Protocol Document

Smart Bee Hive

Measurement Unit

Project Code:	2016NJ-SBH
Document Nr:	FONTYS-1019PD0001
Document Naam:	Protocol Document
Auteur(s):	Rafał Grasman (Fontys-Groep-1 / T34)
Revisie:	1
Datum:	2016-10-19
Status:	Draft

Documenthistorie

Revisies

Revisie	Status	Datum	Wijzigingen
1	Draft	2016-10-19	Document aangemaakt

Goedkeuring

Dit document behoeft de volgende goedkeuringen:

Revisie	Datum goedkeuring	Naam	Functie
1	2016-10-31	Rafal Grasman	Schrijver van Protocol Document
1			

Distributie

Dit document is verstuurd aan:

Revisie	Datum verzending	Naam	Functie
1	2016-10-31	Op Slack Chat room	Algemeen Communicatiekanaal

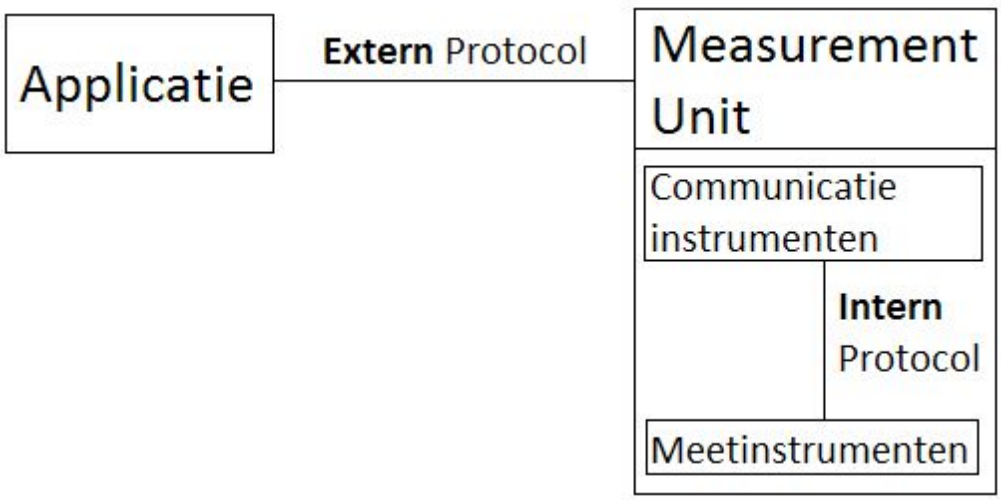
Inhoudsopgave

Inleiding

In dit document zal worden beschreven op welke manier de communicatie zal verlopen tussen de *Measurement Unit* en de *Applicatie* en intern in de Measurement Unit. Voor de protocollen worden de volgende onderwerpen besproken: Formaat van de data, welke technologieën worden voor communicatie gebruikt, functies per protocol onderdeel, parameters van alle functies (data), en alle mogelijke return waarden van elke functie.

Schema

In figuur 1 is het schema te bezichtigen, hierin wordt visueel omschreven waar de externe protocol en de interne protocol aanwezig zijn.



Figuur 1 - Schema protocollen (voor PXL is Extern protocol van belang, voor Fontys: Intern)

Keuze technologie

Er is gekozen voor het gebruik van WebSockets, dit omdat dit een geschikt, simpel en snel message-based protocol is voor real-time communicatie. Ook is WebSockets gestandariseerd en globaal beschikbaar voor alle populaire programmeertalen.
De keuze voor JSON als het data formaat is omdat het zich kenmerkt in:

1) Het een schoon formaat is om data weer te geven;
2) Simpel in data weergave en simpel om te debuggen
3) Simpel om te parsen en er zijn parsers (serializers en deserializers) beschikbaar voor alle populaire programmeertalen.

Data Formaat

Data die uitgewisseld wordt door de protocollen zal in JSON formaat zijn. Dit formaat is gekozen omdat het eenvoudig te verwerken is, makkelijk te leren, simpel om te lezen, en voor elke populaire programmeertaal implementaties bestaan voor het parsen, verwerken en creëren van JSON geformatteerde data.

Communicatie Protocol

Voor de communicatie over het netwerk (Extern protocol) is gekozen voor WebSockets technologie. Deze is gekozen omdat het een simpel, robuust en gestructureerd protocol bovenop TCP is die voor alle populaire programmeertalen een bestaande implementatie beschikbaar heeft, en zelf simpel te implementeren is. Omdat het universeel gebruikt wordt voor real-time communicatie met veelal services hedendaags is WebSockets een veelzijdige oplossing in dit geval. De communicatie voor de meetinstrumenten zal verlopen over seriële verbinding. Seriële verbindingen zijn simpel om op te stellen en elk apparaat in de Measurement Unit beschikt over deze functionaliteit, er is daarvoor geen extra hardware nodig en er zijn hoogstens triviale aanpassingen nodig. Om te samenvatten zijn figuur 2 en figuur 3 te bezichtigen.

