
            return;
        }
        if (getHumidity() == false) {
            delay(1000);
            digitalWrite(BUILTIN_LED, LOW);
            Serial.println("Failed to read from DHT sensor [H]");
            return;
        }

        // Publish results
        client.publish("outTopic", getJSONString());
        Serial.println("Message published");
        Serial.println();
        millisPrevMsg = now;
        delay(500);
        digitalWrite(BUILTIN_LED, LOW);
    } else {
        // Manual delay loop
        delay(250);
    }
}

// -----
```

Opmerkingen:

Indien parsen fout gaat, kan het volgende een oplossing zijn in de Arduino code onder //12:

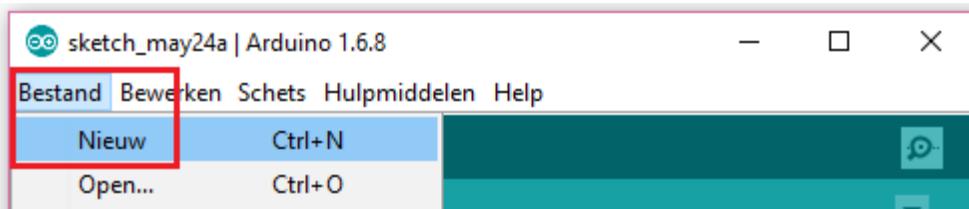
```
const char * getJSONString() {
vervangen door:
const String getJSONString() {

    client.publish("outTopic", getJSONString());
vervangen door:
    client.publish("outTopic", getJSONString().c_str());
```

Part 1: The 35 euro IoT project

5.2.3.3 Code uploaden

Start de Arduino IDE → menu → Bestand → Nieuw



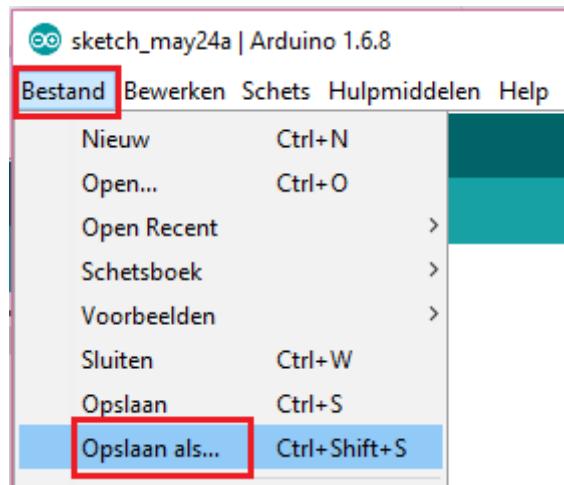
Vervang de nieuwe schets door de volledige source:

A screenshot of the Arduino IDE code editor. The title bar says 'sketch_may24a | Arduino 1.6.8'. The code editor window shows the following C++ code:

```
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <DHT.h>
#include <RTClib.h>
#include <NtpClientLib.h>

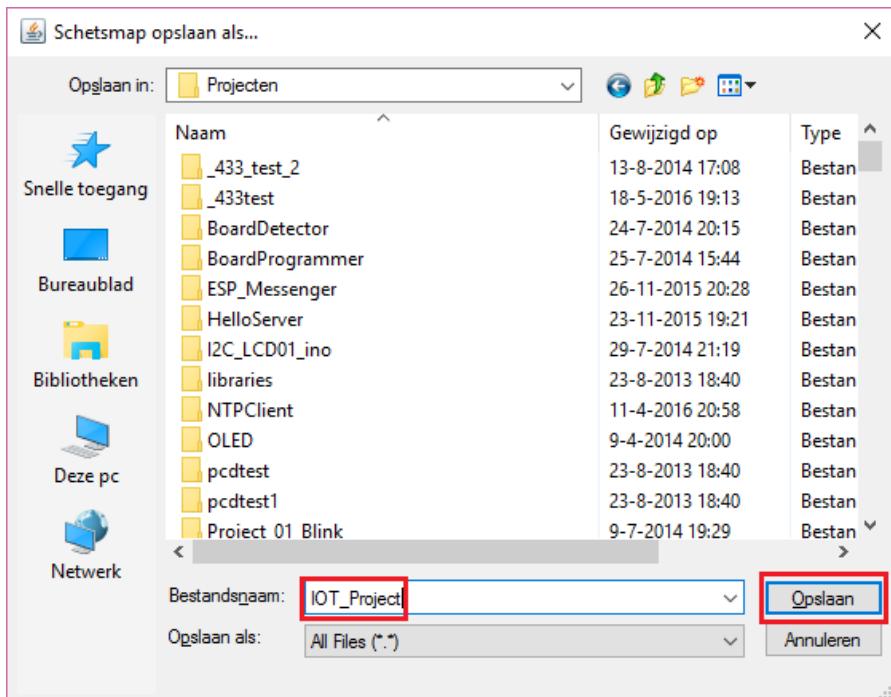
// DHTxx Temperature and Humidity Sensor
// xx = 11 or 22
// GPIO 5 = D1
#define DHTPIN 5
#define DHTTYPE DHT22
```

Menu → Bestand → Opslaan als...

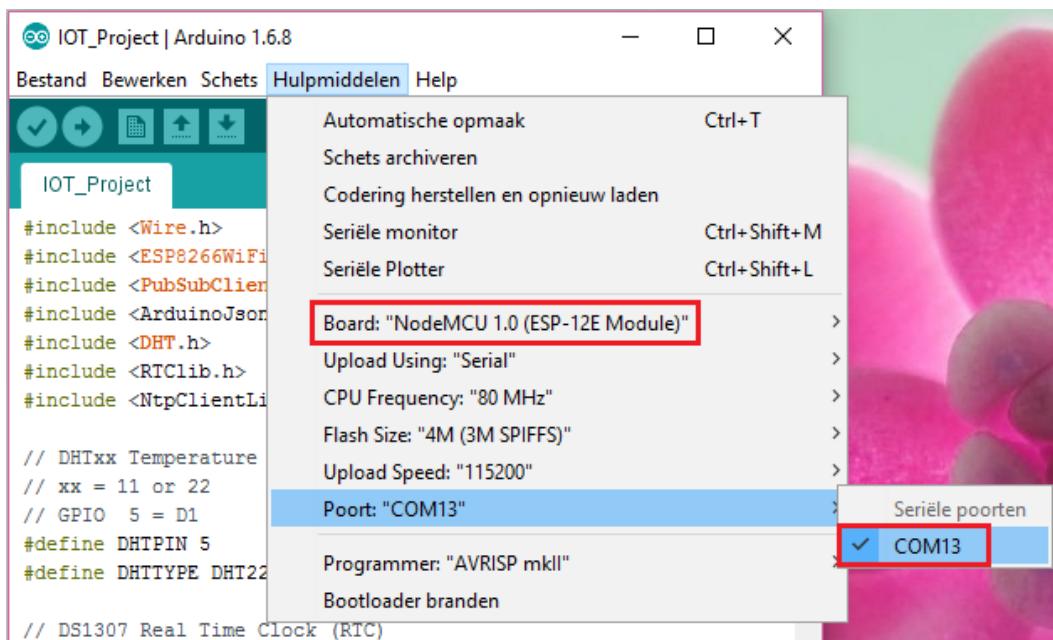


Part 1: The 35 euro IoT project

Bestandsnaam: IOT_Project → Opslaan

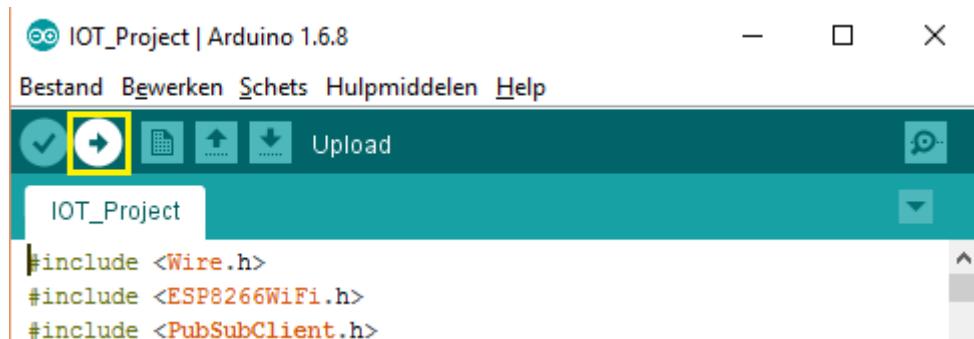


Controleer of het juiste board en de juiste COM-poort zijn geselecteerd.



Part 1: The 35 euro IoT project

Upload via 



IOT_Project | Arduino 1.6.8

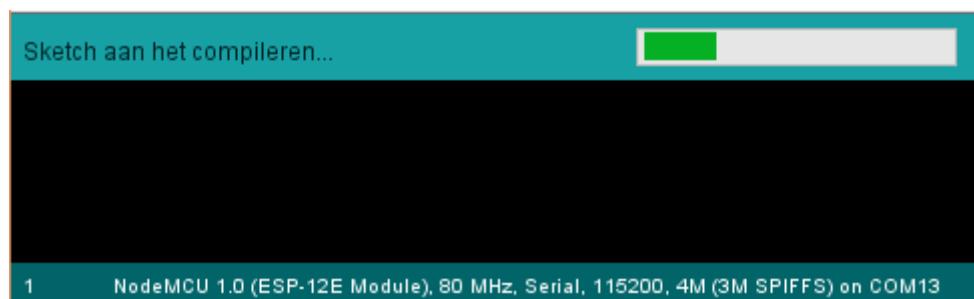
Bestand Bewerken Schets Hulpmiddelen Help

Upload

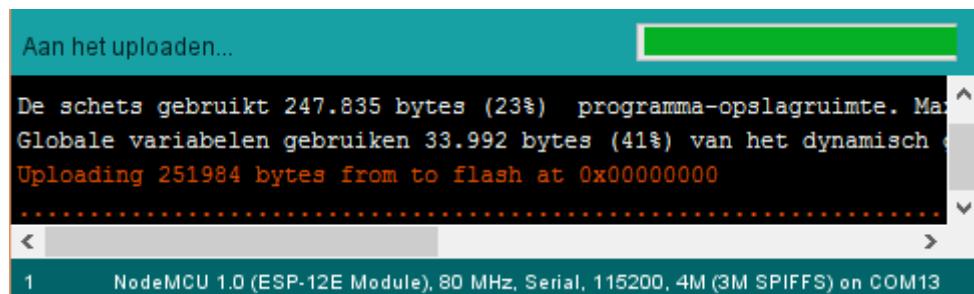
IOT_Project

```
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

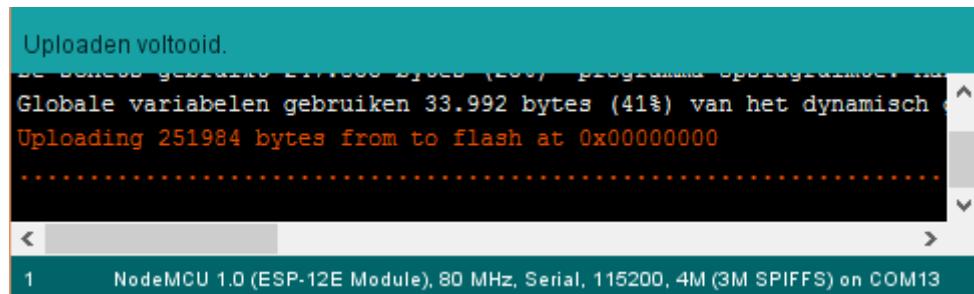
Even geduld...



Uploaden...



Klaar...



