Internet of Things - Internet  
Beroeps Product - Weerstation (BP1)  
  
  
naam: Jorn Brouwer  
Studentnummer: 515472  
Klas: IoT -N sept  
Docent: Jorg Visch

Voorwoord  
//Leerproces beschrijven aan de hand van de deelopdrachten.   
In de opdracht van "week 1 - Arduino en Internet"   
  
//Transitie van een simpel sensormeting naar een configureerbare webserver.   
--> actieve benadering van webserver vanaf Deelopdracht B.   
//Transitie van webserver naar gateway + cloud storage.  
vanaf 4.4 Gateway (eindopdracht)

Inhoudsopgave

[Inleiding 4](#_Toc464490939)

[Opdracht 1: Parseren request – line 5](#_Toc464490940)

[Opdracht 2: GET en POST op de Arduino 6](#_Toc464490941)

[Opdracht 3: Weerstation als webserver 7](#_Toc464490942)

[Deelopdracht 3A 7](#_Toc464490943)

[3.1.2 Code 7](#_Toc464490944)

[3.2.1 Functionaliteiten 7](#_Toc464490945)

[3.2.3. Communicatieprotocol 7](#_Toc464490946)

[3.2.4 Grenzen van de Arduino 7](#_Toc464490947)

[Opdracht 4: Eindopdracht 8](#_Toc464490948)

[4.1 Opdrachtomschrijving 8](#_Toc464490949)

[4.4 Planning 9](#_Toc464490950)

[5. Conclusie 10](#_Toc464490951)

[6. Bronnenlijst 11](#_Toc464490952)

[7. Bijlagen 12](#_Toc464490953)

# Inleiding

# Opdracht 1: Parseren request – line

HTTP-message = start-line

\*( header-field CRLF )

CRLF

[ message-body ]

A recipient MUST parse an HTTP message as a sequence of octets in an

encoding that is a superset of US-ASCII [[USASCII](https://tools.ietf.org/html/rfc7230#ref-USASCII)]. Parsing an HTTP

message as a stream of Unicode characters, without regard for the

specific encoding, creates security vulnerabilities due to the

varying ways that string processing libraries handle invalid

multibyte character sequences that contain the octet LF (%x0A).

String-based parsers can only be safely used within protocol elements

after the element has been extracted from the message, such as within

a header field-value after message parsing has delineated the

individual fields.

An HTTP message can be either a request from client to server or a

response from server to client. Syntactically, the two types of

message differ only in the start-line, which is either a request-line

(for requests) or a status-line (for responses), and in the algorithm

for determining the length of the message body

start-line = request-line / status-line

A request-line begins with a method token, followed by a single space

(SP), the request-target, another single space (SP), the protocol

version, and ends with CRLF.

request-line = method SP request-target SP HTTP-version CRLF

Recipients typically parse the request-line into its component parts

by splitting on whitespace

Recipients of an invalid request-line SHOULD respond with either a

400 (Bad Request) error or a 301 (Moved Permanently) redirect with

the request-target properly encoded. A recipient SHOULD NOT attempt

to autocorrect and then process the request without a redirect, since

the invalid request-line might be deliberately crafted to bypass

security filters along the request chain.

# Opdracht 2: GET en POST op de Arduino

Opdracht 3: Weerstation als webserver  
//arduino wordt webserver, dwz er moeten HTTP requests gemaakt kunnen worden en deze moeten op //de juiste manier worden afgehandeld. (HTTP-protocol)

Deelopdracht 3A   
3. 1.1 Opstelling   
Fritzing  
Foto

3.1.2 Code   
//Bouw de basisopstelling na  
//Test deze d.m.v. een Arduino sketch, deze moet minimaal:  
-->de ledjes kunnen laten knipperen  
--> Sensoren moeten meting verrichten wanneer er op de knop gedrukt word.  
-->Sensor waarden moeten naar de seriele port worden gestuurd.  
  
Deelopdracht 3B

3.2.1 Functionaliteiten  
Lijst met functionaliteiten  
  
3.2.2. Ontwerp  
Maak een gedegen ontwerp. dmv. state-machines/sequence diagrams, activity diagrams.

3.2.3. Communicatieprotocol  
Let het communicatieprotocol vast.  
--> beschrijf alle requests en responses in tekst.   
(Minimaal)  
- Get/ --> html pagina met alle statussen (HTML)   
- Get /temp --> response temperatuur en, min en max (JSON)   
- Get /lux --> response lux (JSON)   
- Get /data --> alle data (JSON)   
- Put of post? /temp --> instellen min en max (voor de kleurtjes van de leds) (JSON)

