

1.	Introduktion.....	2
1.1.	Formål og Afgrænsning/Scope .....	2
1.2.	Referencer .....	2
2.	Resume .....	4
2.1.	Opgavebeskrivelse.....	4
2.2.	Valgte Parter .....	4
3.	Analyser .....	5
3.1.	Touch Sensor (S1) .....	5
3.2.	RGB Color Sensor (S2).....	7
3.3.	Angle Sensor (S3).....	9
3.1.	Relay (A1).....	10
3.2.	Aktuator Buzzer/Sounder (A2) .....	11
3.3.	Platform IoT Device (P1).....	13
3.4.	Weather Service (W1) .....	15
3.5.	Sonos (W2) .....	16
4.	Bilag, Test af RGB Sensor .....	20
4.1.	Verificer RGB Color Sensor med AD2 Protocol Analyzer .....	20
5.	Bilag, Test af Vejr API (OpenWeatherMap) .....	32

## 1. Introduktion

### 1.1. Formål og Afgrænsning/Scope

Dette dokument indeholder analyser og testresultater for udvælgelse af egnede komponenter og sensor-moduler, der opfylder de formelle krav til funktionalitet som beskrevet i kravsspecifikationen [1].

Dokumentet indeholder ikke information om hvorfor den enkelte funktionalitet er vurderet til at være nødvendig, da dette er forklaret i projektets introduktionsvideo [2]

Design for implementation afsnit afspejler kun et øjebliksbillede i den indledende fase, men der henvises til diagrammerne der forefindes i sub-folders, der udgør et ”hjem” for den enkelte komponent.

### 1.2. Referencer

Ref	Beskrivelse	Link
[1]	Kravspecifikation	Er i MiotyWaffles projektrapporten
[2]	Projektets introduktionsvideo	<a href="https://www.youtube.com/watch?v=oOoYVhcU9do">https://www.youtube.com/watch?v=oOoYVhcU9do</a> Upload version:
[3]	TCS3472 Datasheet (RGB Sensor)	<a href="https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf">https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf</a> Rev. "TAOS135 – August 2012" Senest hentet d. 19-09-2020
[4]	PKM22EPP-4001-B0 datasheet (Buzzer/Sounder)	<a href="https://www.arduino.cc/documents/datasheets/PIEZO-PKM22EPPH4001-BO.pdf">https://www.arduino.cc/documents/datasheets/PIEZO-PKM22EPPH4001-BO.pdf</a> Rev. " P37E17.pdf 02.3.6" Senest hentet d. 23-09-2020
[5]	TCSW520 datasheet (Angle Sensor)	<a href="https://asset.conrad.com/media10/add/160267/c1/-/en/001606623DS01/datablad-1606623-tru-components-haeldningssensor-tc-sw-520-tc-sw-520-maleomrade-180-max-loddepins.pdf">https://asset.conrad.com/media10/add/160267/c1/-/en/001606623DS01/datablad-1606623-tru-components-haeldningssensor-tc-sw-520-tc-sw-520-maleomrade-180-max-loddepins.pdf</a> Rev. " V1_18012018_01_en" Senest hentet d. 23-09-2020
[6]	Particle Argon Datasheet (Platform)	<a href="https://docs.particle.io/datasheets/wi-fi/argon-datasheet/">https://docs.particle.io/datasheets/wi-fi/argon-datasheet/</a> Rev. "V004" Senest hentet 24-09-2020
[7]	Sonos Developer Portal (For REST API)	<a href="https://developer.sonos.com/">https://developer.sonos.com/</a> Senest hentet 24-09-2020
[8]	Adafruit 1334 Technical data	Eagle Diagram m.m. <a href="https://learn.adafruit.com/assets/15703">https://learn.adafruit.com/assets/15703</a> Senest hentet 09-10-2020
[9]	Weather API	Vejr API, OpenWeatherMap <a href="https://openweathermap.org/">https://openweathermap.org/</a> Senest hentet 22-10-2020
[10]	Touch Sensor Datasheet	AT42QT1010 Datasheet <a href="http://ww1.microchip.com/downloads/en/DeviceDoc/40001946A.pdf">http://ww1.microchip.com/downloads/en/DeviceDoc/40001946A.pdf</a> Senest hentet 23-10-2020

<b>[11]</b>	Relæ (module og komponent)	Module: <a href="https://www.elextra.dk/p/5v-rel%C3%A6modul-m-optokobler-til-arduino-2-kanals/H16117">https://www.elextra.dk/p/5v-rel%C3%A6modul-m-optokobler-til-arduino-2-kanals/H16117</a> Senest hentet 03-11-2020  Relæ komponent: <a href="https://datasheetspdf.com/datasheet/SRD-12VDC-SL-C.html">https://datasheetspdf.com/datasheet/SRD-12VDC-SL-C.html</a> Senest hentet 03-11-2020
<b>[12]</b>	Vaffeljern, OBH nordica elite EAN 5708642069533	Vaffeljern specifikation <a href="http://www.obhnordica.dk/madlavning/vaffeljern/elite-single-stainless-steel">www.obhnordica.dk/madlavning/vaffeljern/elite-single-stainless-steel</a> Senest hentet 03-11-2020

*Tabel 1: Referencer*

## 2. Resume

### 2.1. Opgavebeskrivelse

Der skal i produktet bruges en Touch Sensor til at give brugeren mulighed for at påbegynde processen. Et relæ skal bruges til at agere tænd/sluk kontakt, og en RGB farvesensor bruges til at registrere tilstanden af vaffeljernet ud fra dets integrerede status LED. Der bruges en Vinkelsensor til at vurdere om låget er åbent eller lukket, og der bruges en lyd-giver/Buzzer til at kommunikere hørbare systembeskeder til brugeren.

### 2.2. Valgte Parter

#### 2.2.1. Sensorer

ID	Komponent Funktion	Valgt Komponent	Forhandler
S1	Touch Sensor	AT42QT1010	Conradelektronik.dk
S2	RGB Color Sensor	Adafruit 1334 (TCS34725)	ArduinoTech.dk
S3	Angle Sensor	TCSW520	Conradelektronik.dk

Tabel 2: Sensorer der skal bruges til projektet

#### 2.2.2. Aktuatorer

ID	Komponent Funktion	Valgt Komponent	Forhandler
A1	Relay	H16117	Elextra.dk
A2	Buzzer	PKM22EPP-4001-B0	AU Herning

Tabel 3: Aktuatorer der skal bruges til projektet

#### 2.2.3. Platform

ID	Komponent Funktion	Valgt Komponent	Forhandler
P1	IoT Device (System MCU)	ARGN-H (Particle Argon)	Elfa Distrelec

Tabel 4: Aktuatorer der skal bruges til projektet

#### 2.2.4. Vaffeljern

ID	Komponent Funktion	Valgt Komponent	Forhandler
U1	Vaffeljern (1000W)	OBH Nordica Elite 6953	Proshop.dk

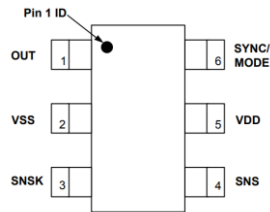
Tabel 5: Vaffeljern der skal bruges til projektet

### 3. Analyser

#### 3.1. Touch Sensor (S1)

##### 3.1.1.1. Part Detaljer

Der er fundet en kapacitiv Touch Sensor i skuffen, som kan opfylde behovet for at give en touch knap funktion til at starte processen. Der benyttes en AT42QT1010 fra Microchip, som er i en SOT23-6 pakke, som vist i Figur 1.



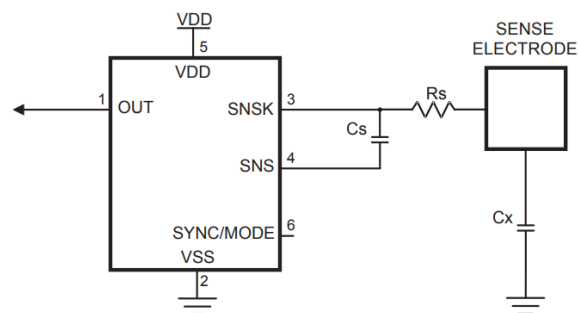
Figur 1: AT42QT1010 i SOT23-6 pakke, pinout

##### 3.1.1.2. Interface

Name	Pin	Type	Comments	If Unused, Connect To...
OUT	1	O	Output state	—
VSS	2	P	Supply ground	—
SNSK	3	I/O	Sense pin	Cs + Key
SNS	4	I/O	Sense pin	Cs
VDD	5	P	Power	—
SYNC	6	I	SYNC and Mode Input	Pin is either SYNC/Slow/Fast Mode, depending on logic level applied (see <a href="#">Section 3.1</a> )

Tabel 6: AT42QT1010 pin beskrivelse

### 3.1.1.3. Design for Implementation

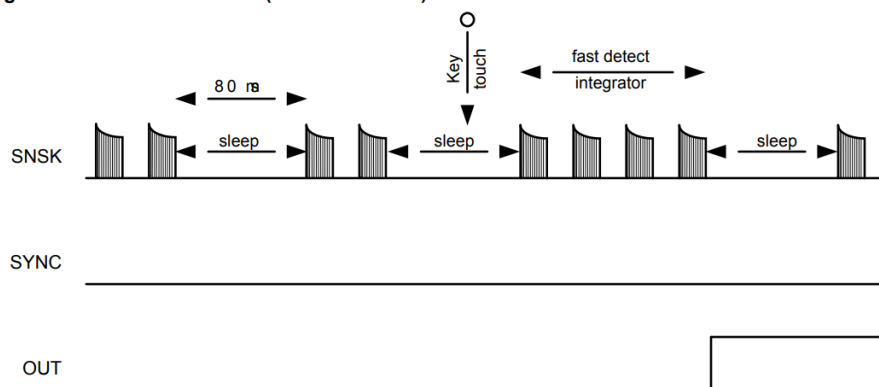


Note: A bypass capacitor should be tightly wired between Vdd and Vss and kept close to pin 5.