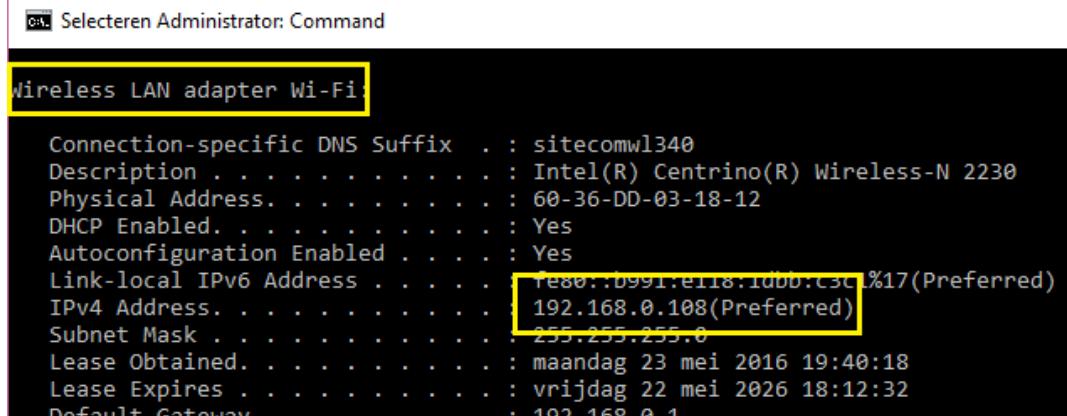
6 Testen

6.1 ESP-12E en WiFi

6.1.1 WiFi thuis

Open een command prompt en geeft het volgende commando: ipconfig /all

Zoek naar Wireless LAN adapter Wi-Fi



```
Windows Selecteren Administrator: Command

Wireless LAN adapter Wi-Fi:

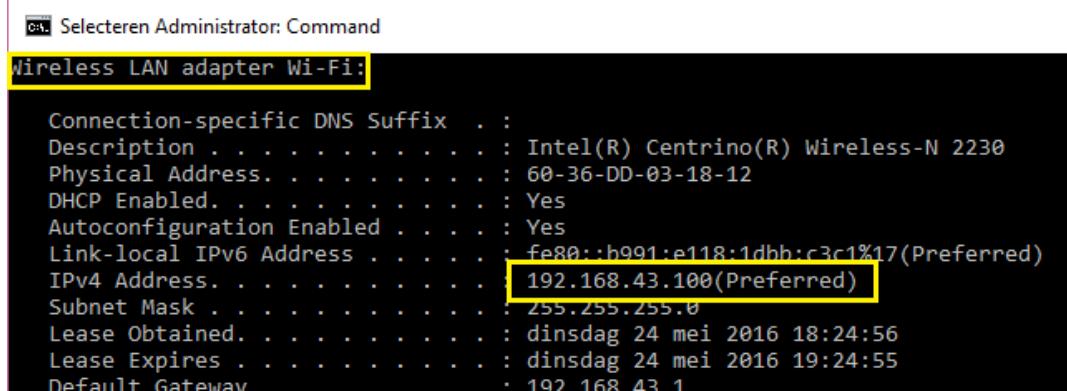
Connection-specific DNS Suffix . : sitecomwl340
Description . . . . . : Intel(R) Centrino(R) Wireless-N 2230
Physical Address. . . . . : 60-36-DD-03-18-12
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::b991:e118:1dbb:c3c%17(PREFERRED)
IPv4 Address. . . . . : 192.168.0.108(PREFERRED)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : maandag 23 mei 2016 19:40:18
Lease Expires . . . . . : vrijdag 22 mei 2026 18:12:32
Default Gateway . . . . . : 192.168.0.1
```

Het getoonde IP-adres dient voor de MQTT-broker te gebruiken.

6.1.2 Mobiele Hotspot

Start de mobiele hotspot op de Smartphone. Open een command prompt en geeft het volgende commando: ipconfig /all

Zoek naar Wireless LAN adapter Wi-Fi



```
Windows Selecteren Administrator: Command

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : 
Description . . . . . : Intel(R) Centrino(R) Wireless-N 2230
Physical Address. . . . . : 60-36-DD-03-18-12
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::b991:e118:1dbb:c3c%17(PREFERRED)
IPv4 Address. . . . . : 192.168.43.100(PREFERRED)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : dinsdag 24 mei 2016 18:24:56
Lease Expires . . . . . : dinsdag 24 mei 2016 19:24:55
Default Gateway . . . . . : 192.168.43.1
```

Het getoonde IP-adres dient voor de MQTT-broker te gebruiken.

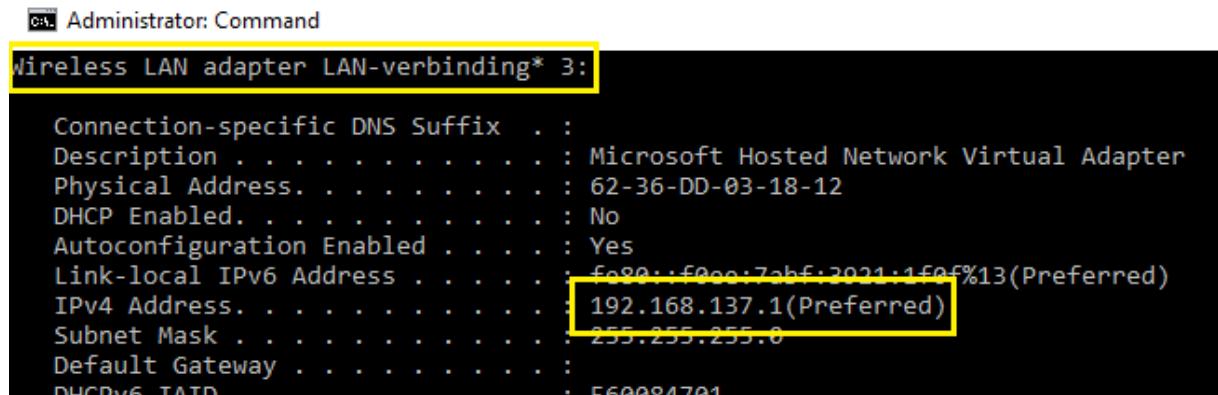
Part 1: The 35 euro IoT project

6.1.3 Virtual Router

Start Virtual Router.

Open een command prompt en geeft het volgende commando: ipconfig /all

Zoek naar Wireless LAN adapter LAN-verbinding*



```
c:\ Administrator: Command  
Wireless LAN adapter LAN-verbinding* 3:  
  
Connection-specific DNS Suffix . :  
Description . . . . . : Microsoft Hosted Network Virtual Adapter  
Physical Address. . . . . : 62-36-DD-03-18-12  
DHCP Enabled. . . . . : No  
Autoconfiguration Enabled . . . . . : Yes  
Link-local IPv6 Address . . . . . : fe80::f0ee:7abf%2021:1f0f%13(PREFERRED)  
IPv4 Address. . . . . : 192.168.137.1(PREFERRED)  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . :  
DHCPv6 TAIID . . . . . : E60084701
```

Het getoonde IP-adres dient voor de MQTT-broker te gebruiken.

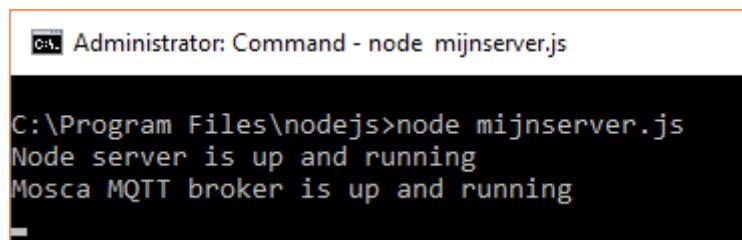
6.2 ESP-12E, NodeJS en Mosca

6.2.1 Starten NodeJS Express server en Mosca broker

Open een command prompt en ga naar: C:\Program Files\nodejs

Start de broker met de commando's:

```
cd \Program Files\nodejs  
node mijnserver.js
```

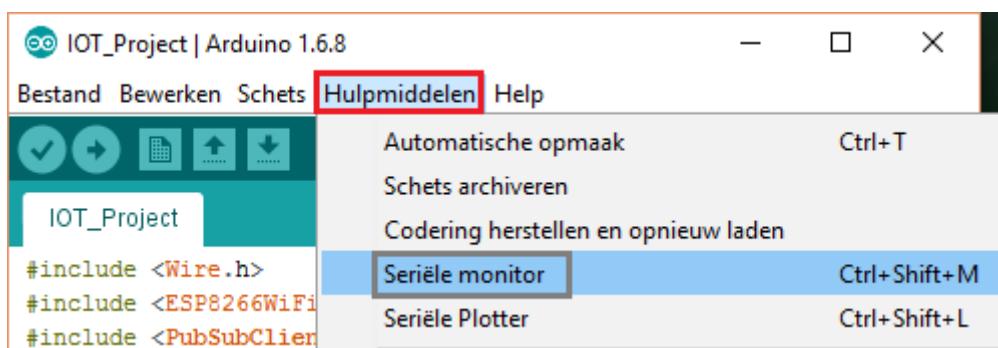


```
c:\ Administrator: Command - node mijnserver.js  
  
C:\Program Files\nodejs>node mijnserver.js  
Node server is up and running  
Mosca MQTT broker is up and running  
-
```

Part 1: The 35 euro IoT project

6.2.2 Starten ESP-12E

Open de Arduino IDE en sluit de ESP-12E aan op de voeding en de USB.
Arduino IDE → menu → Hulpmiddelen → Seriële monitor



Console:

```
COM13

.

WiFi connected
IP address:
192.168.0.13

Connecting to ntp .NTP setup: 19:19:08 24/05/2016

RTC init!
Millis 21060206075632
NTP -> RTC adjust: 19:19:08 24-05-2016
RTC OK!

Attempting MQTT connection...connected
Temperature: 22.0 C - Humidity: 56%RH
```

Op de command prompt waarop de MQTT-broker is gestart worden de gepubliceerde berichten getoond:

```
C:\Program Files\nodejs>node mijnserver.js
Node server is up and running
Mosca MQTT broker is up and running
client connected dht22publish
subscribed : inTopic
hdt22String: {"hdt22": {"datetime": "20160524192252", "temperature": "22.2", "humidity": "55"}}
Date/time : 20160524192252
Temperature: 22.2
Humidity   : 55
hdt22String: {"hdt22": {"datetime": "20160524192257", "temperature": "22.2", "humidity": "55"}}
Date/time : 20160524192257
Temperature: 22.2
Humidity   : 55
```

Part 1: The 35 euro IoT project

6.2.3 Starten Chrome browser

Open een browser, bij voorkeur een Chromebrowser.