Data Layer	JSON Formatted Data
Communication Layer	Websockets
Network Layer	TCP
Physical Layer	Ethernet/WiFi/Bluetooth/3G/4G

Figuur 2 - Extern Protocol opbouw

Data Layer	JSON Formatted Data
Communication Layer	Serial Connection
Physical Layer	Serial Hardware

Figuur 3 - Intern Protocol opbouw

Algemene opbouw JSON data

De basis JSON structuur voor het versturen van data bestaat uit:

1) Het berichttype (in dit geval actie)
2) De actie die ondernomen moet worden
3) (optioneel) een ID om de actie uniek te identificeren (bij een response op de actie)
4) De actie parameters (indien aanwezig)

De JSON hiervoor ziet er als volgt uit, bijvoorbeeld voor het opnemen van de temperatuur van sensoren 1 3 en 8 (van de bijvoorbeeld maximaal 8):

```
{
	"type":"action",
	"action":"GetTemperature",
	"id":18457,
	"temperatures":[1,3,8]
}
```

Daarnaast is het belangrijk dat er een JSON structuur afgesproken wordt voor de antwoorden (responses):

1) Het berichttype (in dit geval response)
2) De actie waarbij de response hoort
3) Status (error, warning of success)
4) Het statusbericht
5) De statuscode (bijvoorbeeld indien error/warning, de code die bij de error/warning message hoort, indien success is de code 1, indien onbekende fout/warning, code 0)
6) Indien opgegeven bij de actie, een ID die bij de actie is meegegeven
7) Return parameters indien de actie data terugstuurt

De bijbehorende response voor het laatste voorbeeld kan er als volgt uitzien:

```
{
	"type":"response",
	"status":"success",
	"status_code":1,
	"message":"",
	"action":"GetTemperature",
	"id":18457,
	"temperatures":
	{
		"1":15,
		"3":18,
		"8":16
	}
}
```

Extern Protocol

Hier worden de mogelijk acties en responses beschreven die van buitenaf uit te voeren zijn, inclusief alle parameters en wat deze doen, welke waarden mogelijk zijn en welke error codes ge-returnt worden, en wat de error codes betekenen. De acties worden altijd verstuurd door de aanvrager en de responses worden verstuurd door de ontvanger van de acties. Bij alle types wordt er van de default types uitgegaan (string, integer, float, e.d.).

GetSensorData

Beschrijving: Leest de huidige data van de gespecificeerde sensoren.
Actie Parameters:

1) (optioneel) ‘sensors’ : array van strings van welke sensoren moeten worden uitgelezen, indien niet gespecificeerd worden alle sensoren uitgelezen. De volgende sensoren zijn mogelijk om uit te lezen:
    a) ‘temperature’ : in graden celsius
    b) ‘sound’ : in decibel
    c) ‘weight’ : in kilogram
    d) ‘humidity’ : in %
    e) ‘pressure’ : in atmosfeer
    f) ‘light-intensity’ : in lux

    Indien er meerdere sensoren aanwezig zijn voor een type, kan na de string van de type een array worden meegegeven van de sensoren die uitgelezen moeten worden, indien deze array ontbreekt zullen alle sensoren worden uitgelezen. JSON voorbeeld:

```
"sensors" : {
        "temperature": [ 0, 1, 7 ],
        "weight": []
}
```

Response Parameters:

1) ‘sensors’: array van welke sensor welke data meet, incl welke sensor index bijvoorbeeld

```
"sensors" : {
        "temperature": { 0:15, 1:16, 7:10 },
        "weight": { 0:14.43 }
}
```

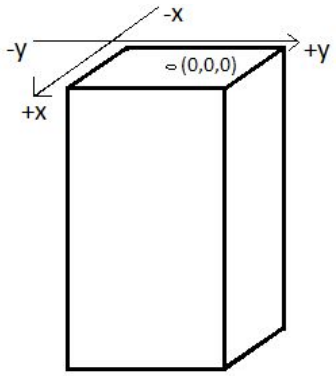
Mogelijke Error Codes (“status_code”):

2 - Apparaat beschikt niet over de apparatuur om temperatuur te meten.
3 - Apparaat beschikt niet over de apparatuur om geluid te meten.
4 - Apparaat beschikt niet over de apparatuur om gewicht te meten.
5 - Apparaat beschikt niet over de apparatuur om luchtvochtigheid te meten.
6 - Apparaat beschikt niet over de apparatuur om luchtdruk te meten.

7 - Apparaat beschikt niet over de apparatuur om lichtintensiteit te meten.