Figur 2: AT42QT1010 Basic kredsløbsdiagram fra [10]

Ref	Komponent Funktion	Valgt Komponent
Cs	Sense/Sample Capacitor	5-20pF (Måske blot en større mængde loddetin)
Cx	Load Capacitor	Forsøg med 100nF
Rs	ESD/EMC Resistor	
Sensor		
Vdd	+ Power pin (Voltage Drain)	+1V8 to 5V5 (sættes til 3V3)
Sync/Mode	Styrer hvor ofte output opdateres	Sættes i Low Power Mode for at spare mest muligt strøm. Dette medfører at opdateringen sker periodisk med 80ms. (Info: Fast mode er 1ms)

Figure 3-2. Low Power Mode (SYNC Held Low)



Figur 3: Opdatering af output ift. SYNC/Mode pin

For at spare tid, og for ikke at "gen-opfinde den dybe-tallerken", er valget faldet på at købe et færdigt modul som implementerer ovenstående. Se 2.2.1 for valgt komponent. Komponentens interface: GND, VDD og Output

## 3.2. RGB Color Sensor (S2)

### 3.2.1.1. Part Detaljer

Sensoren består af et CCA fra Adafruit, dette kan også kaldes et Modul. På modulet findes den egentlige sensor, som er en Photosensor, der har et RGB filter foran (indbygget). Denne sensor komponent er fra Texas Advanced Optoelectronic Solutions.

Part	Part nummer
Modul, RGB Sensor	Adafruit 1334
Komponent, RGB Sensor	TCS34725

Tabel 7: RGB Color Sensor, Overordnet Detaljer

### 3.2.1.2. Interface

X1 : 1x7 Pin Header

Terminal navn	Beskrivelse	Outline
LED	Intern Pull-up. Tilslut GND for at disable den indbyggede LED.	 <p>Slave addr: 0x29</p>
INT	Interrupt signaler, kan indstilles.	
SDA	I2C Data	
SCL	I2C Clock	
3V3	3V3 Output	
GND	PWR Return	
VIN	+3V3 til +5V	

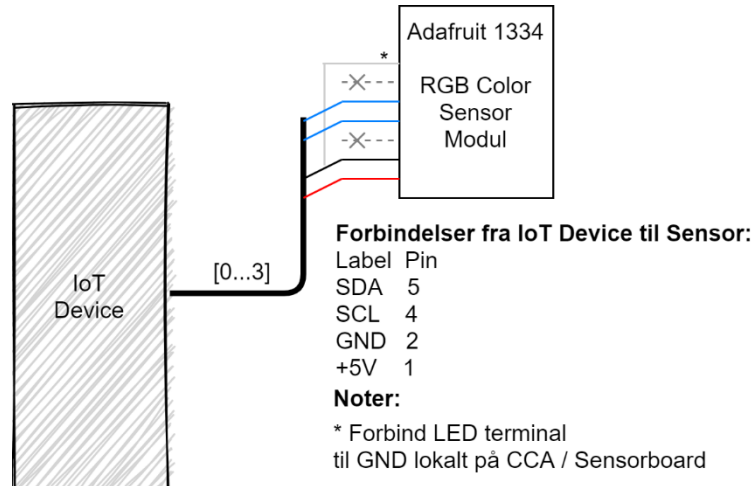
Tabel 8: RGB Color Sensor 1x7 Pin Header

Se tekniske detaljer i [8]

### 3.2.1.3. Design for Implementation

CCA, RGB Color Sensor modulet tilsluttes IoT Device (Particle Argon), overordnet som vist i blok-diagrammet herunder

Adafruit I2C slave adresse: 0x29



Figur 4: Design for implementering af S2

### 3.2.1.4. Verificering

Sensoren er verificeret med Analog Discovery 2 Protocol Analyzer. Og det er verificeret at den kan give forventelige digitale værdier for Clear, Red, Green og Blue (CRGB) under forskellige scenarier, hvor der er stimuleret med forskellige lyskilder, for at teste sensorens performance i at skelne farverne for forskellige farvespektre – herunder også testet med vaffeljernet Status LED, og det blev verificeret at den kan skelne "Heating state" (Orange LED lys) og "Ready state" (Grønt LED lys) fra hinanden.

Test	Overordnet Test Resultat
Se 4 Bilag, Test af RGB Sensor	RGB Sensor kan bruges

Tabel 9: Verificering af RGB Sensor, Resultat



3.3. Angle Sensor (S3)

3.3.1.1. Part Detaljer

Denne vinkelsensor er valgt for at detektere hvornår låget er åbent og lukket på vaffeljernet.

De vigtigste parametre er at den kan bruges til 5VDC, og at den er pålideligt ON/OFF når låget kan tilte 0°-180°.

Part	Part nummer
Komponent, Angle Sensor	TCSW520

Tabel 10: Angle Sensor, Overordnet Detaljer

3.3.1.2. Interface

Vinkelsensoren har to leads og disse er enten sluttet eller afbrudt internt.



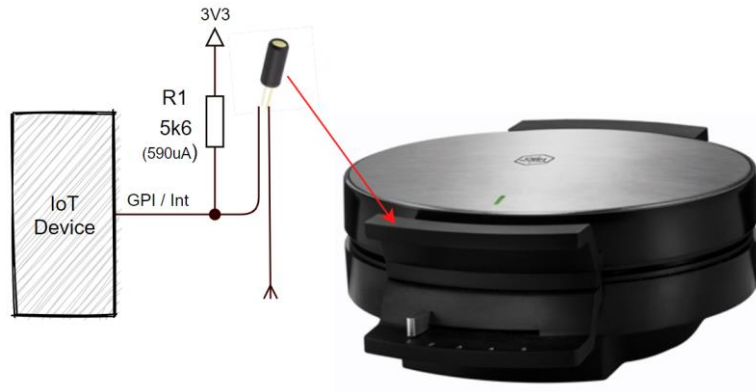
Figur 5: Angle Sensor, Billede

Den kan tåle op til 6V / ~5mA i ca. 2ms.

Terminal navn	Beskrivelse	Outline
Pin A	*	<p>Unit = mm</p>
Pin B	*	
*: ON: <10 ohm OFF: >10M ohm		

### 3.3.1.3. Design for Implementation

Når Låget er lukket vil vinkelsensoren trække 3V3 inputsignal til stel, og når låget er åbnet, vil den afbryde og IoT Device vil registrere at signalet på GPI er højt.



Figur 6: Design for implementering af Angle Sensor

## 3.1. Relay (A1)


### 3.1.1.1. Part Detaljer

Der er fundet et relæ som kan bruges til at switche 230VAC / 1000VA.

Dette relæ er kapabel til at switche 240VAC/10A, eller 1200VA jf. [11]

Part	Part nummer
Module, Relay	H16117

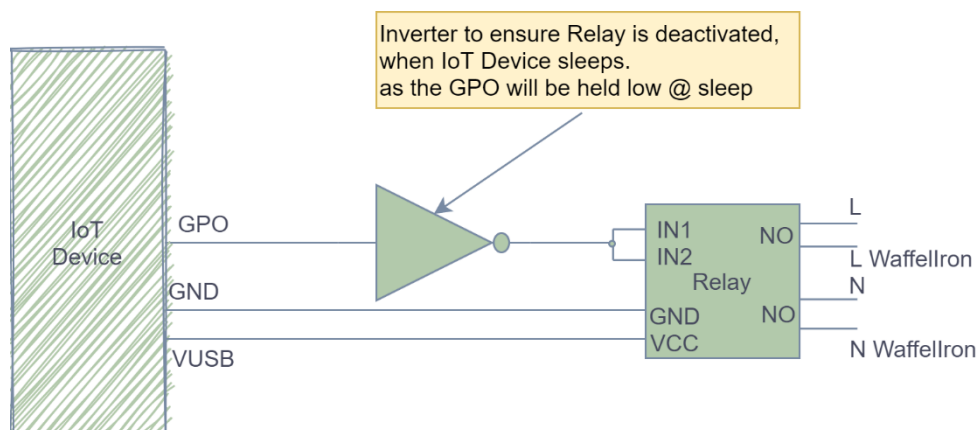
### 3.1.1.2. Interface

Terminal navn	Beskrivelse	Outline
<b>GND</b>	GND	 <p>2x SPST relay</p>
<b>VCC</b>	+5V	
<b>IN1</b>	0V = Relæ1 aktivt >3V = Relæ1 deaktivt	
<b>IN2</b>	0V = Relæ2 aktivt >3V = Relæ2 deaktivt	

Tabel 11: Relay pinout

### 3.1.1.3. Design for Implementation

For at sikre at relæet holdes inaktivt når IoT Device sover, skal der implementeres en inverter. Evt. brug et relæ eller Op-amp for at inverterer GPO, da relæet er aktivt ved 0V. Der skal desuden implementeres en pull-down modstand eller lignende, for at sikre at der går strøm nok gennem relæ-spolerne fra inaktiv->aktiv state se [11] (~70mA).