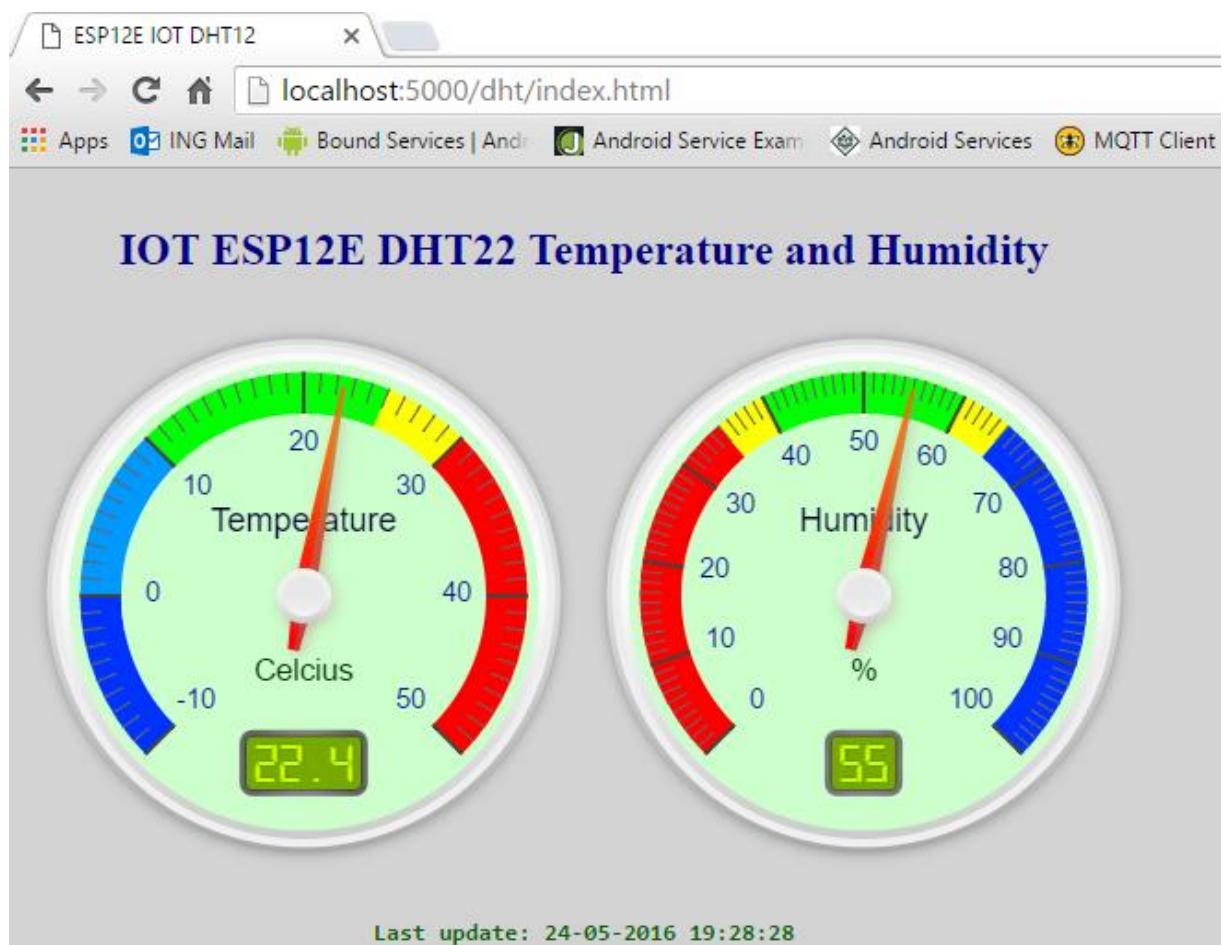
Gebruik het IP-adres van de MQTT-broker, in dit voorbeeld 192.168.0.108 of localhost.

Ga naar:

<http://localhost:5000/dht/index.html>

of

<http://192.168.0.108:5000/dht/index.html>



Vergelijk dit met de laatste stand op command prompt van de MQTT-broker:

```
Humidity : 55
hdt22String: {"hdt22":{"datetime":"20160524192848","temperature":"22.4","humidity":"55"}}
Date/time : 20160524192848
Temperature: 22.4
Humidity : 55
```

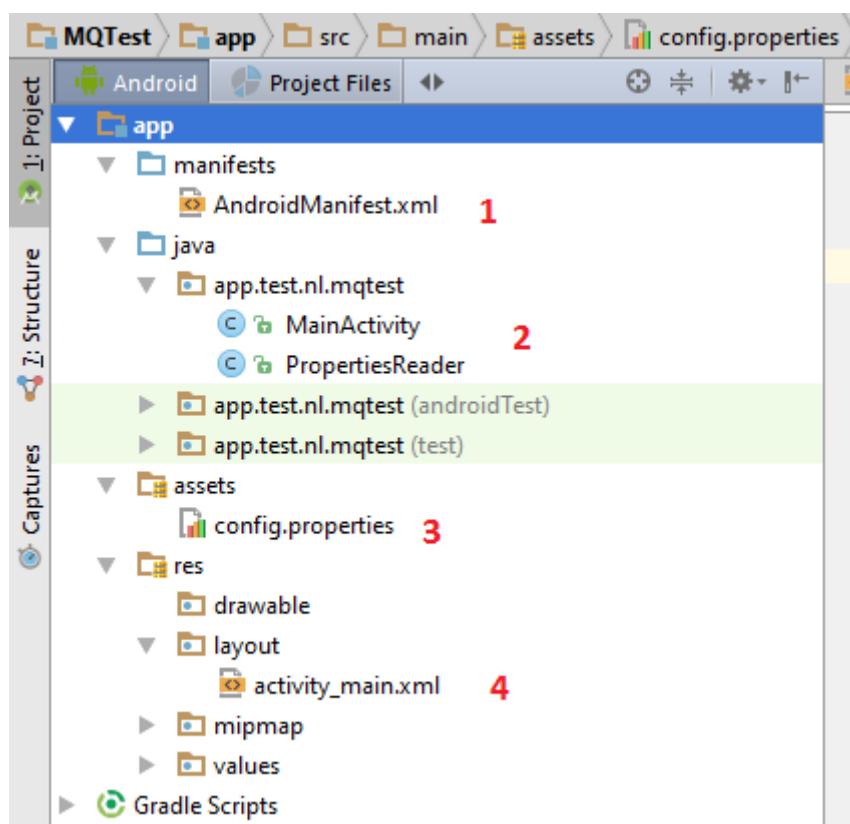
7 Android

7.1 App maken

Start Android Studio en maak een nieuwe App MQTest aan.

De Android App bestaat uit een aantal onderdelen:

- Een manifest bestand (1)
- Een configuratie bestand (3)
- Een scherm beschrijving (4) en
- Het uiteindelijke Android Javaprogramma (2)



Part 1: The 35 euro IoT project

7.1.1 AndroidManifest.xml

In het manifest worden o.a. opgenomen:

- De Android API-versies die door de APP worden ondersteund (in dit voorbeeld minimaal API 21 en maximaal API 23)
- De permissies die aan de App worden toegekend (in dit voorbeeld Internet toegang)
- En waar de App moet beginnen als deze wordt gestart (in dit voorbeeld in MainActivity).

```
<?xml version="1.0" encoding="utf-8" ?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="app.test.nl.mqtest">

    <uses-sdk
        android:minSdkVersion="21"
        android:maxSdkVersion="23" />

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name=
                    "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

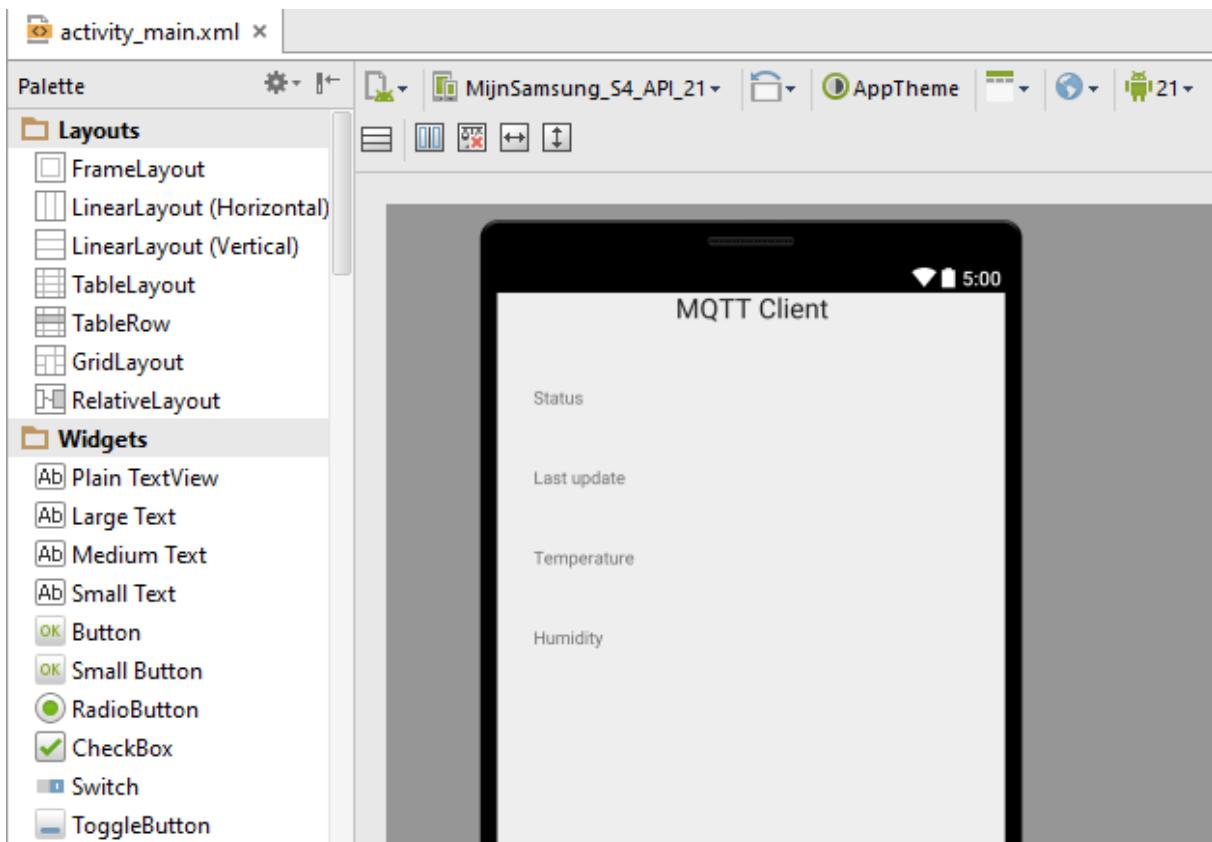
</manifest>
```

Zie: Sources → Android → MQTest → app → src → main → AndroidManifest.xml

Part 1: The 35 euro IoT project

7.1.2 activity_main.xml

Het activity_main.xml bestand bevat de scherm lay-out van de App.



Het scherm kan met drag&drop van lay-outs worden opgebouwd. Ook is het mogelijk dit te doen via de tekst editor.

Zie: Sources → Android → MQTest → app → src → main → res → layout →activity_main.xml

7.1.3 config.properties

In het config.properties bestand worden de volgende configuratie variabelen gezet:

- Het IP-adres van de MQTT-broker (zoals al eerder bepaald).
- Het poortnummer dat door de MQTT-broker wordt gebruikt.
- De topic naam die moet worden afgevangen, in dit voorbeeld de wildcard "#" (staat dus voor alle topics).

```
# Configuration File
ip_address=192.168.0.108
port_number=1883
topic_name=#
```

Zie: Sources → Android → MQTest → app → src → main → assets → config.properties

Part 1: The 35 euro IoT project

7.1.4 Android Java code

7.1.4.1 Inleiding

De Javacode bestaat uit twee delen:

- Een stukje code om de properties uit config.properties te lezen en
- Het uiteindelijke programma dat het scherm opmaakt en de gepubliceerde topics verwerkt.

7.1.4.2 PropertiesReader.java

In deze Javacode wordt het bestand config.properties gelezen en worden de properties voor IP-adres, poortnummer en topic naam gezet. Deze properties worden via een drietal getter functies beschikbaar gesteld.