GetSensorSetup

Beschrijving: Leest de huidige configuratie van de gespecificeerde sensoren. De mogelijke configuratieparameters zijn: “**min**” - minimale waarde, “**max**” - maximale waarde, “**unit**” - in welke eenheid gemeten wordt, “**position**” - plaatsing in/bij de bijenkast ten opzichte van top-midden in xyz (lbh - richting naar deur = +x) in meters, “**placement**” - waar de sensor zich bevindt, mogelijke waarden: “*inside*” (binnen) en “*outside*” (buiten).



Positionering (er wordt tegen de voorkant aangekeken)

Actie Parameters:

1) (optioneel) ‘sensors’ : array van strings van welke sensoren moeten worden uitgelezen, indien niet gespecificeerd worden alle sensoren uitgelezen. De volgende sensoren zijn mogelijk om uit te lezen:
    g) ‘temperature’
    h) ‘sound’
    i) ‘weight’
    j) ‘humidity’
    k) ‘pressure’
    l) ‘light-intensity’

    Indien er meerdere sensoren aanwezig zijn voor een type, kan na de string van de type een array worden meegegeven van de sensoren die uitgelezen moeten worden, indien deze array ontbreekt zullen alle sensoren worden uitgelezen. JSON voorbeeld:

```
"sensors" : {
        "temperature": [ 0, 7 ],
        "weight": []
}
```

Response Parameters:

1) ‘sensors’: array van welke sensor welke configuratie bevat, incl welke sensor index bijvoorbeeld


```
{
 "sensors":{
  "temperature":{
   "0":{
   "min":-50, "max":100, "unit":"*C", "placement":"outside", "position":[ 0, 0, 0.1 ]
   },
   "7":{
   "min":-50, "max":100, "unit":"*C", "placement":"inside", "position":[ 0, 0, -0.1 ]
   },
  },
  "weight":{
   "0":{
   "min":0, "max":65, "unit":"Kg", "placement":"outside", "position":[ 0, 0, -1.25 ]
   },
  }
 }
}
```


Mogelijke Error Codes:

2,3,4,5,6,7, voor beschrijvingen zie “GetSensorData - Mogelijke Error Codes”

GetDeviceSetup

Beschrijving: Leest de huidige configuratie van het apparaat. De mogelijke configuratieparameters zijn: “**serial**” - de seriele code van het apparaat, “**firmware**” - array met informatie over de firmware, “**timestamp**” - huidige tijd in unix timestamp formaat, “**timezone**” - tijdzone.

Actie Parameters:

Geen parameters

Response Parameters:

1) “serial” als string
2) “firmware” als een array met informatie voor:
    a) De “server”:
        i) “os” - operating system informatie & versie
        ii) “free” - vrije opslagruimte in megabytes voor meetdata
    b) De “measurement” apparaten:
        i) “version” - firmware versie
3) “devices” - aangesloten sensoren lijst als array incl hoeveelheid van elke sensor die aanwezig is
4) “capabilities” - lijst van mogelijke verbindingsmethoden (“ethernet10m”, “ethernet100m”, “ethernet1000m”, “ethernet10gbps”, “3g”, “4g”, “wifi24ghz80211b”, “wifi24ghz80211g”, “wifi24ghz80211n”, “wifi50ghz80211a”, “wifi50ghz80211h”, “wifi50ghz80211j”, “wifi50ghz80211n”,“ wifi50ghz80211ac”, “bluetooth20”, “bluetooth21”, “bluetooth30”, “bluetooth31”, “bluetooth40”, “bluetooth41”, “bluetooth42”, “bluetooth50”, bij elke capability worden de netwerkgegevens getoond indien deze verbonden is
5) “protocol” - protocol revisie

Voorbeeld:


```
{
    "setup": {
        "serial": "A5683E32C2F937D4",
        "capabilities": {
                "ethernet100m": {
                        "ip": "10.0.0.4/8",
                        "gateway": "10.0.0.1",
                        "configuration": "DHCP"
                },
                "bluetooth42": {},
                "3g": {},
                "4g": {},
                "wifi24ghz80211b": {}
        },
        "devices": {
                "temperature": 2,
                "weight": 1,
                "sound": 4
        },
        "firmware": {
                "server": {
                        "os": "Linux HOSTNAME 3.4.0+ #1 PREEMPT Thu Aug 1 17:06:05 CST 2013
x86_64 x86_64 x86_64 GNU/Linux",
                        "free": "13423"
                },
                "measurement": {
                        "version": "1.0.1.1"
                },
                "protocol": 1
        }
    }
}
```


Mogelijke Error Codes:
nvt

Begrippen

Measurement Unit - Gedeelte van het Smart Bee Hive project dat zorgt voor het verzamelen en meten van data. Wordt gemaakt door Fontys.

Applicatie - Overig deel van het Smart Bee Hive project dat wordt gemaakt door PXL.

Populaire programmeertalen - hieronder wordt verstaan: Programmeer- en Scripttalen zoals JavaScript, Python, Lua, Ruby, Java, C, C++, C# (en andere .NET talen), Go en PHP.