

KTH Projekt och projektmetoder II-1302Teknisk dokumentation
FlowerPower*Wifi-kommunikation mellan
Arduino Uno och Raspberry Pi*

Date	Version	Author	Description
22/04/2018	1	Erik Holm	Första draft
28/04/2018	2	Erik Holm	Komplettering med beskrivande bilder
14/05/2018	3	Erik Holm	Redigering av text
22/05/2018	4	Erik Holm	Sista draft

1. Introduktion
2. Material
3. Utvecklingsmiljö
4. Implementeringen
 - 4.1 Kommunikation mellan ESP8266 och Arduino
 - 4.2 Kommunikation mellan Arduino och Sensor
 - 4.3 Anslutning till nätverk
 - 4.4 Mottagandet av request från client
5. Testspecifikation
6. Sammanfattning
 - 6.1 Eventuella risker vid fortsatt implementation
 - 6.2 Miljö- och hållbarhetsaspekter och Etik

Referenser

1. Introduktion

Det huvudsakliga ansvarsområdet är kommunikation mellan:

Sensor->Arduino->ESP8266->Raspberry

Dokumentets syfte är att underlätta fortsatt underhåll av befintlig modul och implementering av ny modul. Dokumentet går först igenom materialval och utvecklingsval för att därefter gå igenom implementeringen steg för steg. Avslutningsvis diskuteras eventuella risker vid underhåll och reflektioner kring val av hårdvara.

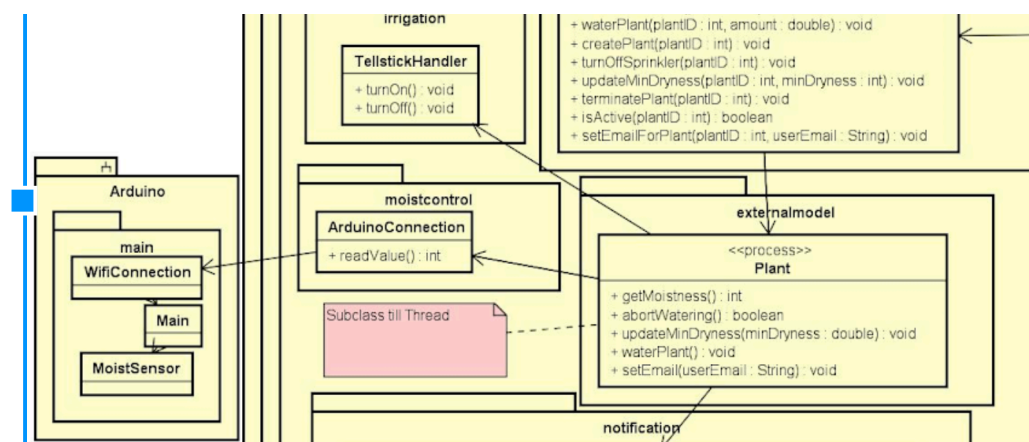


Bild 1 - Arkitektur för kontakt mellan Arduino->Raspberry Pi

Som man ser i arkitekturen i bild 1 kommunicerar Arduino Unos MoistSensor med ArduinoConnection på vår Raspberry Pi. När metoden readValue() kallas på returneras värdet. Kommunikationen mellan sensor, wifi-modul och Arduino sker genom koppling på breadboard. Kommunikationen till Raspberry Pi sker genom wifi.

2. Material

- Arduino Uno MicroController
- ESP8266 Wifi-modul
- Breadboard
- Female/Male Wires
- Raspberry Pi
- Elektrokit fuktmätare

3. Utvecklingsmiljö

- Arduinos egna utvecklingsmiljö och språk
- Python för raspberry Pi
- Tester i pytest

4. Implementering

4.1 Kommunikation mellan ESP8266 och Arduino

Vi vill skicka ett värde från Arduino till Raspberry genom wifi så naturligt var det första steget att kommunicera mellan Arduino Uno och ESP8266. På så sätt skulle vi kunna skicka vidare det värde som plockades upp från sensorn.

Arduinos standardinställning är att kommunicera med wifi-modulen via seriellporten på GPIO-stift 0 och 1 (RX och TX). Här sker en krock eftersom man använder den för att skicka data till Serial monitor på datorn och det går inte att dela på porten. Det löste jag med library SoftwareSerial som leder kommunikationen till andra stift än de redan nämnda stiften.