Zie: Sources → Android → MQTest → app → src → main → java → app → test → nl → mqtest → PropertiesReader.java

```
package app.test.nl.mqtest;

import android.content.Context;
import android.content.res.AssetManager;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

// Properties reader from assets/config.properties
public class PropertiesReader {
    private Context context;
    private Properties properties;

    private String ip_address;
    private String port;
    private String topic;

    public PropertiesReader(Context context) {
        this.context = context;

        // Creates a new object 'Properties'
        properties = new Properties();
    }

    public void readPropertiesFromFile(Context context) {
        // Reads the configuration file
        PropertiesReader propertiesReader = new
            PropertiesReader(context);
        properties=propertiesReader.getProperties("config.properties");

        // Get parameter values
        ip_address =properties.getProperty("ip_address");
        port = properties.getProperty("port_number");
        topic = properties.getProperty("topic_name");
    }
}
```

Part 1: The 35 euro IoT project

```
// Getters
public String getIp_address() {
    return ip_address;
}

public String getPort() {
    return port;
}

public String getTopic() {
    return topic;
}

// Get properties from file assets/<fileName>
private Properties getProperties(String fileName) {
    try {
        // Access to the folder 'assets'
        AssetManager amgr = context.getAssets();
        // Opening the file
        InputStream inputStream = amgr.open(fileName);
        // Loading of the properties
        properties.load(inputStream);
    }
    catch (IOException e) {
        Log.e("PropertiesReader",e.toString());
    }
    return properties;
}
}
```

Part 1: The 35 euro IoT project

7.1.4.3 MainActivity.java

MainActivity.java is het uiteindelijke startpunt van de applicatie en bestaat uit:

1. Een aantal benodigde import statement met o.a. de Paho MQTT-implementatie en JSON.
2. MainActivity is het zogenaamde startpunt van de App.
3. On het OnCreate event wordt het scherm voor de App opgebouwd.
4. De properties voor IP-adres, poortnummer en topic name worden bepaald.
5. De MQTT URI wordt opgemaakt in de vorm `tcp://<ip_adres>:<poortnummer>`, bijvoorbeeld:
<tcp://192.168.0.108:1883>
6. Er wordt vervolgens geprobeerd om connectie met de MQTT-broker te maken.
7. Indien er connectie is wordt er geabonneerd (subscribe) op het eerder bepaalde topic.
8. Om de App niet te blokkeren wordt een nieuw thread gestart waarin op het publiceren van berichten wordt gewacht.
9. Elke 2 seconden wordt er een scherm refresh gedaan.
10. Voor een scherm refresh is het nodig dat er een thread op de gebruikersinterface wordt gezet (`runOnUiThread`).
11. Via de functie `watch` wordt gekeken of er een bericht is gepubliceerd.
12. Indien er een bericht is gepubliceerd wordt het scherm geüpdateet met de nieuwe waarden.
13. Hier wordt de thread (zie 8) gestart.
14. In de functie `watch` wordt een zogenaamde callback functie op de MQTT-clieënt gezet. Dit om te kunnen bepalen of er een bericht is gepubliceerd.
15. Als er een bericht wordt gepubliceerd wordt er een `MessageArrived` event getriggerd.
16. Het binnengekomen bericht wordt verwerkt.
17. In deze functie worden de schermvelden voor o.a. datum, temperatuur en luchtvochtigheid gezet.
18. In deze functies wordt een (fout)boodschap gezet.
19. In deze functie wordt de datum omgezet van:
YYYYMMDDHHmmSS naar DD-MM-YYYY HH:mm:SS

Zie: Sources → Android → MQTest → app → src → main → java → app → test → nl → mqtest → MainActivity.java

```
package app.test.nl.mqtest;

// 1
import android.app.Activity;
import android.content.Context;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
import org.json.JSONObject;
```

Part 1: The 35 euro IoT project

```
// 2
// MainActivity: show data from dht22 sensor using MQTT
public class MainActivity extends Activity {

    private MqttClient client;
    private boolean isMessageArrived;
    private String lastUpdate;
    private String temperature;
    private String humidity;
    private TextView txtLastUpdate;
    private TextView txtTemperature;
    private TextView txtHumidity;
    private TextView txtStatus;
    private TextView txtMessage;
    private MemoryPersistence persistence;

    // 3
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Set TextViews
        txtLastUpdate = (TextView) findViewById(R.id.datetime);
        txtTemperature = (TextView) findViewById(R.id.temperature);
        txtHumidity = (TextView) findViewById(R.id.humidity);
        txtStatus = (TextView) findViewById(R.id.status);
        txtMessage = (TextView) findViewById(R.id.message);

        // 4
        // Get properties
        Context context = this;
        PropertiesReader p = new PropertiesReader(context);
        p.readPropertiesFromFile(context);

        // 5
        // Assemble MQTT URI string
        String mqttURI = "tcp://" + p.getIp_address() + ":" +
            p.getPort();

        // Init
        isMessageArrived = false;

        try {
            // 6
            // Try to connect MQTT Client to broker
            persistence = new MemoryPersistence();
            client = new MqttClient(mqttURI, "", persistence);
            client.connect();

            // 7
            // Subscribe to topic
            client.subscribe(p.getTopic());
            Log.i("MQTTClient", "Client Connected: topic: " +
                p.getTopic());
            setText(new String[]{"Connected", "...", "...", "..."});
            setMessageText("Waiting...");
        }
    }
}
```

Part 1: The 35 euro IoT project

```
// 8
// Start new thread to regulary update view
Thread t = new Thread() {

    @Override
    public void run() {
        try {
            while (!isInterrupted()) {

                // 9
                // Wait for 2 seconds before doing a new
                // refresh
                Thread.sleep(2000);

                // 10
                // Run on UI thread to be able doing the
                // refresh
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        // 11
                        // Watch for new message
                        Log.i("MQTTClient", "Watch...");
                        watch();

                        // 12
                        // Update view
                        if(isMessageArrived) {
                            setText(new String[]{"Connected",
                                lastUpdate, temperature,
                                humidity});
                            setMessageText(
                                "Message Received!");
                            isMessageArrived = false;
                        } else {
                            setMessageText("Waiting...");
                        }
                    }
                });
            }
        } catch (InterruptedException e) {
            setMessageText("Waiting...");
        }
    }

    // 13
    // Start thread
    t.start();
}

} catch (MqttException e) {
    setMessageText("MQTT Exception: " + e.getMessage());
    e.printStackTrace();
    Log.i("MQTTClient", "Exception: " + e.getMessage());
}
}
```

Part 1: The 35 euro IoT project

```
// 14
private void watch() {
    client.setCallback(new MqttCallback() {
        @Override
        public void connectionLost(Throwable throwable) {
        }

        @Override
        // 15
        public void messageArrived(String arg0, MqttMessage arg1)
            throws Exception {
        // A message arrived on the topic
        Log.i("MQTTClient", "Message Received");
        Log.i("MQTTClient", "Topic : " + arg0);
        Log.i("MQTTClient", "Message: " + arg1.toString());

        try {
            // 16
            // Parse JSON object
            JSONObject mainObj = new JSONObject(arg1.toString());

            lastUpdate = convertDate(
                mainObj.getJSONObject("dht22").
                    getString("datetime") );
            temperature = mainObj.getJSONObject("dht22").
                getString("temperature") + "°C";
            humidity = mainObj.getJSONObject("dht22").
                getString("humidity") + "%";

            Log.i("MQTTClient", "JSON string is parsed");

            isMessageArrived = true;
        } catch (Exception e) {
            setMessageText("JSON Exception!");
            e.printStackTrace();
            Log.i("MQTTClient", "Exception: " + e.getMessage());
        }
    }

    @Override
    public void deliveryComplete(
        IMqttDeliveryToken iMqttDeliveryToken) {
    }
});;

};

// 17
private void setText(String[] args) {
    // Set 4 TextViews (4 input string are expected)
    if (args.length != 4) {
        return;
    }
    txtStatus.setText(args[0]);
    txtLastUpdate.setText(args[1]);
    txtTemperature.setText(args[2]);
    txtHumidity.setText(args[3]);
}
```

Part 1: The 35 euro IoT project

```
// 18
private void setMessageText(String arg) {
    // Set message
    setMessageText(arg, false);
}

private void setMessageText(String arg, boolean isError) {
    // Set (error) message
    if (isError) {
        txtMessage.setTextColor(Color.RED);
    } else {
        txtMessage.setTextColor(Color.BLUE);
    }
    txtMessage.setText(arg);
}

// 19
private String convertDate(String inputDate) {
    // yyyyMMddhhmmss
    return inputDate.substring(6, 8) + "-" +
        inputDate.substring(4, 6) + "-" +
        inputDate.substring(0, 4) + " " +
        inputDate.substring(8, 10) + ":" +
        inputDate.substring(10, 12) + ":" +
        inputDate.substring(12);
}
```

7.2 App testen

7.2.1 Voorbereiding

Voor het tonen van de grafieken dienen de volgende servers te worden gestart:

- MongoDB
- NodeJS Express server (mijnserver.js)

Start de MongoDB server met de commando's:

```
cd \
cd Program Files\MongoDB\Server\3.2\bin
mongod
```

Start de NodeJS Express server met de commando's:

```
cd \
cd Program Files\nodejs
node mijnserver2.js
```

Start voor het verkrijgen van data:

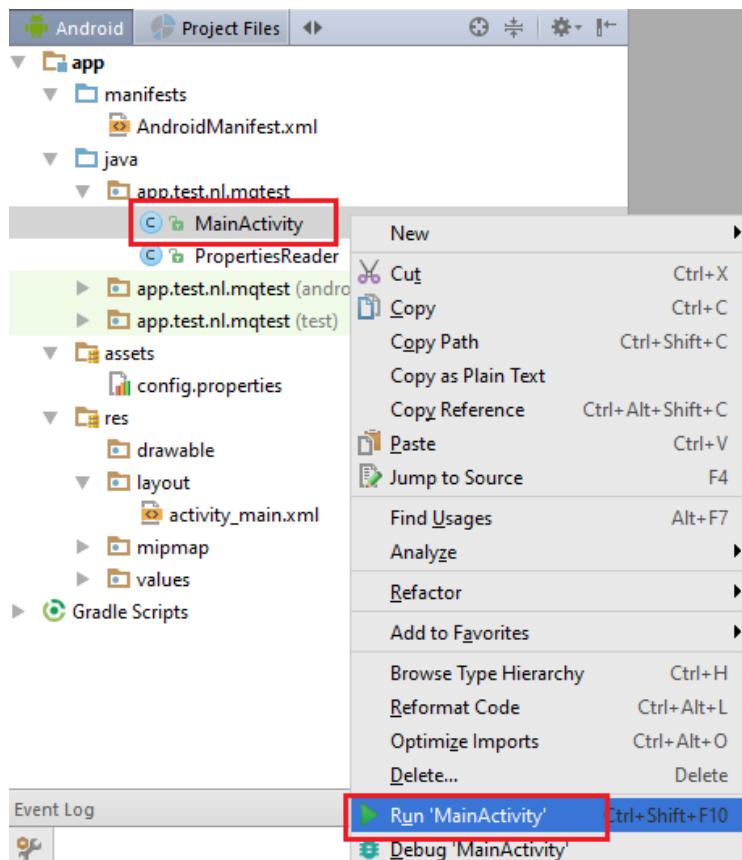
- De ESP12E oplossing (zie hoofdstuk 6) of
- Via de PAHO Java Client (zie hoofdstuk 4).

N.B: Check de IP-adressen die in de diverse componenten zijn gezet.