Figur 7: Design for Implementering af relæ

## 3.2. Aktuator Buzzer/Sounder (A2)

### 3.2.1.1. Part Detaljer

Denne sounder er tidligere benyttet i forbindelse med brugeroplevelses projekter i kurset "Brugeroplevelser i Indlejrede Systemer" på 4. semester (E4BIS).

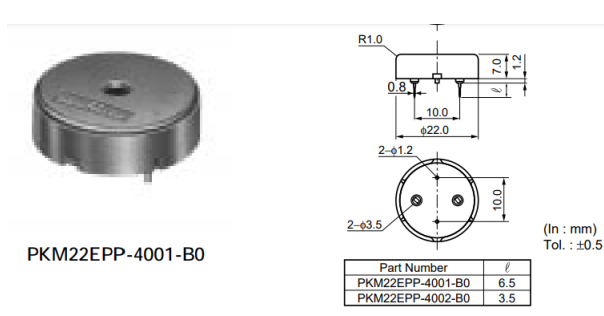
Derfor vælges denne sounder "as is".

Part	Part nummer
<b>Komponent, Buzzer/Sounder</b>	PKM22EPP-4001-B0

Tabel 12: Buzzer/Sounder Overordnet Detaljer

### 3.2.1.2. Interface

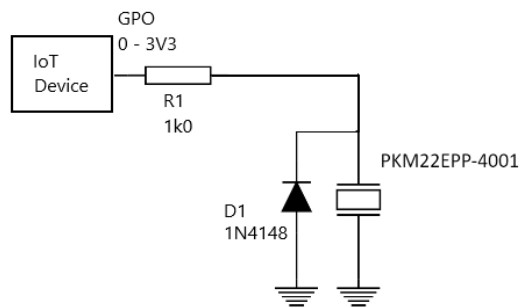
To ledere (Pin A og B) der er Ac-coupled, og derfor er der ingen polariseringsangivelse.

Terminal navn	Beskrivelse	Outline
Pin A	3V3pp eller Return	
Pin B	3V3pp eller Return	

Tabel 13: Buzzer/Sounder Pinout

### 3.2.1.3. Design for Implementation

Sounder skal stimuleres med 3Vpp / 2kHz sq.wave, gennem en sikkerheds modstand og diode, for at sikre IoT Device mod komponent defekts. Dette kredsløb er beskrevet i databladet [4] side 21



Figur 8: Design for Implementering af Sounder

### 3.3. Platform IoT Device (P1)

#### 3.3.1.1. Part Detaljer

Der benyttes en platform for systemet, der har trådløs internetforbindelse og 20 digital pins der kan konfigureres, se under features.

Part	Part nummer
Device, Particle Argon	ARGN-H

#### Features

Particle Argon har 20 pins, som kan bruges til GPIO, PWM, UART, SPI, I2C eller ADC-input. Dog er der begrænsning for hvor mange forskellige der kan benyttes, som det kan ses i Tabel 14

Peripheral Type	Qty	Input(I) / Output(O)
Digital	20	I/O
Analog (ADC)	6	I
UART	1	I/O
SPI	1	I/O
I2C	2	I/O
USB	1	I/O
PWM	8	O

Tabel 14: Digital Pins, Particle Argon

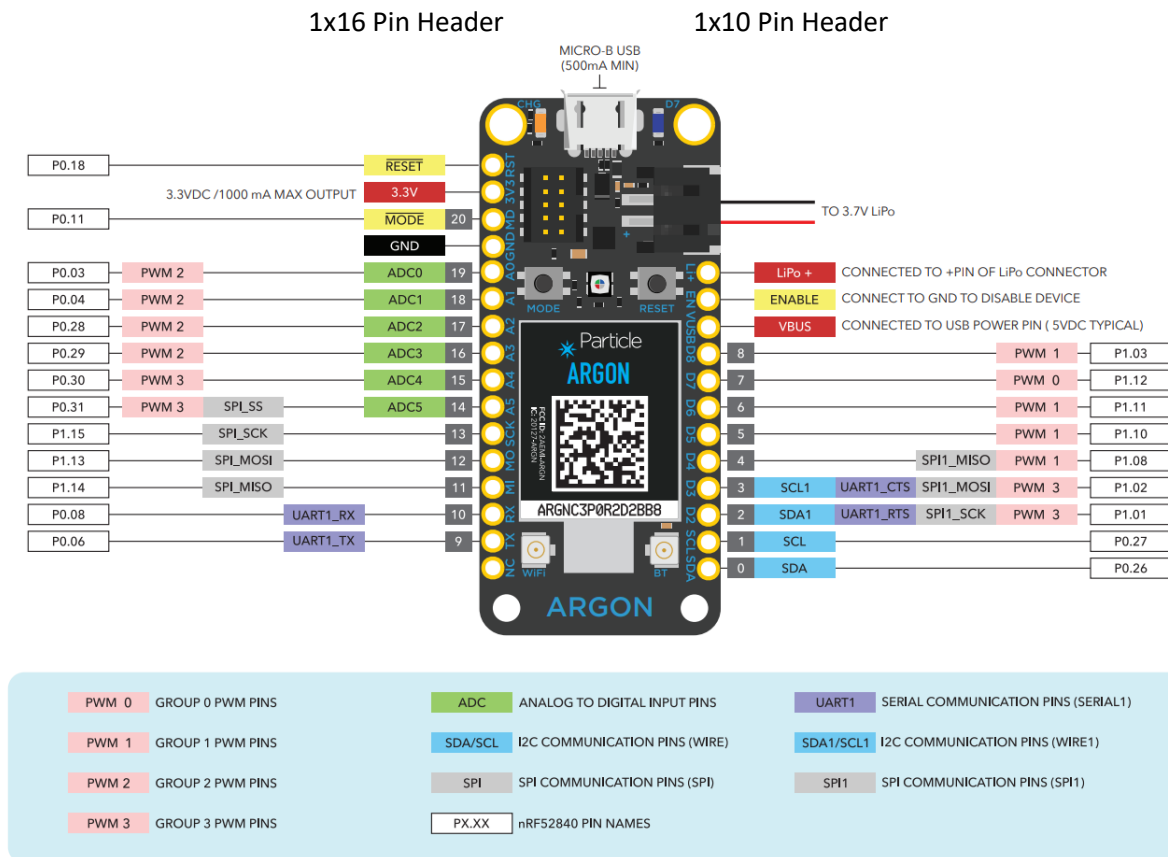
De digitale pins kan levere 3V3 / 2mA (Normalt) og 3V3 / 9mA som maksimum ved High drive setting, se Tabel 15

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input high voltage	$V_{IH}$		0.7*3.3	--	3.3	V
Input low voltage	$V_{IL}$		0		0.3*3.3	V
Current at GND+0.4 V, output set low, high drive	$I_{OL,HDL}$	$V_{3V3} \geq 2.7V$	6	10	15	mA
Current at $V_{3V3}$ -0.4 V, output set high, high drive	$I_{OH,HDH}$	$V_{3V3} \geq 2.7V$	6	9	14	mA
Current at GND+0.4 V, output set low, standard drive	$I_{OL,SD}$	$V_{3V3} \geq 2.7V$	1	2	4	mA
Current at $V_{3V3}$ -0.4 V, output set high, standard drive	$I_{OH,SD}$	$V_{3V3} \geq 2.7V$	1	2	4	mA
Pull-up resistance	$R_{PU}$		11	13	16	kΩ
Pull-down resistance	$R_{PD}$		11	13	16	kΩ

Tabel 15: Digital pin Settings/Kapabilitet

### 3.3.1.1. Interface

Particle Argon er outlined herunder i Figur 9



v1.0

Figur 9: Particle Argon, Pinout

### 3.3.1.2. Design for Implementation

Der laves et samlet diagram for hele systemet med Platformen som central unit.

Particle Argon forsynes med 5V på micro-USB konnektor.

### 3.4. Weather Service (W1)

#### 3.4.1.1. API Detaljer

Der er undersøgt forskellige API's på et overordnet plan.

Herunder er disse undersøgt, men ikke nærmere beskrevet i denne rapport:

- AccuWeather
- OpenWeatherMap
- Weatherbit
- Dark Sky
- Weather2020
- WeatherUnderground
- ClimaCell

Kravene til vejr API'et er at det skal være gratis at benytte, og det skal være forholdsvis nemt at implementere ift. at hente aktuel lokale vejrdato.