3.2.4 Grenzen van de Arduino  
Denk na over de limiteringen van bijvoorbeeld de Arduino in het ontwerp.

Opdracht 4: Eindopdracht

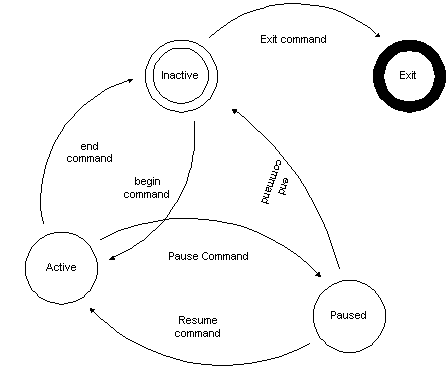
4.1 Opdrachtomschrijving  
De eindopdracht is een gecombineerd product met als basis de producten zoals beschreven bij **(zie: hfst x.x)**.   
  
  
4.2 Analyse wensen en eisen   
Hardware / Software /   
(prioritiseer deze)  
  
Eisen:  
Het netwerk moet minimaal bestaan uit:  
1. Minstens twee *weerstations* die hun data doorsturen naar de *gateway*.   
*1. Gateway* stuurt de data naar de *webserver*.  
*2. Webserver* moet deze data kunnen visualiseren in een website met bijvoorbeeld grafieken.  
3. De temperatuurgrenzen, voor de leds, moeten *per weerstation* kunnen worden geconfigureerd via de *gateway*.  
4. De data moet herleidbaar zijn naar het desbetreffende specifieke weerstation/gateway. Dus de identificatie mag niet (alleen) op basis van IP-nummers gebeuren.  
5. Elke weerstation moet minimaal dezelfde functionaliteiten hebben als het weerstation uit week 2 **beschreven in (hfst 6 Eindopdracht “weerstation” – week 2 Web Services.pdf)**.

Wensen:  
Documentatie

4.3 Globale architectuur   
Welke tools ga ik gebruiken om de totale architectuur overzichtelijk weer te geven.  
*Wat moet ik laten zien?*   
🡪fysieke componenten   
🡪Digitale componenten  
🡪 Software o.a. d.m.v. State machines,   
🡪communicatie  
🡪 protocol beschrijvingen

--> beschrijf waar de fysieke componenten zich bevinden en op welke laag in het OSI-model ze invloed (mogen/kunnen) hebben.   
Hoofdcomponenten:  
Webserver  
Gateway  
Sensor Nodes  
  
Digitale componenten:  
MySQL database  
Sensor Node Code  
Gateway Code  
Webserver Code   
C# code

State machines gebruiken om te laten zien welke componenten wanneer een taak uitvoeren.  
Zo heb je in de sensor nodes de ‘interne temperatuur check’, die de LED’s constant blijft beheersen. Maar er MOET ook actief gereageerd kunnen worden op een Request vanaf de webservice.

  
  
--> hoe ze met elkaar communiceren  
--> welke berichten worden er verstuurd?

4.4 Planning  
Maak een planning van welke onderdelen er zijn en in welke volgorde ik deze maak.   
Wat heb ik uit week 1 - arduino en internet nodig?  
Wat heb ik uit week 2 - webservices nodig?  
  
*Tijdens de course veel iteratief gewerkt (bron?) onder andere doordat de PDF documenten met opdrachten over hetzelfde onderwerp gingen: een webservice bouwen met de Arduino.* Tijdens de documenten worden er extra functionaliteiten toegevoegd en

Iteratie 1  
Sensor Nodes  
Gateway  
Webserver  
  
  
Iteratie 10  
Iteratie 100

Gateway  
Sensor Node  
Webserver

Planning eindopdracht  
De eindopdracht bestaat uit de volgende onderdelen:  
🡪 ontwerp en realisatie van de fysieke onderdelen

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pdf wk1 | Pdf Wk2 |  |
| Sensor node(s) | - |  | Minimaal 1 node gaat zich gedragen als actief benaderbaar eind-punt en data opsturen naar de gateway |
| Gateway | - | Combi van gateway en Sensornode.  stuurt via REST-protocol dmv jason data naar webserver | Gateway staat 24/7 te luisteren naar Sensor Nodes om vervolgens de data door te sturen naar de Webservice. |
| Webserver | Simpele versie met HTML. REST via Arduino | Webserver krijgt controllers om HTTP Requests af te handelen en Responses te geven. Ontvangen data word in MySQL opgeslagen. | De webservice |

🡪 Ontwerp, realisatie en versiebeheer van de benodigde software

Documenteren

5. Conclusie  
conclusie

6. Bronnenlijst  
Bronnenlijst

# 7. Bijlagen

Bijlagen