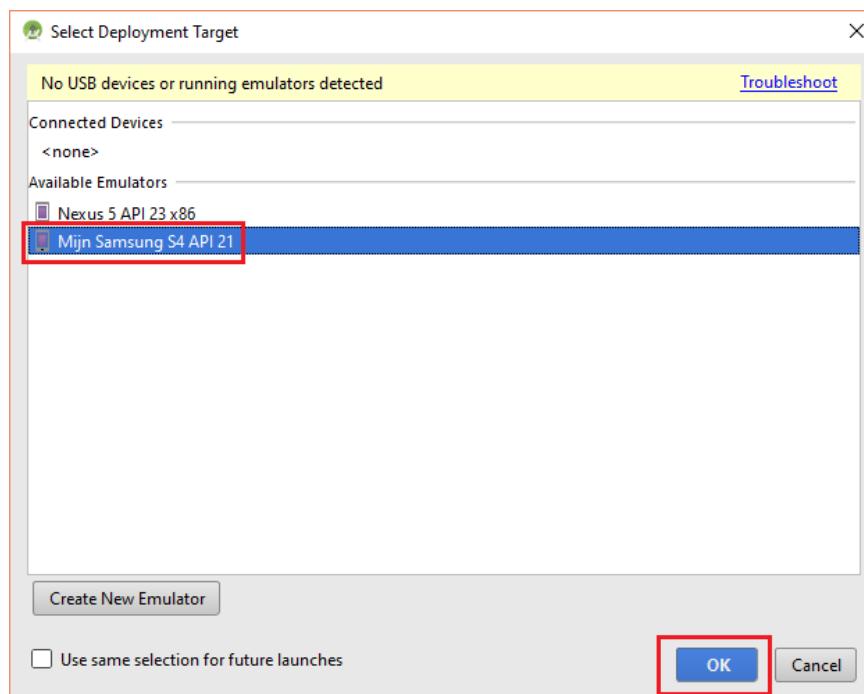
Part 1: The 35 euro IoT project

7.2.2 Android App

Ga naar Android Studio → App → Java → MainActivity → Run 'MainActivity'

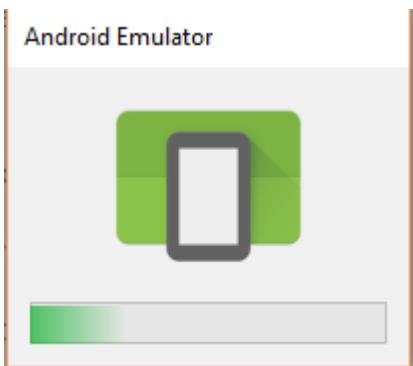


Selecteer een Emulator (in dit voorbeeld 'Mijn Samsung S4 API 21') → OK

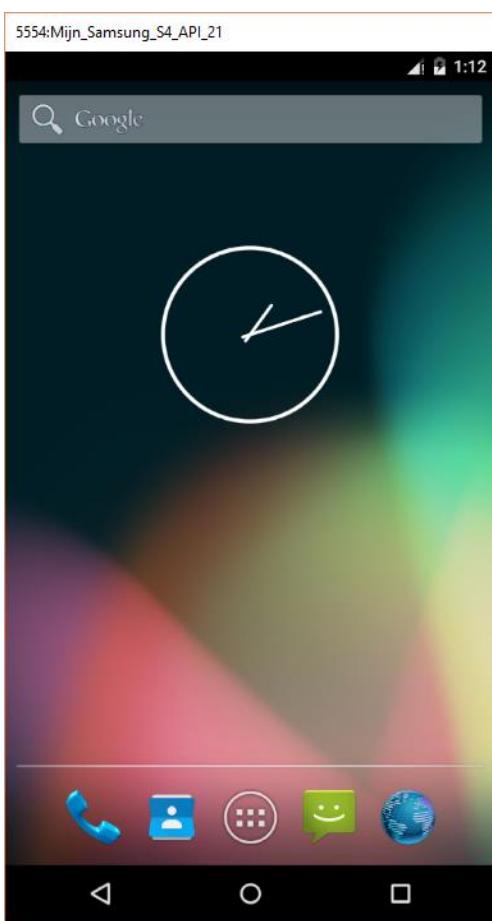


Part 1: The 35 euro IoT project

Even geduld...

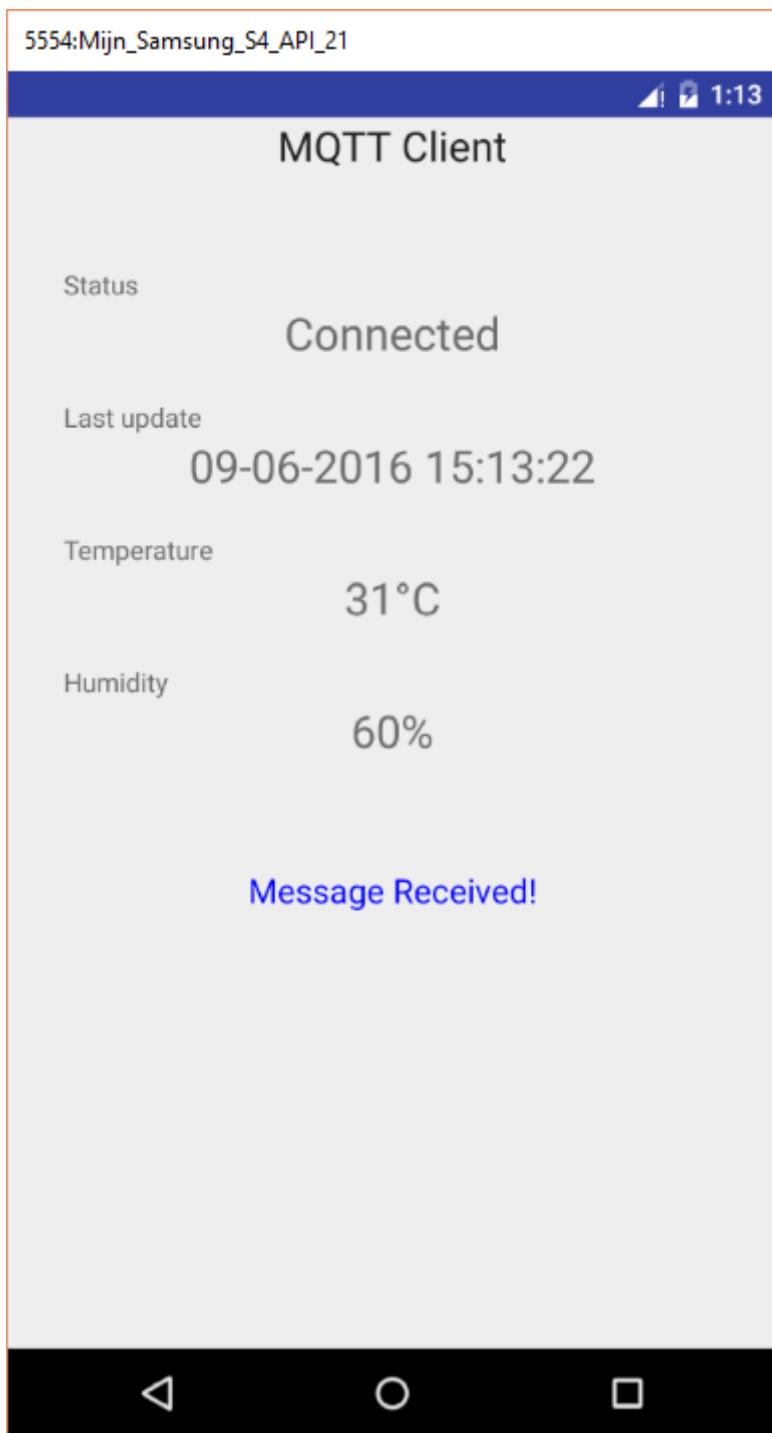


Nog wat meer geduld... ;-)



Part 1: The 35 euro IoT project

Resultaat:



Part 1: The 35 euro IoT project

7.2.3 Smartphone test

Sluit de smartphone via een USB-kabel aan op de pc, laptop of tablet. Zorg ervoor dat de juiste USB-driver is geïnstalleerd. Elke mobiele telefoonmerk heeft een eigen USB-driver. In dit boek wordt de Samsung USB-driver gebruikt.

7.2.3.1 Samsung USB-driver

Voor Samsung ga naar:

<http://developer.samsung.com/technical-doc/view.do?v=T000000117>

Download ZIP-bestand

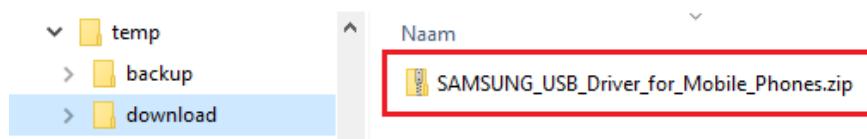
Samsung Android USB Driver for Windows

[Android] Feb 10, 2015

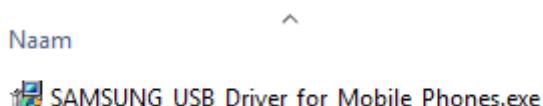
↳  SAMSUNG_USB_Driver_for_Mobile_Phones.zip (15.3MB)

The USB Driver for Windows is available for download in this page. You need the driver only if you are developing on Windows and want to connect a Samsung android device to your development environment over USB.

Pak het ZIP-bestand uit.

