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

Opdrachten:  
Opdracht 1: Parseren request - line  
  
Opdracht 2: GET en POST op de Arduino  
  
Opdracht 3: Weerstation als webserver  
Deel opdracht A  
testopstelling / sketch  
  
Deel opdracht B  
1. Functionaliteiten  
2. Ontwerp  
3. Communicatieprotocol  
4. Limitaties  
  
Opdracht 4:Eindopdracht   
1. Opdracht omschrijving  
2. Eissen en wensen  
3. Architectuur  
4. Planning

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)  
Deel opdracht A  
//Bouw de basisopstelling na  
//Test deze dmv 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 B  
1. Lijst met functionaliteiten  
2. Maak een gedegen ontwerp. dmv. state-machines/sequence diagrams, activity diagrams.  
3. 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)   
4. Denk na over de limitaties van bijvoorbeeld de arduino in het ontwerp.

Opdracht 4: Eindopdracht  
1. Opdrachtomschrijving  
De eindopdracht is een gecombineerd product met als basis de producten zoals beschreven bij **(zie: hfst x.x)**.   
  
  
2. Analyse wensen en eisen (prioritiseer deze)  
Hardware / Software /   
  
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  
  
  
3. Maak een globale architectuur.   
--> beschrijf waar de fysieke componenten zich bevinden  
Hoofdcomponenten:  
Webserver  
Gateway  
Sensor Nodes

--> hoe ze met elkaar communiceren

--> welke berichten worden er verstuurd?

4. 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?

Gateway  
Sensor Node  
Webserver

Conclusie  
  
Bronnenlijst  
  
Bijlagen