Valget er imidlertid faldet på at bruge OpenWeatherMap [9], da det er gratis og umiddelbart nemt at implementere.

Der er lavet en test af [9] i python, som kan ses i afsnit 5.

#### 3.4.1.2. Interface

GET Request til at hente lokal vejr-rapport:

`api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}`

Responset vil indeholde et JSON-skema, hvor der skal bruges værdierne som er highlighted med gult:

Raw JSON Respons skema for city_name = Aarhus		
{ 'coord': { 'lon': 10.21, 'lat': 56.16 }, 'weather': [ { 'id': 801, 'main': 'Clouds', 'description': 'few clouds', 'icon': '02n' } ], 'base': 'stations', 'main': { 'temp': 283.97, 'feels_like': 280.47, 'temp_min': 282.59, 'temp_max': 285.15, 'pressure': 1002, 'humidity': 87 }, 'visib': 10000, 'wind': { 'speed': 4.6, 'deg': 230 }, 'clouds': { 'all': 22 }, 'dt': 1603398937, 'sys': { 'type': 1, 'id': 1571, 'country': 'DK', 'rise': 1603346748, 'sunset': 1603382463 }, 'timezone': 7200, 'id': 2624652, 'name': 'Aarhus', 'cod': 200 }		

Herunder et skema med fokus på de værdier der skal hentes, se. Evt python script i afsnit 5 for hvordan det ser ud i kode:

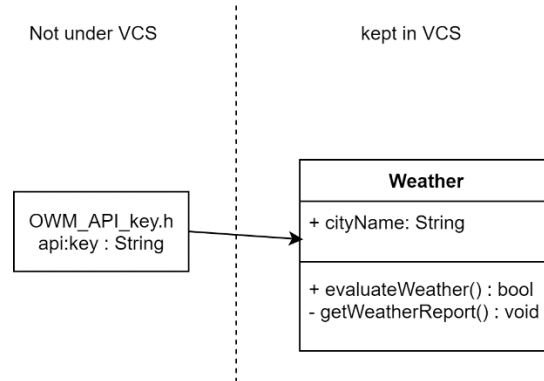
Key	Subkey	Value
Weather	Main	f.eks. "Clouds"
Main	Temp	Middeltemp i Kelvin
Main	Humidity	Luftfugtighed i procent

Tabel 16: JSON key-value par der hentes fra Vejrrapport

### 3.4.1.3. Design for Implementation

API Nøglen holdes i en separat header fil, som ikke inkluderes i GIT (Den tilføjes til gitignore).

Klassen Weather implementere API kaldet til OpenWeatherMap.



## 3.5. Sonos (W2)

### 3.5.1.1. API, Analyse

Der findes to kendte løsninger til at få kontrol over sine Sonos højtalere. Sonos API eller IFTTT (som er integreret ind i Sonos API).

#### 3.5.1.1.1. Løsning 1: Sonos API

Den første løsning er at benytte Sonos's eget RESTful API, som giver brugeren fuld kontrol, men som til gengæld er komplekst at bruge, da der bruges 3-lags authentication.

- 1) Først skal der laves en "Integration", som vil sige at der skal laves en egentlig Applikation som får sit eget ID.
- 2) Derefter skal brugeren logge ind via denne integration, som søger bekræftelse på identiteten hos Sonos.
- 3) Samtidigt søger brugeren authentication hos Sonos, og Sonos sender en bekræftelse til Integrationen.
- 4) Herefter dannes en tidsbegrænset nøgle (token), som skal holdes løbende opdateret. Herefter kan Sonos's Control API bruges til at styre sonos højtaleren.

Se dette forløb beskrevet herunder i Figur 10 som er taget fra [7]



# Authorize

Use the Sonos login service to authenticate with a Sonos user's household. The login service uses the OAuth 2.0 authorization framework and three-legged authentication. Three-legged authentication includes three participants, the user, your integration, and Sonos. These participants interact in three stages:

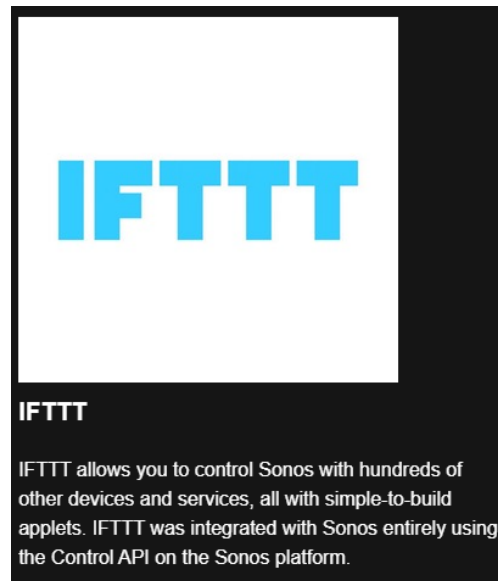
1. **Your integration** sends **the user** to the **Sonos** login service with your client credentials and parameters.
2. **The user** authenticates with **Sonos** and **Sonos** sends an authorization code back to **your integration**.
3. **Your integration** uses the authorization code to get an access token and a refresh token from **Sonos**.

Your integration then sends your access token to Sonos in every API call. When the access token expires, your integration can use the refresh token to get a new one. See the [Authorize Code Grant flow in the OAuth RFC](#) for more information.

*Figur 10: Snippet fra Sonos API vedr. Authentication*

#### 3.5.1.1.2. Løsning 2: IFTTT API (Integreret i Sonos API)

If This Then That er en anerkendt Web Service udbyder, som giver mulighed for at forbinde/integrere ellers uafhængige Web API's med hinanden. IFTTT er fuldt integreret i Sonos's Control API. Det vil sige at der ikke er en tung authentication proces som skal håndteres i dette projekt, men dette er indeholdt i IFTTT's integration – så når IFTTT og Sonos er blevet forbundet med den aktuelle Sonos Konto (manuel proces), så er der givet tilladelse til at IFTTT kan authenticate på vegne af den aktuelle sonos konto der blev forbundet.

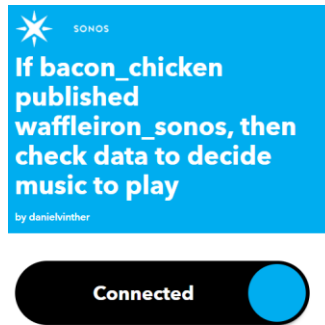


*Figur 11: IFTTT Integration med Sonos Control API*

### IFTTT Fremgangsmåde:

På IFTTT's hjemmeside oprettes en Applet, som er en Web Hook der indstilles til at subscribe til et event.

Med dette event modtages en payload (data streng), som eventet foretager valg ud fra. Den kan f.eks. sættes til at afspille et musiknummer på en bestemt højtaler i tilfælde af et event, som vist herunder, hvor particle Argon ved navn "bacon\_chicken" subscriber til eventet "waffleiron\_sonos" og beslutter ud fra data, hvilket nummer der skal afspilles.



Herefter vil højtaleren afspille det valgte nummer hver gang der sendes et event.






Der skal dog oprettes særskilte events for hver "data" mulighed, så ovenstående er en generisk label.

#### **3.5.1.2. Interface**

Det er valgt at benytte IFTTT Web API, som sættes op til at modtage et event. Når IFTTT modtager eventet behandles det med henblik på efterfølgende at interagere med Sonos API på vores vegne. Det vil sige IFTTT skal godkendes til at kunne autorisere med Sonos med brugeridentifikation.

#### **3.5.1.3. Design for Implementation**

Der implementeres en klasse, der som minimum indeholder en metode for at sende et event til Particle cloud, der er hooked til IFTTT/Sonos.