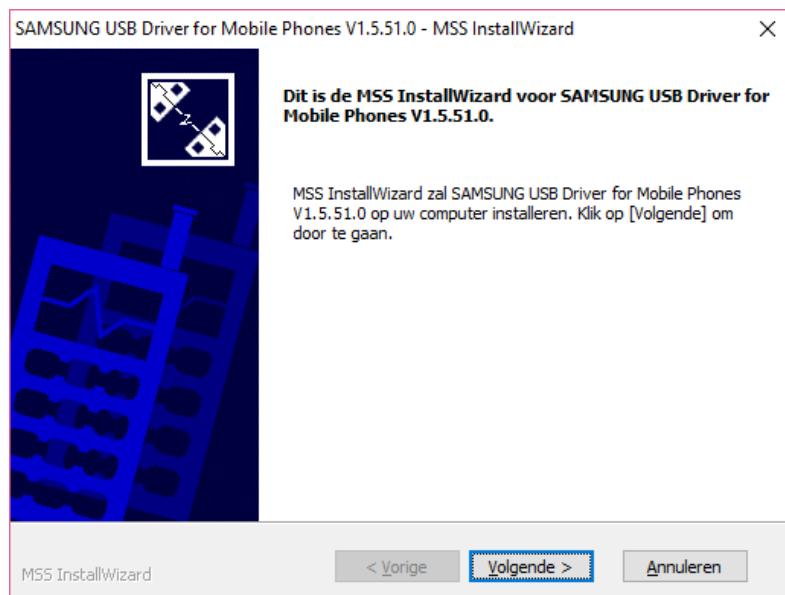


Start installatie

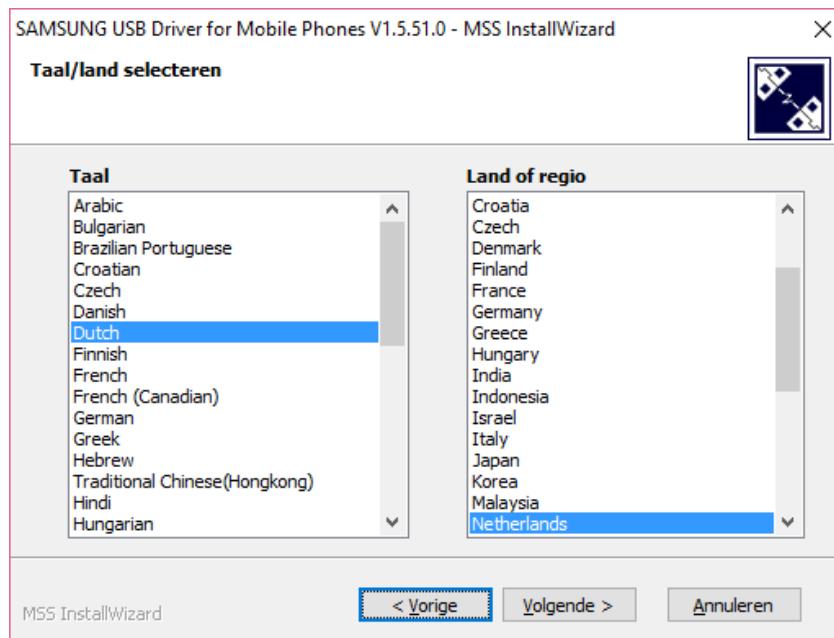


Part 1: The 35 euro IoT project

→ Volgende

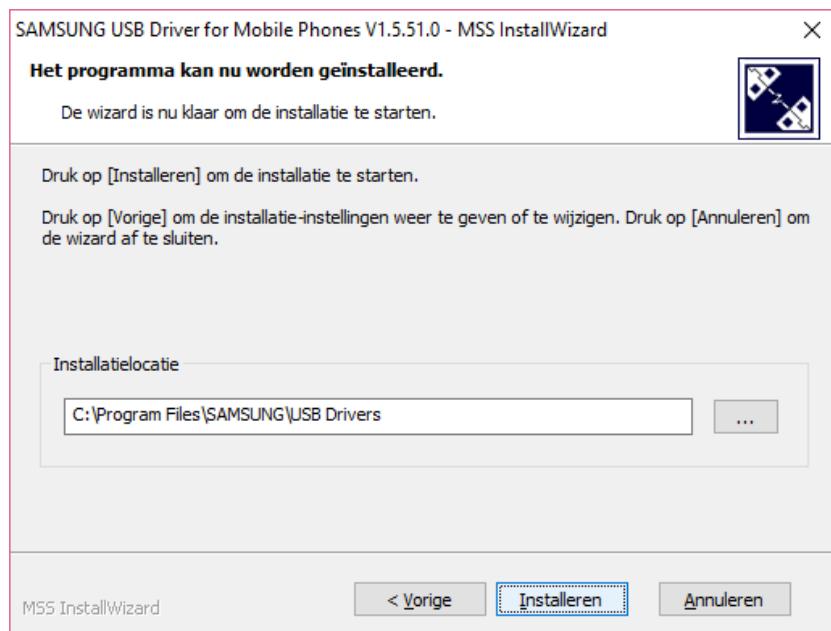


→ Nederland → Volgende

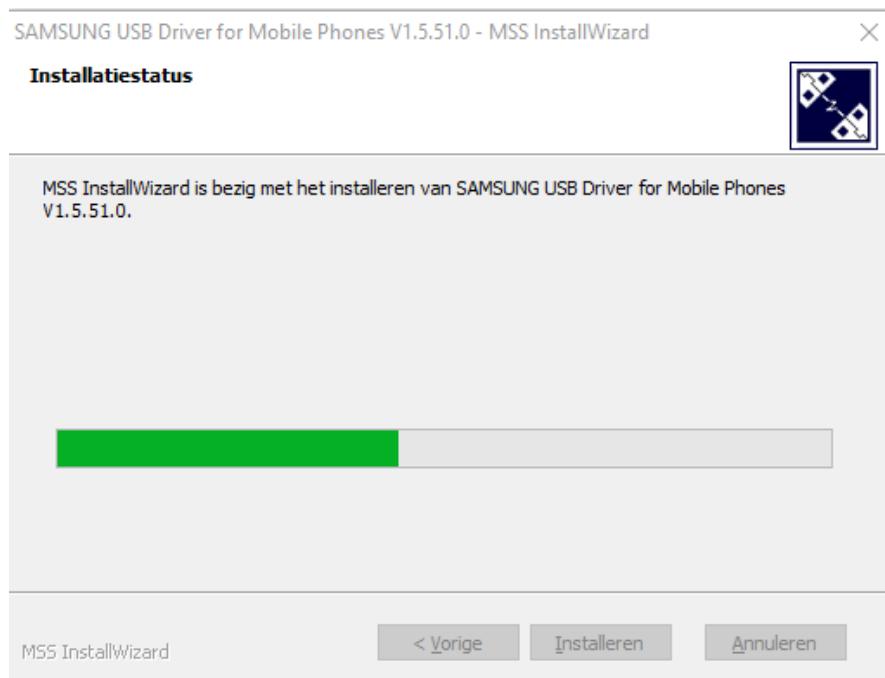


Part 1: The 35 euro IoT project

→ Installeren

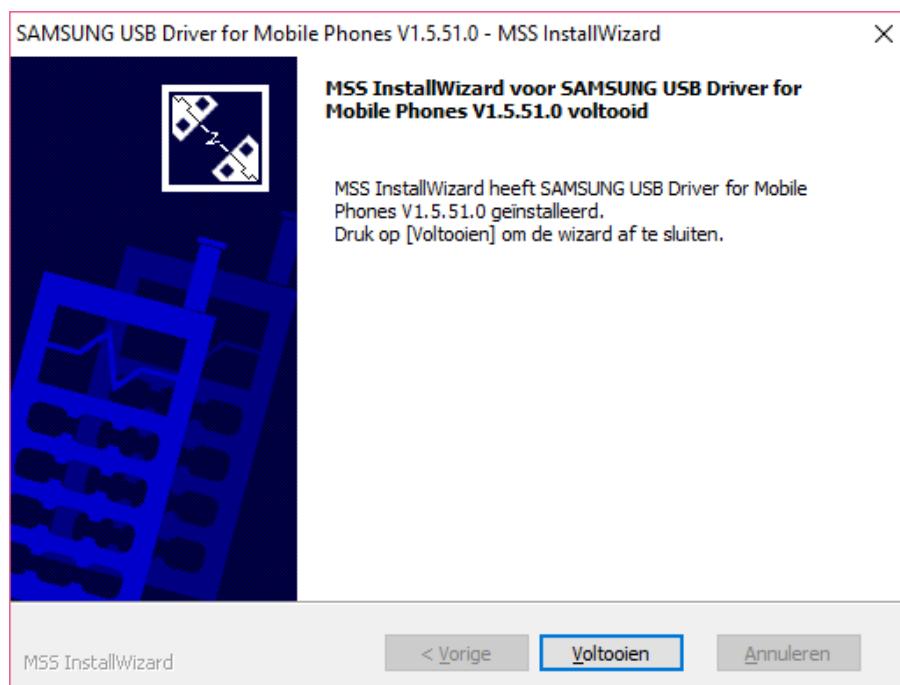


Even geduld...



Part 1: The 35 euro IoT project

→ Voltooien



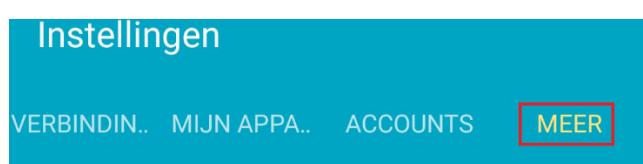
7.2.3.2 Ontwikkelaarsopties inschakelen

Om met een smartphone te kunnen testen dienen de ontwikkelaarsopties te worden ingeschakeld. Vanaf Android 4 is dat niet langer zo eenvoudig.

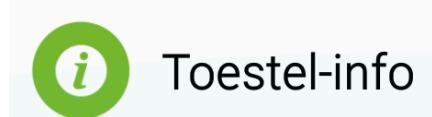
Klik op Instellingen



Tab: Meer

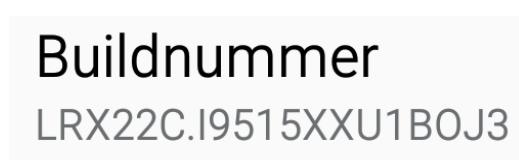


Toestel-info



Part 1: The 35 euro IoT project

Klik 10x op Buildnummer



De ontwikkelaarsopties worden ingeschakeld (indien nodig inschakelen via de groene knop).



7.2.3.3 MQTest App testen

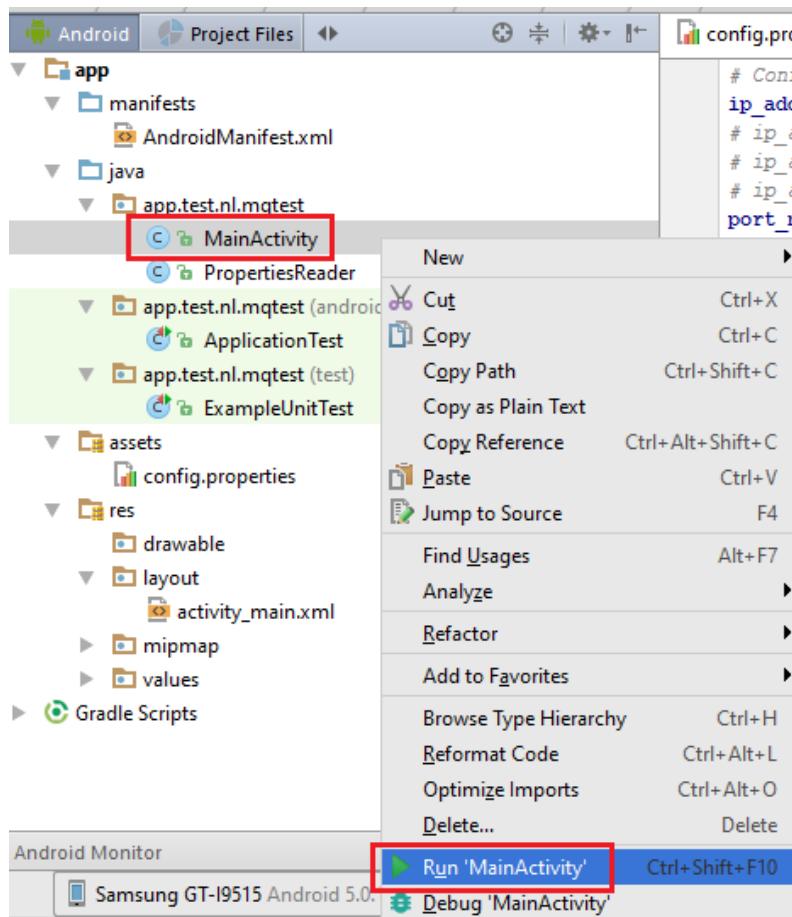
Start Android Studio en laad met project MQTest en zet (indien nodig) het juiste Wi-Fi IP-adres in config.properties → Save

```
# Configuration File
ip_address=192.168.0.108
# ip_address=192.168.43.100
# ip_address=192.168.137.1
# ip_address=127.0.0.1
port_number=1883
topic_name=mosquitto_demo/test
topic_name=
```

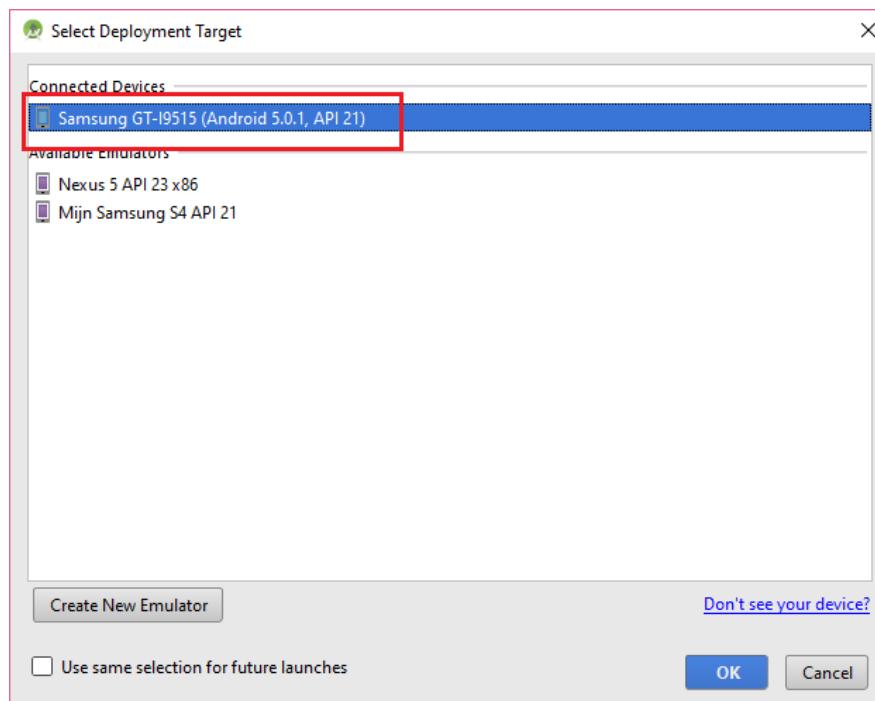
Sluit de smartphone aan op de USB-aansluiting van de pc, laptop of tablet.