Det gjorde man genom att öppna upp serials kommunikationsfönster och ange kommandot:

```
AT+UART_DEF=9600,8,1,0,0
```

Kommunikation kunde nu ske med ändrad Baud(hur många gånger per sekund som vår signal ändras).

Kopplingen mellan ESP8266 kopplades sedan som i bild 2 där pin 6 samt 7 nu kan kommunicera med ESP:n:

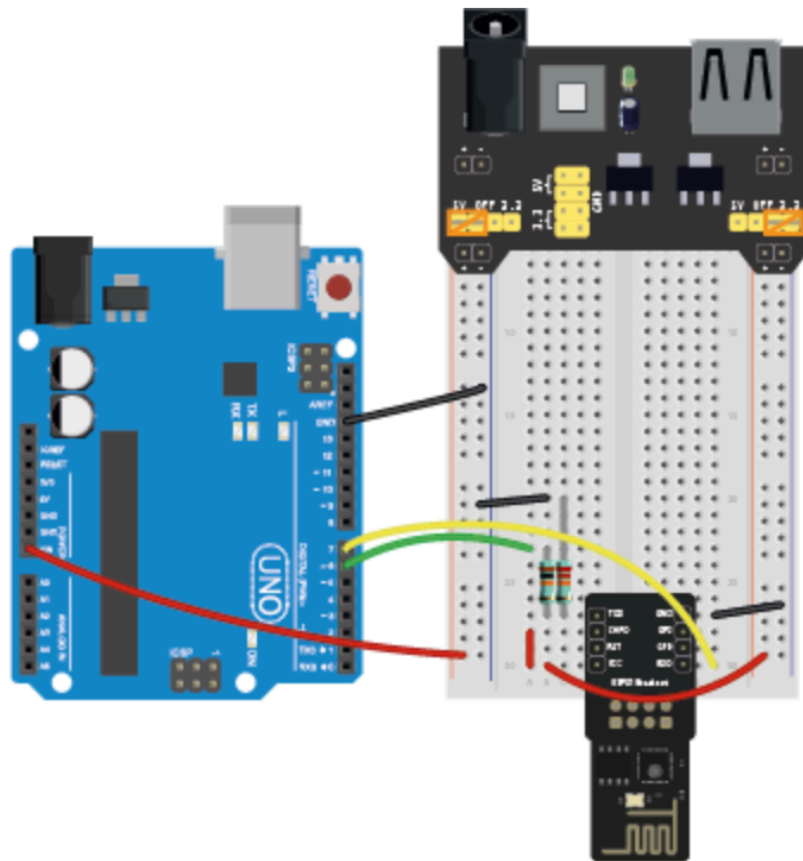


Bild 2 - Kopplingsschema mellan wifi-modul och Arduino

Slutligen sätter vi in två resistorer på 10 kohm samt 22 kohm för att sänka spänningen till 3.3V istället för den ursprungliga 5V. Det är inte bra att driva ESP:n med 5V, vilket är alldeles för högt.

4.2 Kommunikation mellan Arduino och Sensor

Härnäst behöver Arduino plocka upp ett värde från Sensorn. Sensorn kopplades till analog pin A0 och kunde plockas upp direkt till Arduinon. Det krävdes inga fler förberedelser vid användning av en sensor. Vid implementering av fler än en sensor kan det uppstå brist på output pins. Läs kravspecifikation av icke-funktionella krav för förslag på tillvägagångssätt. Sammanfattning (punkt 6.1) sammanfattar en eventuell lösning.