	Sonos
	const String _event
	const String[] _states
	bool _publishToCloudSuccess
	SonosPlay()

```
_event = "waffleiron_sonos"  
_states: "heating", "ready", "baking"
```


4. Bilag, Test af RGB Sensor

4.1. Verificer RGB Color Sensor med AD2 Protocol Analyzer

For at afprøve og vurdere data fra sensoren, og se hvorledes den performer inden den kan godkendes som RGB color sensor til projektet, forbindes den til Analog Discovery 2 som vist i

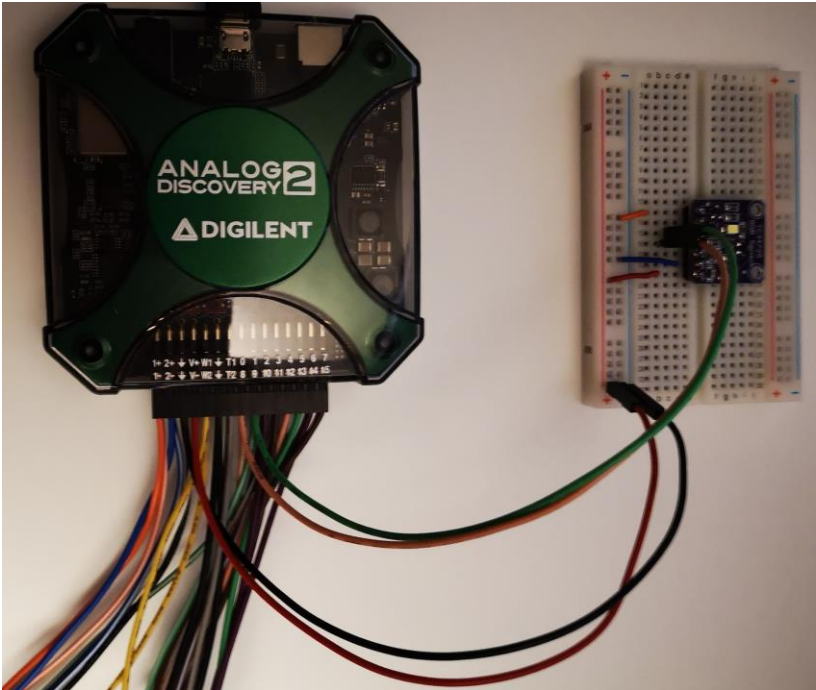
4.1.1. Test Setup

Forbind DUT til Analog Discovery 2 som vist i Tabel 17 og Figur 12.

Destination (Forbind til AD2)	Terminaler på DUT	Outline af DUT
GND	LED	
NC	INT	
DIO 1 (GRN)	SDA	
DIO 0 (PNK)	SCL	
NC	3V3	
GND (BLK)	GND	
V+ (5V) (RED)	VIN	

Tabel 17: RGB Color Sensor (DUT) forbindelser til AD2

Testopstilling, hvor DUT er forbundet til AD2, er vist herunder i Figur 12



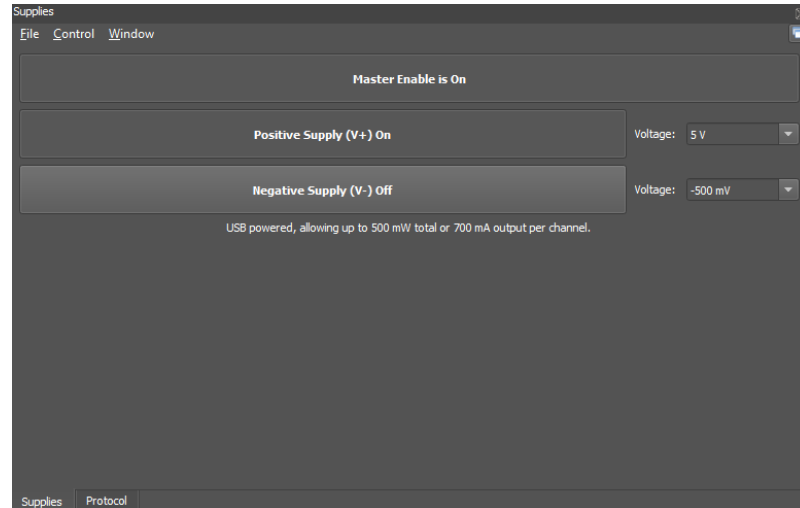
Figur 12: Testopstilling RGB Color Sensor (DUT)



#### 4.1.2. Indstilling af Waveforms (AD2)

Tilslut 5V til DUT:

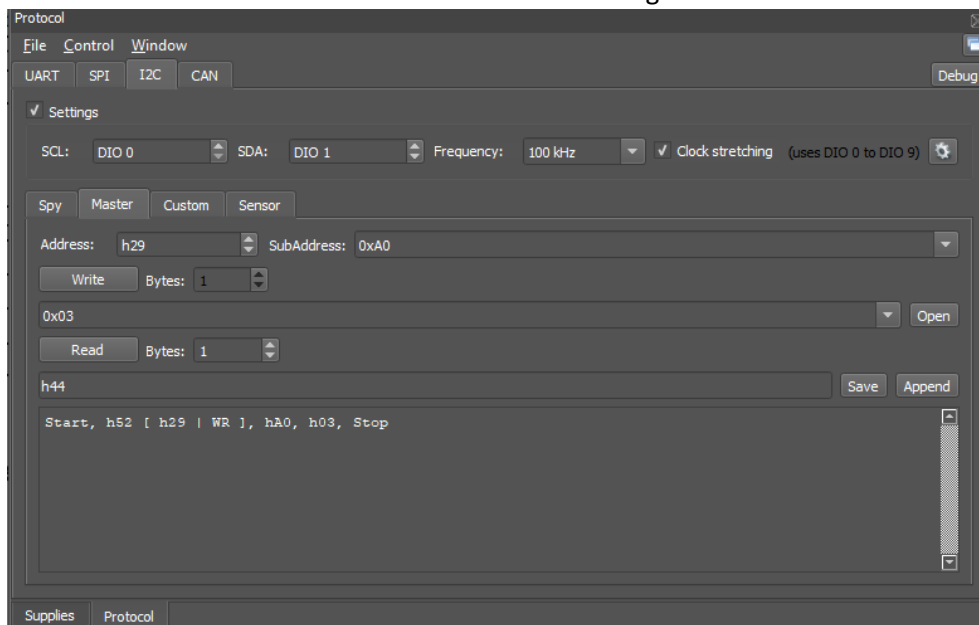
Indstil V+ til 5V og Tænd Positive Supply fra Supplies tab



Figur 13: Indstilling af +5V forsyning

Opsætning af RGB Color Sensor:

1. Åben Protocol
2. Vælg I2C
3. Set Address til 0x29 Se [3] p3 (TCS34725)
4. Set SubAddress til 0xA0 For at Bruge Command Reg. og auto-inc. Reg [3] p14
5. Set Write Data til 0x03 For at Enable RGBC og Power ON I Enable reg. [3] p15
6. Klik Write for at sende data
7. Observer at det sendte ser ud som vist i Figur 14



Figur 14: RGB Color Sensor (DUT) bliver aktiveret for kommunikation

Valider Opsætning:

Som et tjek for at der er skabt en fungerende I2C forbindelse, og som tjek for at Sensor er den rigtige, kan Device ID udlæses fra register 0x12. Det kan ses i Figur 15 at device ID forventes at være 0x44 da det er en TCS34725.

**ID Register (0x12)**

The ID Register provides the value for the part number. The ID register is a read-only register.

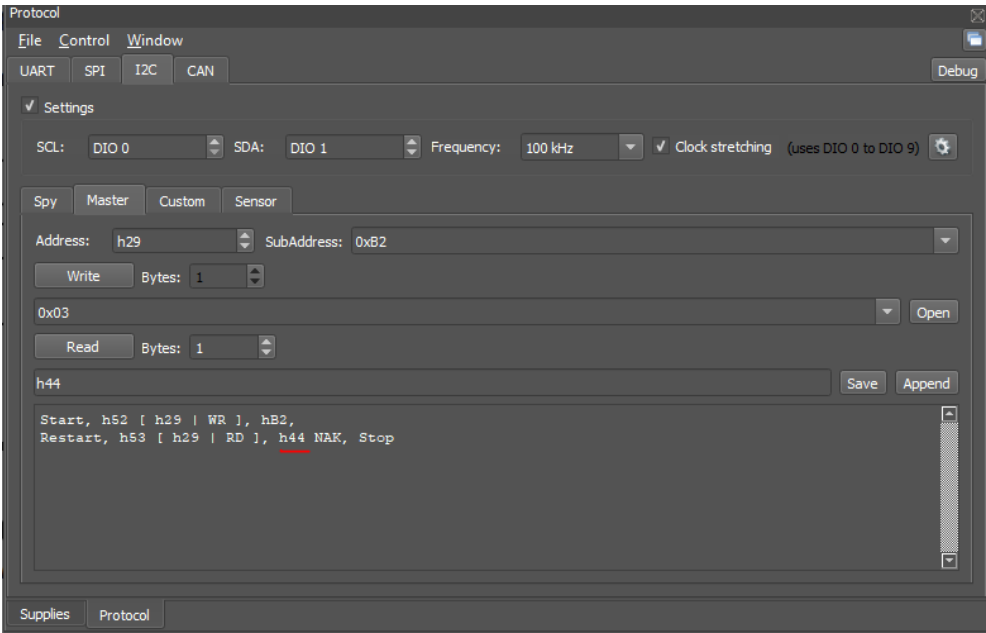
**Table 12. ID Register**

	7	6	5	4	3	2	1	0	
ID	<div style="border: 1px solid black; width: 100%; height: 15px; text-align: center;">ID</div>								Address 0x12
FIELD	BITS	DESCRIPTION							
ID	7:0	Part number identification							0x44 = TCS34721 and TCS34725
									0x4D = TCS34723 and TCS34727

Figur 15: Device ID Register (DUT)

**Fremgangsmåde:**

- 1. Ændre SubAddress til 0xB2      For at læse cmd reg + ID reg (0xA0 + 0x12 = 0xB2)
- 2. Klik Read for at læse ID Reg.
- 3. Observer at data der modtages, er 0x44 som vist i Figur 16



Figur 16: Modtaget Device ID fra DUT

### 4.1.3. Test

#### Udlæs RGB Data fra Sensor:

Der er i alt 8-registre af 1Byte, som kan udlæses for at få de digitale værdier, som den integrerede ADC har dannet ud fra det lysspekter der er opfanget af photo-dioden. Disse registre er vist i Figur 17 og kommer fra databladet [3] p19. Da det er indstillet i "Opsætning af RGB Color Sensor" at registre skal auto-inkrementere fra det den adresse der sendes, skal DUT blot have at vide hvor mange bytes der forventes modtaget, og derefter have angivet start adressen for første register – Dermed modtages data fra de efterfølgende registre automatisk.