Part 1: The 35 euro IoT project

Selecteer MainActivity met de rechtermuisknop → Run 'MainActivity'



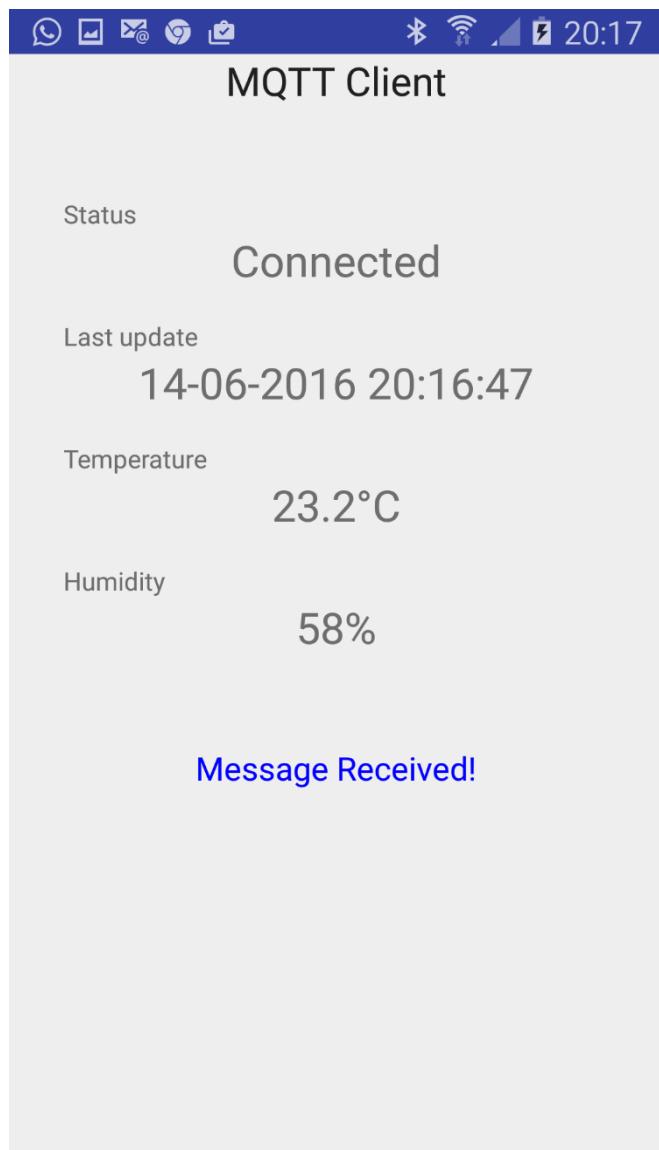
Selecteer de aangesloten smartphone → OK



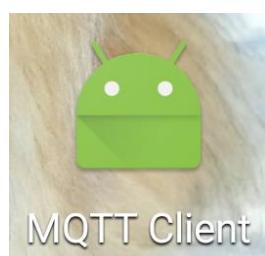
Part 1: The 35 euro IoT project

Even geduld...

Als het goed is wordt de App nu gestart op de smartphone.



Sluit Android Studio en koppel de USB-kabel los. Als het goed is blijft de App gewoon doorwerken en is de geïnstalleerd op de smartphone.



8 The Cloud

8.1 Inleiding

IoT zonder de Cloud is geen IoT wordt weleens gesteld. Ik denk dat daarover gediscussieerd kan worden. In hoeverre is een externe Cloud provider betrouwbaar genoeg? Wat gebeurd er met dat data die wordt opgeslagen? Wat gebeurt er als een Cloud provider failliet gaat? En wat kost het? En natuurlijk zijn er ook “gratis” Cloud providers. Maar “gratis” komt vaak wel met een prijs of een beperking in bijvoorbeeld de hoeveelheid data die men mag opslaan.

Alternatief is een private Cloud. Nadeel is dat men die dan wel zelf moet inrichten inclusief de benodigde hardware.

In dit boek wordt de Cloud gesimuleerd door MongoDB als netwerkdatabase te gebruiken.

8.2 Grafieken

De temperatuur en luchtvochtigheidsdata die wordt opgeslagen kan prima worden gebruikt voor het tonen van grafieken. Voor het tonen van de grafieken wordt gebruik gemaakt van Chart.js. De verwerking bestaat uit twee delen:

- Het starten van een NodeJS Express server en het ophalen van de benodigde data uit de MongoDB database.
- Een HTML-pagina met een stukje Java Script voor het ophalen van de data en het opmaken van de grafieken met Chart.js.

8.2.1 NodeJS Express server

Maak in C:\Program Files\nodejs een script mijnserver2.js aan.

1. Eerst worden een aantal variabelen gedeclareerd o.a. voor de Express server en een MongoDB cliënt. Dit script wordt op de localhost uitgevoerd.
2. In de findData functie worden de laatste 30 rijen uit de MongoDB opgehaald. De opgehaalde data wordt gezet in een JSON-string gezet met het volgende formaat:

```
{"dht22":  
  [  
    {"datetime":20160608142840,"temperature":26.5,"humidity":47},  
    {"datetime":20160608142845,"temperature":23.6,"humidity":61},  
    ...  
    {"datetime":20160608143107,"temperature":21.1,"humidity":59}  
  ]  
}
```

Tussen [] staan de herhalende groepen met temperatuur en vochtigheidsdata.

De data wordt in array gepushed volgens het Last In – First Out (LIFO) principe. Dit betekent bijvoorbeeld dat in de data array eerst de datetime, daarna de temperatuur en tenslotte de luchtvochtigheid wordt gezet (push). Als deze data uit de array wordt gehaald (pop) dan gebeurd dat in de volgorde luchtvochtigheid, temperatuur en tenslotte datetime.

Part 1: The 35 euro IoT project

Voor de row array betekent het dat de jongste entry het eerst in de array wordt gepushed en de oudste entry als laatste. Bij het uitpakken betekent dit automatisch dat de oudste entry als eerste uit de array wordt gehaald (pop) en jongste entry als laatste. Wat goed uitkomt bij het opmaken van de grafieken.

3. In de setJSON functie wordt het te versturen JSON-object aangemaakt.
4. Hier wordt de NodeJS Express server gestart. De URL die hieraan is gekoppeld is:
<http://localhost:5000/db> of <http://127.0.0.1/db>

Op basis van dit request wordt het eerder opgemaakte JSON-object als string naar de aanroepende partij gestuurd.

5. Hier wordt connectie met de MongoDB database gemaakt.

mijnserver2.js

```
// 1
var express = require('express'),
serveStatic = require('serve-static');
bodyParser = require('body-parser');
mongoClient = require('mongodb').MongoClient;
mongoDb = "";
mongoDbUrl = "";
row = [];
data = [];
dht22String = "";
ipAddress = '127.0.0.1';
nodeSettings = {
    port : 5000
};
mongoDbSettings = {
    host : ipAddress,
    port : 27017
};

// 2
var findData = function(db, callback) {
    var index = 0;
    var cursor = db.collection('dht22temphum').find().limit(30).sort({_id:-1});
    row = [];
    data = [];
    cursor.each(function(err, doc) {
        if (doc != null) {
            if (doc.dht22.datetime) {
                data.push( doc.dht22.datetime );
                console.log( index + " - " + data[0] );
                ++index;

                var tm = doc.dht22.temperature;
                if(tm.length == 1) {
                    tm = "0" + tm + ".0";
                } else if(tm.length == 2) {
                    tm = tm + ".0";
                }
                data.push( tm );
                data.push( doc.dht22.humidity );
                row.push( data );
                data = [];
            } else {
                callback();
            }
        }
    });
};
```

Part 1: The 35 euro IoT project

```
// 3
var setJSON = function() {
    var index = 0;
    var dt, tm, hm;
    console.log( "---" );
    dht22String = "{\"dht22\":[" ;
    data = row.pop();
    while(data != null) {
        if(index == 0) {
            dht22String += "{ ";
        } else {
            dht22String += ",{ ";
        }
        hm = data.pop();
        tm = data.pop();
        dt = data.pop();
        console.log( index + " - " + dt );
        ++index;
        dht22String += "\"datetime\":" + dt;
        dht22String += ",\"temperature\":" + tm;
        dht22String += ",\"humidity\":" + hm;
        dht22String += "}";
        data = row.pop();
    }
    dht22String += "]}";
    console.log( dht22String );
}

// 4
// Here we start NodeJS
app = express();

app.use( serveStatic('..../Web') );
app.use( bodyParser.json() );

app.get('/db',function(req, res, next) {
    findData(mongoDb, function() {
        setJSON();
    });
    res.write( dht22String );
    res.send(JSON.stringify(res.body));
});

app.listen(nodeSettings.port);
console.log('Node server is up and running')

// 5
// Here we start mongoDb
mongoDbUrl = 'mongodb://' + mongoDbSettings.host + ':' + mongoDbSettings.port +
'/dht22iot';

mongoClient.connect(mongoDbUrl, function(err, db) {
    console.log("MongoDB connected");
    mongoDb = db;
});
```

Zie: Sources → NodeJS → mijnserver2.js

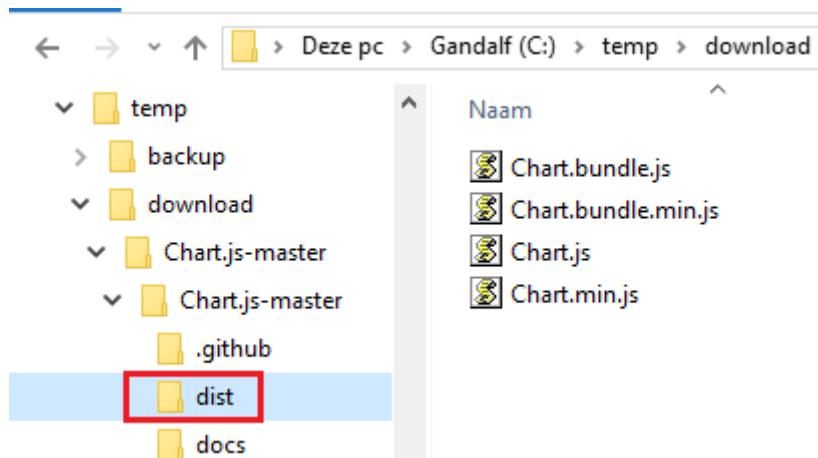
Part 1: The 35 euro IoT project

8.2.2 Index.html

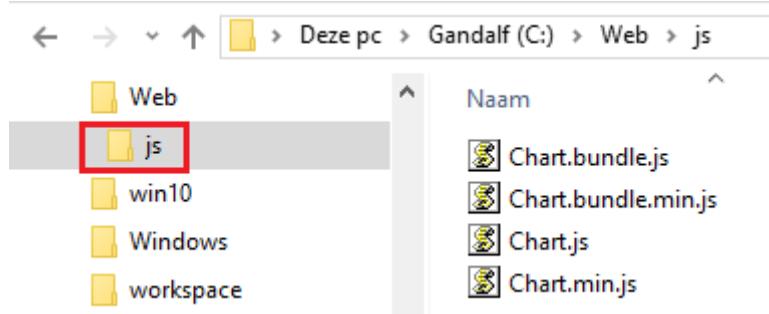
Maak op C: de volgende directory structuur aan:

```
C:  
+--- Web  
    +--- js
```

Kopieer de Chart.js javascripts uit de dist directory:



naar C:\Web\js



Part 1: The 35 euro IoT project

Maak in C:\Web een bestand index.html aan.