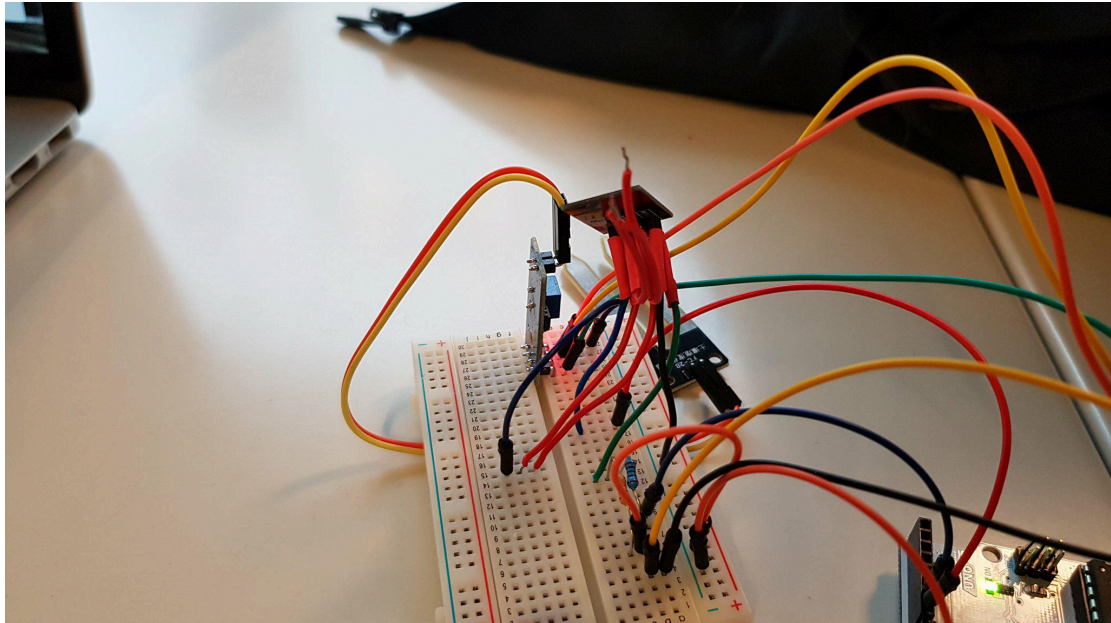


Bild 3 - Breadboard för sensor/wifi-modul/arduino sett från sidan

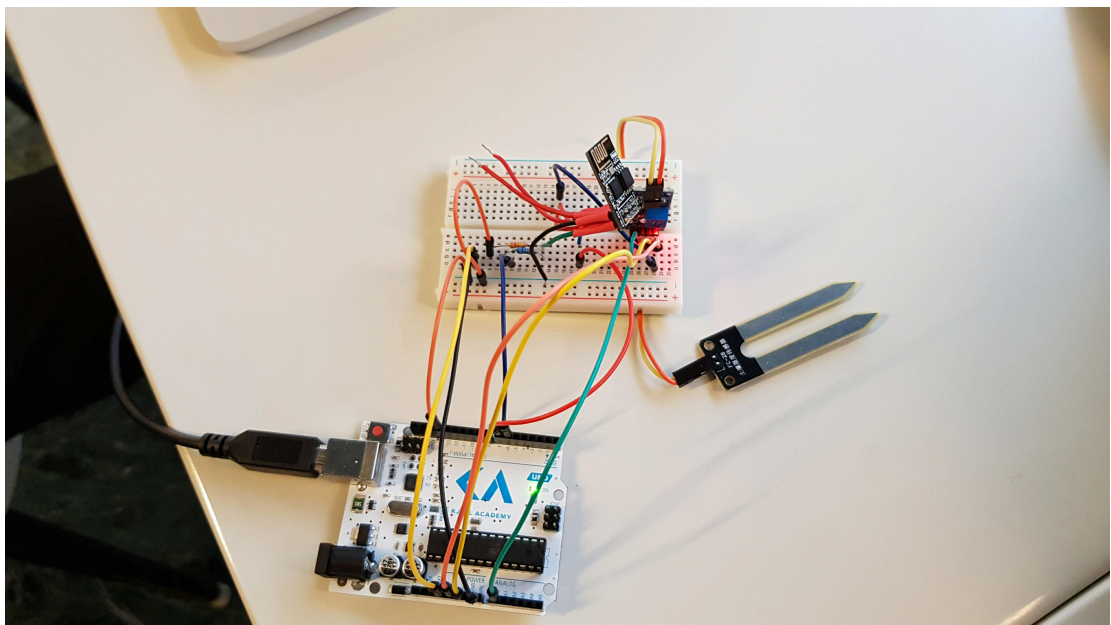


Bild 4 - Breadboard för sensor/wifi-modul/arduino sett ovanifrån

4.3 Anslutning till nätverk

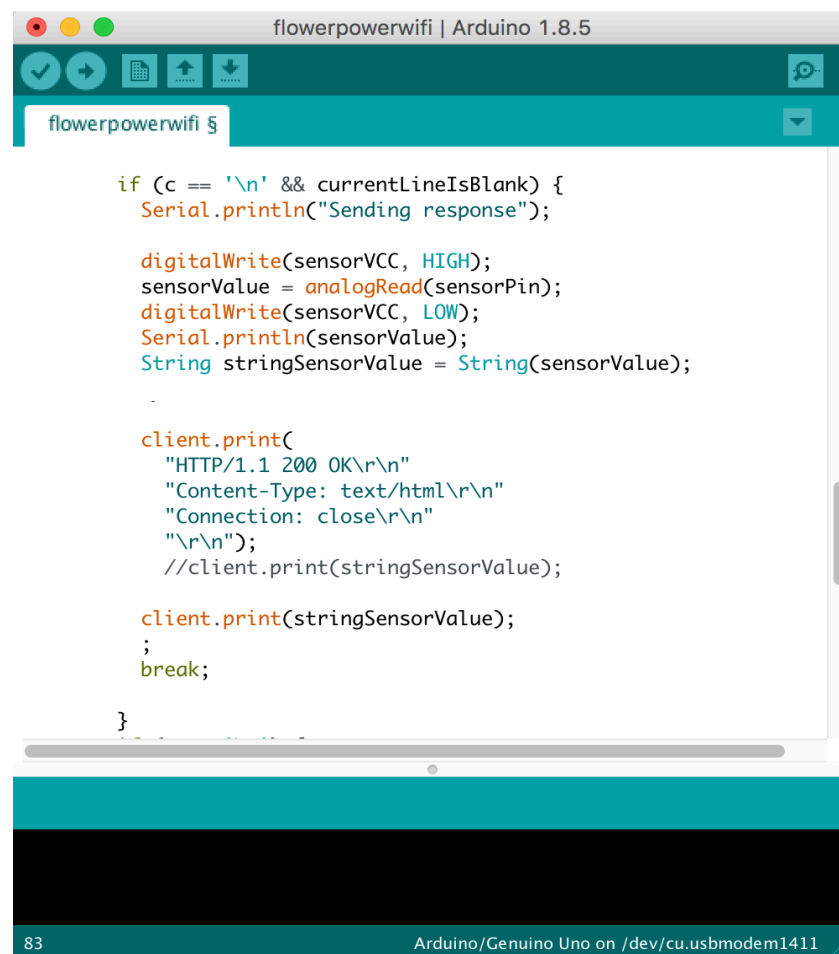
När värdet väl hade plockats upp från sensorn krävdes det en wifi-anslutning genom ESP:n. Vi hade nu en baud rate att serial-kommunicera med ESP-modulen. Därefter använde jag mig av ett bibliotek WiFiESP.h. På så sätt kunde man skicka en initialize request till ESP-modulen. Kommando var följande:

Init(Serial) – Parameter är den påbörjade serial-kommunikationen mellan ESP och Arduino.

Begin(SSID,PASS) – Nätverket lösenord och användarnamn är parameter för att ansluta till WPA/WPA2 nätverket.

4.4 Mottagandet av request från client

Vi har nu kontakt med nätverket och en web server kan startas för att ta emot requests. Arduinon har en loop-funktion som väntar på en http request från client. Svaret blir det värde som Arduinon plockar upp från sin sensor.



```
flowerpowerwifi | Arduino 1.8.5

flowerpowerwifi $

if (c == '\n' && currentLineIsBlank) {
  Serial.println("Sending response");

  digitalWrite(sensorVCC, HIGH);
  sensorValue = analogRead(sensorPin);
  digitalWrite(sensorVCC, LOW);
  Serial.println(sensorValue);
  String stringSensorValue = String(sensorValue);

  client.print(
    "HTTP/1.1 200 OK\r\n"
    "Content-Type: text/html\r\n"
    "Connection: close\r\n"
    "\r\n");
    //client.print(stringSensorValue);

    client.print(stringSensorValue);
  ;
  break;
}

83 Arduino/Genuino Uno on /dev/cu.usbmodem1411
```