#### **RGBC Channel Data Registers (0x14 – 0x1B)**

Clear, red, green, and blue data is stored as 16-bit values. To ensure the data is read correctly, a two-byte read I<sup>2</sup>C transaction should be used with a read word protocol bit set in the command register. With this operation, when the lower byte register is read, the upper eight bits are stored into a shadow register, which is read by a subsequent read to the upper byte. The upper register will read the correct value even if additional ADC integration cycles end between the reading of the lower and upper registers.

**Table 14. ADC Channel Data Registers**

REGISTER	ADDRESS	BITS	DESCRIPTION
CDATA	0x14	7:0	Clear data low byte
CDATAH	0x15	7:0	Clear data high byte
RDATA	0x16	7:0	Red data low byte
RDATAH	0x17	7:0	Red data high byte
GDATA	0x18	7:0	Green data low byte
GDATAH	0x19	7:0	Green data high byte
BDATA	0x1A	7:0	Blue data low byte
BDATAH	0x1B	7:0	Blue data high byte

*Figur 17: RGB Data Register*

#### Fremgangsmåde:

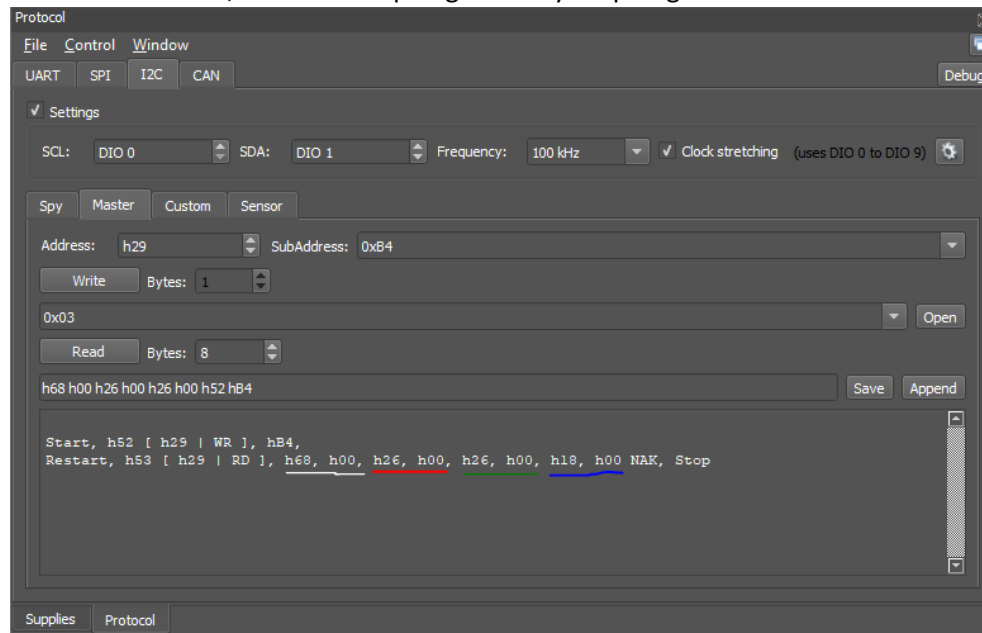
1. Ændre SubAddress til 0xB4      For at læse cmd reg + "start" reg (0xA0 + 0x14 = 0xB4)
2. Indstil Read Bytes til 8
3. Klik Read
4. Observer at der modtages data (Data fluktuerer for hvert read)



#### 4.1.3.1. TS1 - Mørkt Rum

##### Test Scenarie 1:

Data i "mørkt rum, kun bordlampe og skærmlys" opfanges af sensor:

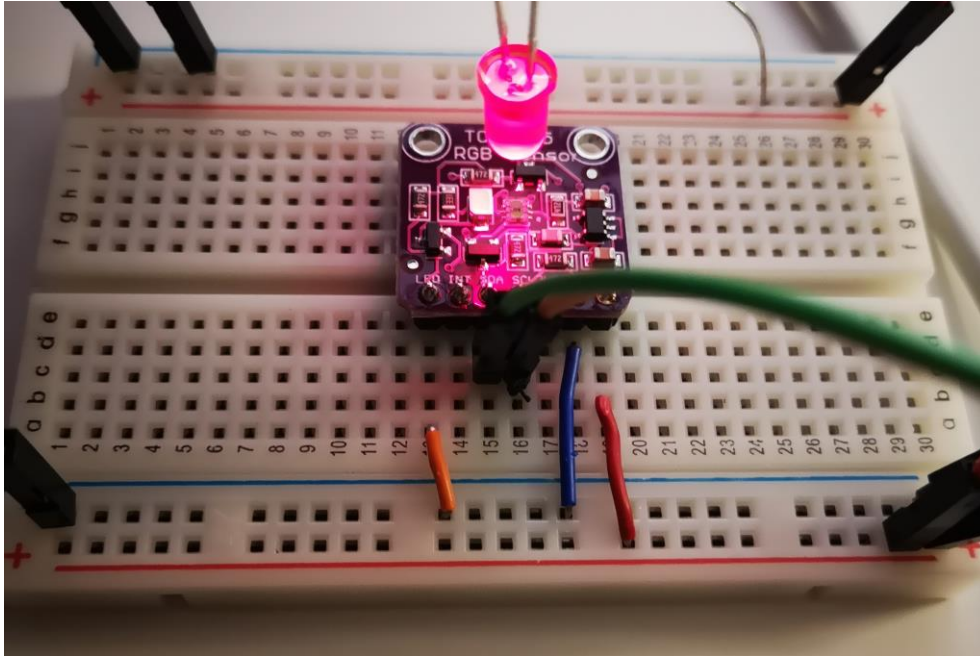


Figur 18: TS1 Data, RGBC Data fra DUT i mørkt rum

#### 4.1.3.2. TS2 – Rød LED

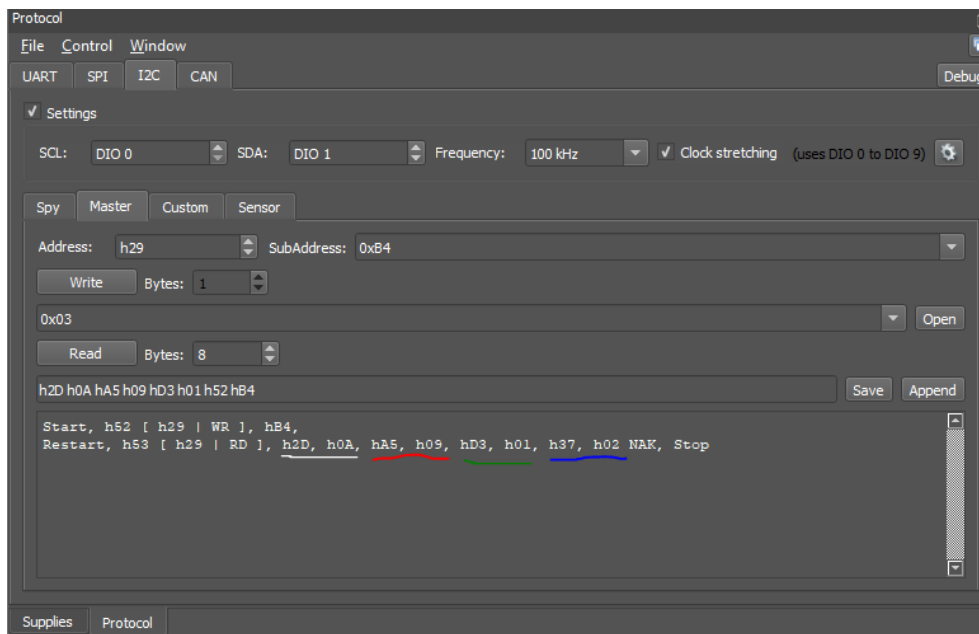
##### Test Scenarie 2, Rød LED:

Data med Rød LED der lyser direkte ind på sensor:



Figur 19: TS2 Test Setup, Rød LED lyser på DUT

Herunder er data fra DUT

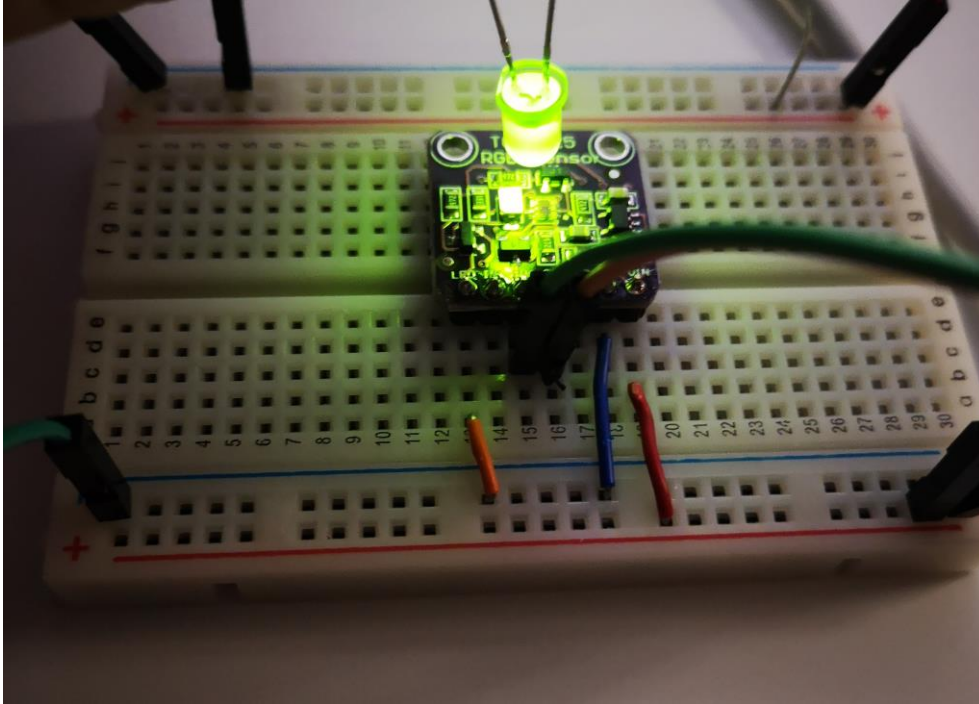


Figur 20: TS2 Data, Rød LED lyser ind på Sensor

#### 4.1.3.3. TS3 – Grøn LED

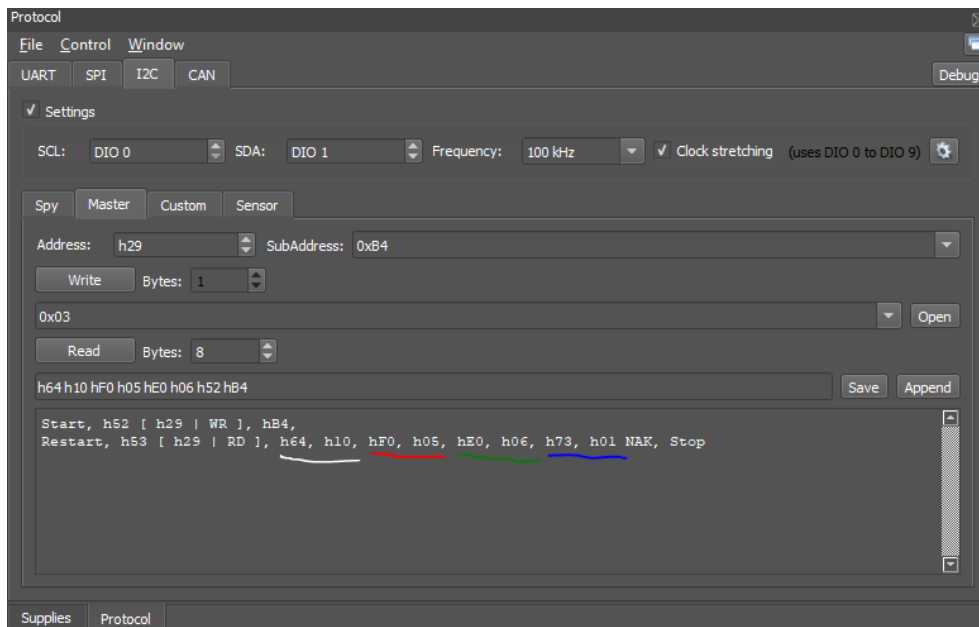
Test Scenarie 3, Grøn LED:

Data med Grøn LED der lyser direkte ind på sensor:



Figur 21: TS3 test setup, Grøn LED lyser på DUT

Herunder er data fra DUT for TS3



Figur 22: TS3 Data, Grøn LED lyser ind på Sensor

#### 4.1.3.4. TS4 – Vaffeljern, Heating State

##### Test Scenarie 4, Heating State:

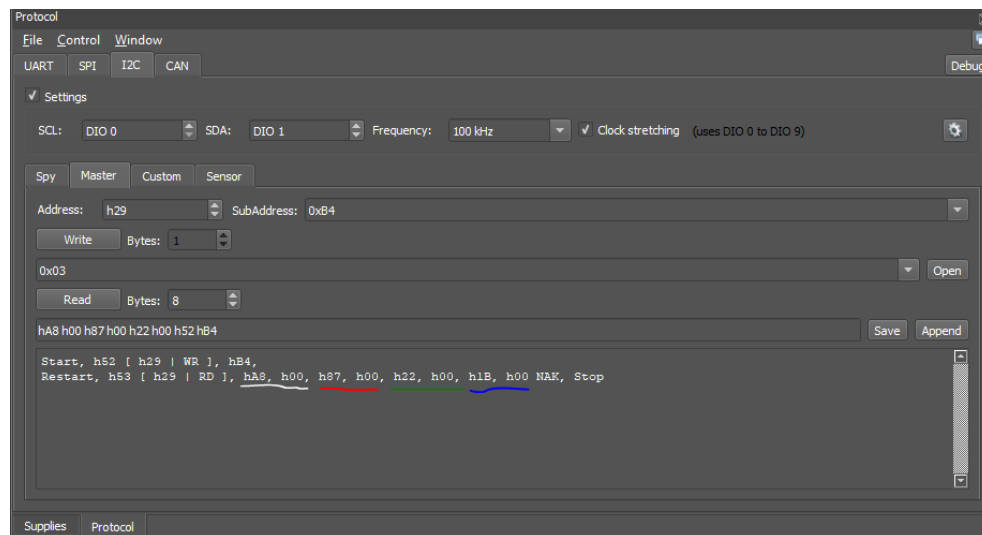
Sensor er klistret fast til vaffeljernet, og opfanger lyset fra status LED-indikatoren.

I dette scenarie er vaffeljernet ved at varme op, og derfor udstråles der Orange lys fra den indbyggede status LED. (Heating State)



Figur 23: Sensor opfanger Orange lys fra Vaffeljernet (Heating state)

Herunder er data fra TS4



Figur 24: TS4 Data, Heating state på Vaffeljernet

#### 4.1.3.5. TS5 – Vaffeljern, Ready State

##### Test Scenarie 5, Ready State:

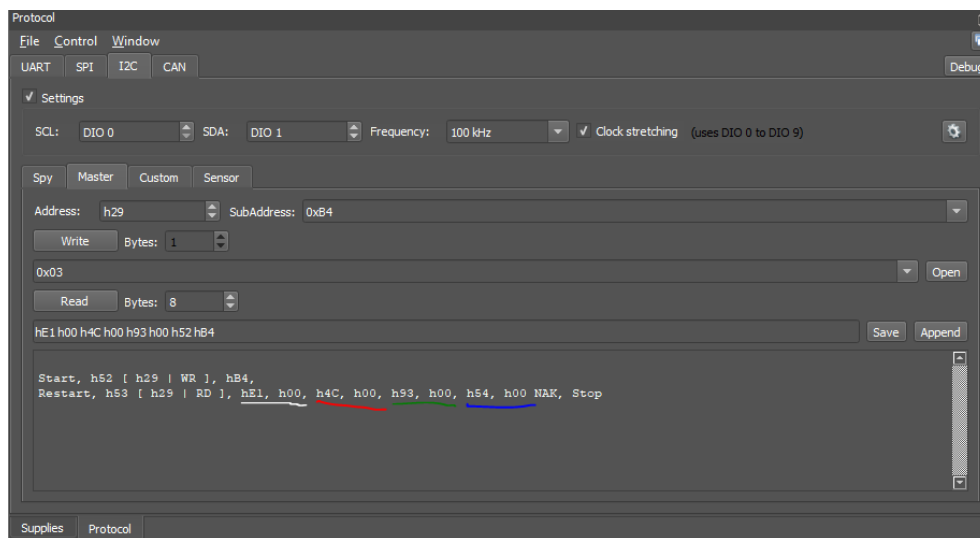
Sensor er klistret fast til vaffeljernet, og opfanger lyset fra status LED-indikatoren.

I dette scenarie er vaffeljernet blevet varmt, og derfor udstråles der Grønt lys fra den indbyggede status LED. (Ready State)



Figur 25: Sensor opfanger lys fra Vaffeljernet, Ready State

Herunder er data fra TS5



Figur 26: TS5 Data, Ready state på Vaffeljernet

#### 4.1.4. Test Resultater

For de Test scenarie 1-3 sammenlignes de modtagne data fra DUT. Det forventes at sensoren (DUT), vil registre rødt lys bedre i TS2, dvs. den samlede værdi for R Low og R High forventes at være signifikant højest i TS2. Der er samme forventninger til at data for grønt lys er højest i TS3.