1. De trigger voor het ophalen van de data is een button. Door te klikken op deze button wordt de functie getData() (zie 6) uitgevoerd.
2. Voor het tonen van de grafieken wordt een canvas (myChart) gebruikt.
3. Het script Chart.min.js is nodig voor het opmaken en tonen van de grafieken.
4. De benodigde globale variabelen voor o.a. de context (myChart canvas), het HTTP Request en de benodigde data arrays (labels, temperatuur en vochtigheid).
5. In de functie createChart wordt de grafiek opgemaakt en getoond.
6. In de functie processRequest wordt de opgehaalde data verwerkt. Door het eerdergenoemde FILO-principe staat de te tonen data al in de juiste volgorde (van oud naar nieuw).
7. Op basis van het click event (zie 1) wordt de functie getData() uitgevoerd d.m.v. een HTTP request naar <http://localhost:5000/db>. Aan dit request wordt een event readytatechange gekoppeld dat wordt uitgevoerd op het moment dat het request klaar is. De functie processRequest (zie 6) wordt dan uitgevoerd.
8. Op basis van het resultaat van het request wordt een melding "Success!" of "Try again!" getoond.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>HTTP GET</title>
</head>
<body>
    <!-- 1 -->
    <button onclick="getData()">Get</button>
    <!-- 2 -->
    <canvas id="myChart" width="600" height="200"></canvas>
    <!-- 3 -->
    <script src="js/Chart.min.js"></script>

    <!-- 4 -->
    <script>
        var ctx = document.getElementById("myChart").getContext("2d");
        var xhr = new XMLHttpRequest();
        var datelabels = [];
        var tempdata = [];
        var humdata = [];
        var msg = '';
```

Part 1: The 35 euro IoT project

```
<!-- 5 -->
function createChart() {
    var mydata = {
        labels: datelabels,
        datasets: [
            {
                label: "Temperature",
                backgroundColor: [
                    "rgba(102,255,255,0.5)"
                ],
                data: tempdata
            },
            {
                label: "Humidity",
                backgroundColor: [
                    "rgba(102,255,51,0.5)"
                ],
                data: humdata
            }
        ]
    }
    var myNewChart = new Chart(ctx, {
        type: "line",
        data: mydata,
    });
}

<!-- 6 -->
function processRequest(e) {
    if(xhr.readyState == 4 && xhr.status == 200) {
        try {
            var jsonresp = JSON.parse(xhr.responseText);
            datelabels = [];
            tempdata = [];
            humdata = [];
            for(var i=0; i<jsonresp.dht22.length; i++) {
                console.log('index = ' + i + " " +
                    JSON.stringify(jsonresp.dht22[i]));
                datelabels.push(
                    jsonresp.dht22[i].datetime);
                console.log(
                    jsonresp.dht22[i].datetime);
                tempdata.push(
                    jsonresp.dht22[i].temperature);
                console.log(
                    jsonresp.dht22[i].temperature);
                humdata.push(
                    jsonresp.dht22[i].humidity);
                console.log(
                    jsonresp.dht22[i].humidity);
                createChart();
            }
            document.getElementById("message").innerHTML =
                "Success!";
        } catch(err) {
            console.log(err.message);
            document.getElementById("message").innerHTML =
                "Try again!";
        }
    }
}
```

Part 1: The 35 euro IoT project

```
<!-- 7 -->
function getData() {
    xhr.open('GET', "http://localhost:5000/db", true);
    xhr.send();
    xhr.addEventListener("readystatechange", processRequest, false);
    xhr.onreadystatechange = processRequest;
}
</script>
<!-- 8 -->
<h3 id="message"></h3>
</body>
</html>
```

Zie: Sources → Web → index.html

8.2.3 Test

Voor het tonen van de grafieken dienen de volgende servers te worden gestart:

- MongoDB
- NodeJS Express server (mijnserver2.js)

Start de MongoDB server met de commando's:

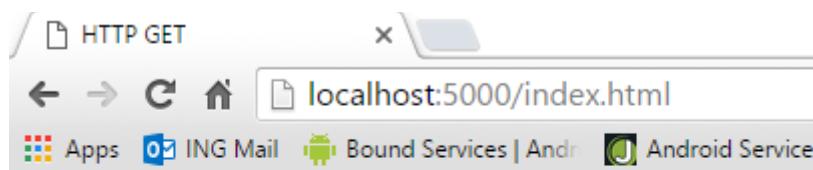
```
cd \
cd Program Files\MongoDB\Server\3.2\bin
mongod
```

Start de NodeJS Express server met de commando's:

```
cd \
cd Program Files\nodejs
node mijnserver2.js
```

Open een browser en ga naar:

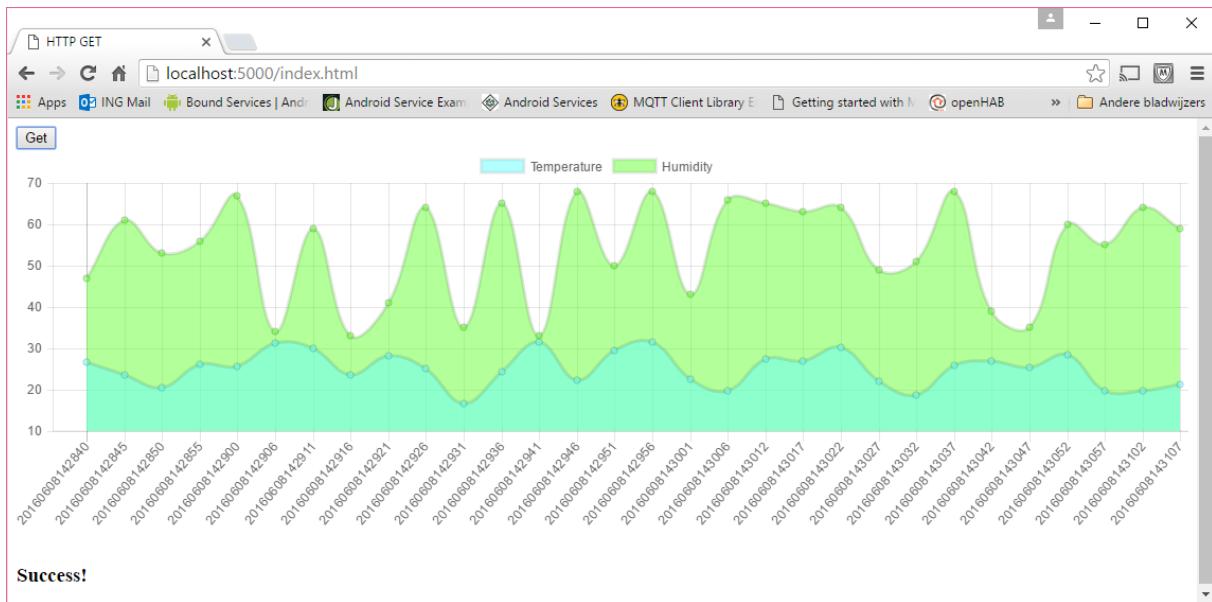
<http://localhost:5000/index.html>



➔ Get (indien de melding "Try Again!" verschijnt het nogmaals proberen).

Part 1: The 35 euro IoT project

Resultaat:



N.B:

Voor het testen van de grafieken is random data gebruikt dat gegeneerd is met de eerder gemaakte Java PAHO-client.

9 Index

- Acknowledged Service, 17
- activity_main.xml, 120, 170
- Additionele Bordenbeheerder, 37
- Afkortingen, 11
- Android, 2, 8, 9, 12, 13, 21, 32, 87, 94, 96, 102, 103, 104, 108, 109, 110, 112, 119, 120, 168, 169, 171, 178
- Android Studio, 32, 96, 102, 103, 104, 108
- AndroidManifest.xml, 119, 169
- AngularJS, 2, 8, 14, 32, 53, 54, 55, 56, 57, 147
- Arduino, 2, 8, 21, 28, 32, 33, 36, 40, 151, 161, 166
- Arduino Bibliotheken, 38
- Arduino IDE, 8, 32, 33
- Arduino loop, 152
- Arduino setup, 152
- Assured Service, 17, 18
- At Least Once, 17
- At Most Once, 17
- authentication, 16
- AVD Manager, 112, 116
- Bi-directional Logic Level Converter, 27
- Blink, 39, 42
- breadboard, 10
- Breadboard, 24, 29, 138
- breadboard voeding, 29
- broker, 8, 15, 16, 17, 18, 20, 21, 32, 63, 78, 87, 94, 145, 165, 174
- Broker, 77
- BSON, 11, 23
- Chart.js, 32, 60, 188, 191
- China, 2, 24, 25, 26, 27, 28, 29, 30, 31
- Chineese websites, 30
- client-sided framework, 14
- Cloud, 12, 23, 188
- communication, 16
- config.properties, 170, 171
- connection, 16
- controller.js**, 58, 59, 147, 148, 150
- DHT11, 26
- DHT22, 8, 24, 26, 140
- DISCONNECT, 20
- DS1307, 8, 24, 26, 27, 141
- DS3231, 26
- Eclipse, 32, 79, 81, 86, 88, 95
- ESP-12E, 2, 8, 9, 21, 24, 25, 32, 39, 40, 41, 43, 44, 139, 140, 141, 151, 164, 165, 166
- ESP8266, 2, 8, 9, 25, 42, 63
- Exactly Once, 18
- Fire and Forget, 17
- gauge, 56, 57
- Gauge, 32, 56, 57
- Hardware, 24, 112, 138, 142
- iDeal, 31
- index.html**, 58, 59, 102, 147, 167, 192, 194
- Index.html, 191
- inklaringskosten, 30, 31
- inleidend deel, 10
- invoerrechten, 30, 31
- IOT, 9
- IOT_Project.ino**, 152
- ipconfig, 137, 164, 165
- Java Hello World, 83
- Java JDK, 96
- Java MQTT cliënt, 87, 88
- JSON, 11, 23, 94, 127, 145, 146, 152, 173, 176, 188, 189, 190, 193
- Jumper cables, 29
- Jumper Cables, 24
- led, 2, 43, 44
- level shifter, 26, 27
- Level Shifter, 24, 27
- libeay32.dll, 73
- LIFO, 11, 188
- LUA loader, 8
- M2M, 15
- MainActivity.java, 119, 120, 173
- Message Queuing Telemetry Transport, 15
- mijnNodeServer.js, 53, 59
- mijnserver.js**, 53, 54, 59, 144, 165, 177
- Mobiele Hotspot, 164
- MongoDB, 8, 23, 121, 125, 126, 127, 129, 130, 144, 145, 177, 188, 189, 190, 194
- Mosca, 21, 32, 63, 144, 145
- Mosquitto, 21, 32, 63, 66, 69, 73, 87, 144, 165
- MQTT, 8, 11, 15, 16, 17, 18, 20, 32, 151, 166
- MQTT broker, 152
- MQTT client, 151
- MQTT-broker, 8, 21, 94, 144, 164, 165, 166, 167, 170, 173
- Nederlandse electronica websites, 31
- Network Time Protocol, 11, 22
- nl.pool.ntp.org, 22
- NodeJS, 21, 32, 45, 53, 59, 63, 121, 125, 128, 144, 145, 165, 177, 188, 189, 190, 194
- NodeMCU, 9, 25

Part 1: The 35 euro IoT project

NosQL, 23
npm install, 52, 125
NTP, 11, 22, 151
OHA, 11, 12
ontwikkelaarsopties, 184, 185
Open Handset Alliance, 11, 12
Paho, 21, 32, 86, 87, 173
PAHO Java cliënt, 87
payload, 16, 93, 94, 146
Ping, 18
PINGREQ/PINGRESP, 18
pop, 188, 189, 190
power adapter, 28
praktisch deel, 10
PropertiesReader.java, 171
pthreadVC2.dll, 69, 74
Publish, 18, 93
publisher, 15, 17, 18, 20, 21, 32
Publisher, 78
push, 188, 189, 193
QoS, 11, 16, 17
QoS0, 17
QoS1, 17
QoS2, 17, 18
Samsung USB driver, 181
SDK, 11, 12
Single Page Applications, 14
smartphone, 8, 9, 181, 184, 185, 186, 187
Smartphone, 9, 164
Software Development Kit, 11, 12
SPA, 14

SSL/TLS, 11, 16
ssleay32.dll, 73
stratum, 22
Subscribe, 18, 174
SUBSCRIBE/SUBACK, 18
subscriber, 15, 17, 18, 20, 79, 94, 95
Subscriber, 78
TCP/IP, 11, 16
termination, 16
TestClient.java, 92
testdb.js, 128, 130
theoretisch deel, 10
topic string, 19
topic tree, 19
tweedehands onderdelen, 31
Uitsluitingskarakter, 19
Unacknowledged Service, 17
Unsubscribe, 18
UNSUBSCRIBE/UNSUBACK, 18
USB driver, 181
USB kabel, 28
USB micro kabel, 24, 28
Verantwoording, 9
Virtual Router, 32, 131, 165
Voedingadapter, 24
voltage divider, 27
Voorkennis, 10
Weerstanden, 24, 29
WiFi, 151
WiFi thuis, 164
wild card, 19, 20