Bild 5 - Arduino-loop plockar upp värde och skickar en text/html till client

Informationen görs om till en string och skickas som html när en request görs från raspberry pi:n. Pi:n gör sin request på följade sätt, med hjälp av biblioteket requests.

```
import json
import requests

class ArduinoConnection:
    #def __init__(self):
    #wlan = WLAN(mode=WLAN.STA)
    #wlan.connect("Isabels iPhone", auth=(net.sec, "ifpejnn09q53y"), timeout=5000)

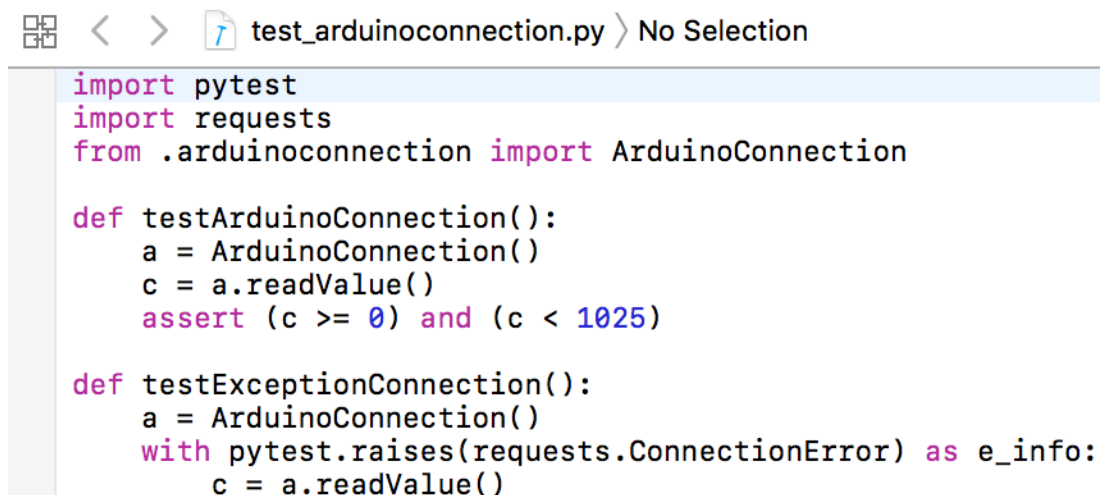
    def getValue(self):
        # r = requests.get('http://172.20.10.3/getValues.json')
        r = requests.get('http://172.20.10.3/')

```

Bild 6 - Raspberrykod för request där svaret sparas som "r"

5. Testspecifikation

Tester för kommunikation skrevs i pytest. Som man ser i bild 7 testas att värdet har ett rimligt värde och att vi inte har fått något fel i anslutningen med server (Arduino Uno). Läs pytests dokumentation och se exempel nedan vid skrivandet av ytterligare testkod.



```
test_arduinoconnection.py > No Selection

import pytest
import requests
from .arduinoconnection import ArduinoConnection

def testArduinoConnection():
    a = ArduinoConnection()
    c = a.readValue()
    assert (c >= 0) and (c < 1025)

def testExceptionConnection():
    a = ArduinoConnection()
    with pytest.raises(requests.ConnectionError) as e_info:
        c = a.readValue()

```

Bild 7 - Pytest-kod med sensorvärdet som svar på request

Test för Arduino fixas internt genom kommunikationen med logg. Det fungerar mest som intern felsökning. All kommunikation som berör användaren hanterar redan pytest på clientsidan. Notera att det är viktigt att skriva ut status till serial för att få en överblick för varje moment (se bild 8).


```
You're connected to the network
SSID: Xperia Z5 Compact_2634
IP Address: 192.168.43.172
[WiFiEsp] Server started on port 80
[WiFiEsp] New client 0
New client
GET / HTTP/1.1
Host: 192.168.43.172
Accept-Encoding: gzip, deflate
Connection: keep-alive
User-Agent: python-requests/2.18.4
Accept: */*

Sending response
999
[WiFiEsp] Disconnecting 0
```

Bild 8 - Exempel på utförlig dokumentation i Arduinokod

6. Sammanfattning

6.1 Eventuella risker vid fortsatt implementation

Vid användning av fler än en sensor kan det behövas en extern modul för fler analoga inputs. En lösning vore en extern power supply alternativt en parkoppling mellan micro controllers. Ett övervägande som får ta med projektets budget i beräkning.

Spänningsförsörjning till Arduinon når inte alltid den utlovade 5V vilket gör att GND måste kopplas in igen efter en första uppstart, det här behöver göras för att det sker en initiell ökning i spänning-/strömförsörjning. För att slippa det krävs en extern strömkälla.