Rå data i hex-værdier:

ID	Test Navn	C Low	C High	R Low	R High	G Low	G High	B Low	B High
TS1	Mørkt Rum	68	00	26	00	26	00	18	00
TS2	Rød LED	2D	0A	A5	09	D3	01	37	02
TS3	Grøn LED	64	10	F0	05	E0	06	73	01
TS4	Heating State	A8	00	87	00	22	00	1B	00
TS5	Ready State	E1	00	4C	00	93	00	54	00

Tabel 18: Data fra DUT for TS1-TS5 (i hex)

Omregning til Decimal-værdier:

Tal fra Tabel 18 Omregnes til decimaltal og indsættes i Tabel 19.

Databredde er 2 bytes, og MSB er High Byte. Decimalværdi beregnes af det samlede 16-bit tal.

f.eks. Hvis High = 09 og Low = A5:

$$(09A5)_{16} = (0 * 16^3) + (9 * 16^2) + (10 * 16^1) + 5 * 16^0) = (2469)_{10}$$

Data omregnet til Decimal-værdier:

ID	Test Navn	Clear	Red	Green	Blue
TS1	Mørkt Rum	104	38	38	24
TS2	Rød LED	2605	2469	467	567
TS3	Grøn LED	4196	1520	1760	371
TS4	Heating State	168	135	34	27
TS5	Ready State	225	76	147	84

Tabel 19: Samlet resultat i Decimal

##### 4.1.1. Konklusion

RGB Color sensor kan skelne mellem Heating State og Ready State, da det ses at når der ses bort fra værdierne for "Clear", ses det at værdierne for "Red" peaker når vaffeljernet er i Heating State, og værdierne for "Green" peaker når vaffeljernet er i Ready State.

##### 4.1.2. Diskussion

Det var forventet at forskellen mellem værdierne for "Green" og "Red" ville være større i hhv. Heating State og Ready State. Der er formentligt en justering af gain af de enkelte værdier der kan justeres. Der er et afsnit omkring det i databladet [3] page 18, som viser at der kan sættes et Gain på op til 60x. Dette skal undersøges ifm. Implementering, om det er nødvendigt at indføre.

0000243101 [app] INFO: Sun Oct 11 18:27:23 2020 Reading Number: 42  
0000243103 [app] INFO: Sun Oct 11 18:27:23 2020 Color is 1  
0000243104 [app] INFO: Sun Oct 11 18:27:23 2020 Colors: Red=57 Green=18 Diff=39  
0000248107 [app] INFO: Sun Oct 11 18:27:28 2020 Reading Number: 43  
0000248108 [app] INFO: Sun Oct 11 18:27:28 2020 Color is 0  
0000248109 [app] INFO: Sun Oct 11 18:27:28 2020 Colors: Red=56 Green=62 Diff=-6  
0000253112 [app] INFO: Sun Oct 11 18:27:33 2020 Reading Number: 44  
0000253113 [app] INFO: Sun Oct 11 18:27:33 2020 Color is 2  
0000253114 [app] INFO: Sun Oct 11 18:27:33 2020 Colors: Red=55 Green=116 Diff=-61  
0000258117 [app] INFO: Sun Oct 11 18:27:38 2020 Reading Number: 45  
0000258119 [app] INFO: Sun Oct 11 18:27:38 2020 Color is 2  
0000258120 [app] INFO: Sun Oct 11 18:27:38 2020 Colors: Red=56 Green=116 Diff=-60  
0000263123 [app] INFO: Sun Oct 11 18:27:43 2020 Reading Number: 46  
0000263124 [app] INFO: Sun Oct 11 18:27:43 2020 Color is 2  
0000263126 [app] INFO: Sun Oct 11 18:27:43 2020 Colors: Red=56 Green=117 Diff=-61  
0000268128 [app] INFO: Sun Oct 11 18:27:48 2020 Reading Number: 47  
0000268130 [app] INFO: Sun Oct 11 18:27:48 2020 Color is 2  
0000268131 [app] INFO: Sun Oct 11 18:27:48 2020 Colors: Red=56 Green=117 Diff=-61  
0000273134 [app] INFO: Sun Oct 11 18:27:53 2020 Reading Number: 48  
0000273136 [app] INFO: Sun Oct 11 18:27:53 2020 Color is 2  
0000273137 [app] INFO: Sun Oct 11 18:27:53 2020 Colors: Red=56 Green=117 Diff=-61  
0000278141 [app] INFO: Sun Oct 11 18:27:58 2020 Reading Number: 49  
0000278142 [app] INFO: Sun Oct 11 18:27:58 2020 Color is 2  
0000278143 [app] INFO: Sun Oct 11 18:27:58 2020 Colors: Red=56 Green=117 Diff=-61  
0000283146 [app] INFO: Sun Oct 11 18:28:03 2020 Reading Number: 50  
0000283148 [app] INFO: Sun Oct 11 18:28:03 2020 Color is 2  
0000283149 [app] INFO: Sun Oct 11 18:28:03 2020 Colors: Red=56 Green=117 Diff=-61  
0000288153 [app] INFO: Sun Oct 11 18:28:08 2020 Reading Number: 51  
0000288155 [app] INFO: Sun Oct 11 18:28:08 2020 Color is 2  
0000288157 [app] INFO: Sun Oct 11 18:28:08 2020 Colors: Red=56 Green=117 Diff=-61  
0000293160 [app] INFO: Sun Oct 11 18:28:13 2020 Reading Number: 52  
0000293162 [app] INFO: Sun Oct 11 18:28:13 2020 Color is 2  
0000293163 [app] INFO: Sun Oct 11 18:28:13 2020 Colors: Red=56 Green=117 Diff=-61  
0000298166 [app] INFO: Sun Oct 11 18:28:18 2020 Reading Number: 53  
0000298168 [app] INFO: Sun Oct 11 18:28:18 2020 Color is 0  
0000298170 [app] INFO: Sun Oct 11 18:28:18 2020 Colors: Red=1 Green=1 Diff=0

## 5. Bilag, Test af Vejr API (OpenWeatherMap)

For at teste API'et, inden der skrives c/c++ kode, er der lavet et Python Script, der henter vejr-rapport for Aarhus, og viser hvad vejret kategoriseret som, hvad temperaturen og luftfugtigheden er.

```
#!/usr/env python
# -*- coding: utf-8 -*-
"""
    Author:          Daniel K. Vinther Wolf
    Created:         22-10-2020
    Description:     Use OpenWeatherMap to Retrieve local weather data
    Dependencies:    API Key from OpenWeatherMap (Account)
    Pre-Requisites: API Key shall be stored in env.var (OPENWEATHERKEY=key)
"""

__author__ = "Daniel Korsgaard Vinther Wolf"
__version__ = "1.0.0"

import os
import requests

# Input:
city_name = "Aarhus"
api_key = os.environ['OPENWEATHERKEY']

# Use Open Weather Map API to get report for city_name
...

Retrieve local weather report (GET String format):
api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
...

# Use JSON format:
url = "https://api.openweathermap.org/data/2.5/weather"
data = {"q": city_name, "appid": api_key}

def convert_to_celcius(temp_K):
    return round(temp_K-273.15)
```



```

if __name__ == "__main__":
    print("This will send a request to open weather map "
          "and request weather data for city name: "+city_name+"\n")

    # Send request and check if response is OK
    response = requests.get(url, data)

    if response.status_code == 200:
        weather_report = response.json()
    else:
        print("Failed with response code:", response.status_code)

    # Collect "Weather.main", "main.temp", "main.humidity"
    weather_headline = weather_report['weather'][0]['main']

    temperature = weather_report['main']['temp']
    temperature = convert_to_celcius(temperature)

    humidity = weather_report['main']['humidity']

    # show weather report findings:
    print("Weather Report for", city_name, ":\n")
    print("Headline:", weather_headline,
          "\nTemperature [C]: ", temperature,
          "\nHumidity [%]: ", humidity)

```

Eksempel på kørsel af scriptet:

```

danie@DKVWPC MINGW64 /c/Users/danie/Dropbox/E5/IOT/projects/MiotyWaffles/MiotyWaffles/miotywaffles (master)
$ "C:/Program Files/Python37/python.exe" "c:/Users/danie/Dropbox/E5/IOT/projects/MiotyWaffles/MiotyWaffles/miotywaffles/doc/Bilag/02 Udv
ælg
else af Komponenter(SW_og_HW)/01 Info om Valgte Komponenter/W1 - Weather/openweathermap_check.py"
This will send a request to open weather map and request weather data for city name: Aarhus

Weather Report for Aarhus :
Rep. {
  Headline: Clouds
  Temperature [C]: 11
  Humidity [%]: 81
}
danie@DKVWPC MINGW64 /c/Users/danie/Dropbox/E5/IOT/projects/MiotyWaffles/MiotyWaffles/miotywaffles (master)

```

Der udskrives overordnet vejr, temperatur og luftfugtighed.