6.2 Miljö- och hållbarhetsaspekter och Etik

Vid val av en extern strömkälla och övriga komponenter är det viktigt att kontrollera inköpskällor, både ur ett etiskt och även ett miljöperspektiv.

E-waste eller E-skrot är utdaterad hårdvara som skickas till utvecklingsländer för återanvändning/ihopsamlande av komponenter. Behovet att byta ut elektronik även om komponenterna funkar ökar naturligtvis de mängder E-skrot som vi producerar årligen. Det syns även i statistiken då vi varje år exporterar 20-50 miljoner ton E-skrot. Anledningen till osäkra siffror beror på att mycket

sker illegalt. Om det här sker på ett okontrollerat sätt kan det påverka människors hälsa när farliga komponenter tas isär.

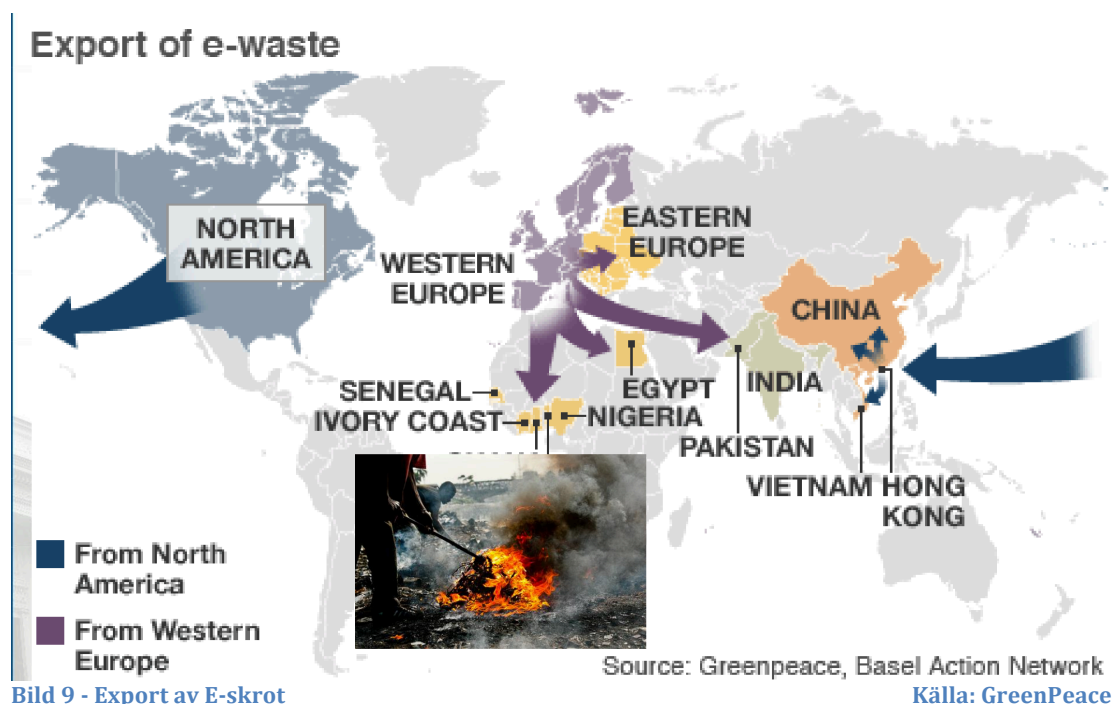


Bild 9 - Export av E-skrot

Som man kan se i bild 9 så är exporten ensidigt från I-länder till utvecklingsländer. Det är därför ett gemensamt ansvar att se över var ens hårdvara tillverkas och under vilka förhållanden. I vårt projekt kom de flesta komponenter från Kjell & Co. Även om Kjell&Co. bara eftersträvar en fullständig dokumentation så har de ett dagligt arbete för att garantera en CE-märkning. För mer information så hänvisas man besöka deras hemsida² CE-märkning visar att produkten överensstämmer med grundläggande krav på exempelvis hälsa, säkerhet, funktion, miljö enligt en EU-standard.

Referenser

[1] <https://www.greenpeace.org/archive-international/en/campaigns/detox/electronics/the-e-waste-problem/>

[2] <https://www.kjell.com/se/om-oss/hallbarhet>