Embeded ex03 - Project Pedro

# Project Pedro

Hele startet med besøk hos Microsoft der Pedro Dias (technical evangelist) demonstrerte “Benny”, som er en plante som kan vanne seg selv og justere lys etter behov. Komisk nok, ble jeg insperert til å lage en temperaturlogger for kjøleskapet i restauranten min, og derfor ble prosjektet kalt for Pedro. Målet er å erstatte gjeldende arbeidsrutine om å hele tiden til et spesielt tidspunkt på dagen, hvor daglig leder må fylle ut et skjema med pen og papir på hva temperaturen er i kjøleskapet. Hvorfor ikke erstatte dette med en Particle photon med DHT11 sensor som kan måle og gi output på hva temperaturen er?

I starten hadde jeg planer om å koble opp flere komponenter, men fant ut etterhvert at det var unødvendig. Gjennom powerBI, kan man filtrere temperatur verdien etter tall, som vil si at hvis jeg vil ha en egen graf for bare temperatur under 10 grader så vil jeg kunne få visualisert bare det. Dette prosjektet sees på som bare en avsender av sensor data og ikke noe mer. Selve produktet er hele skytjenesten med web og web-api.

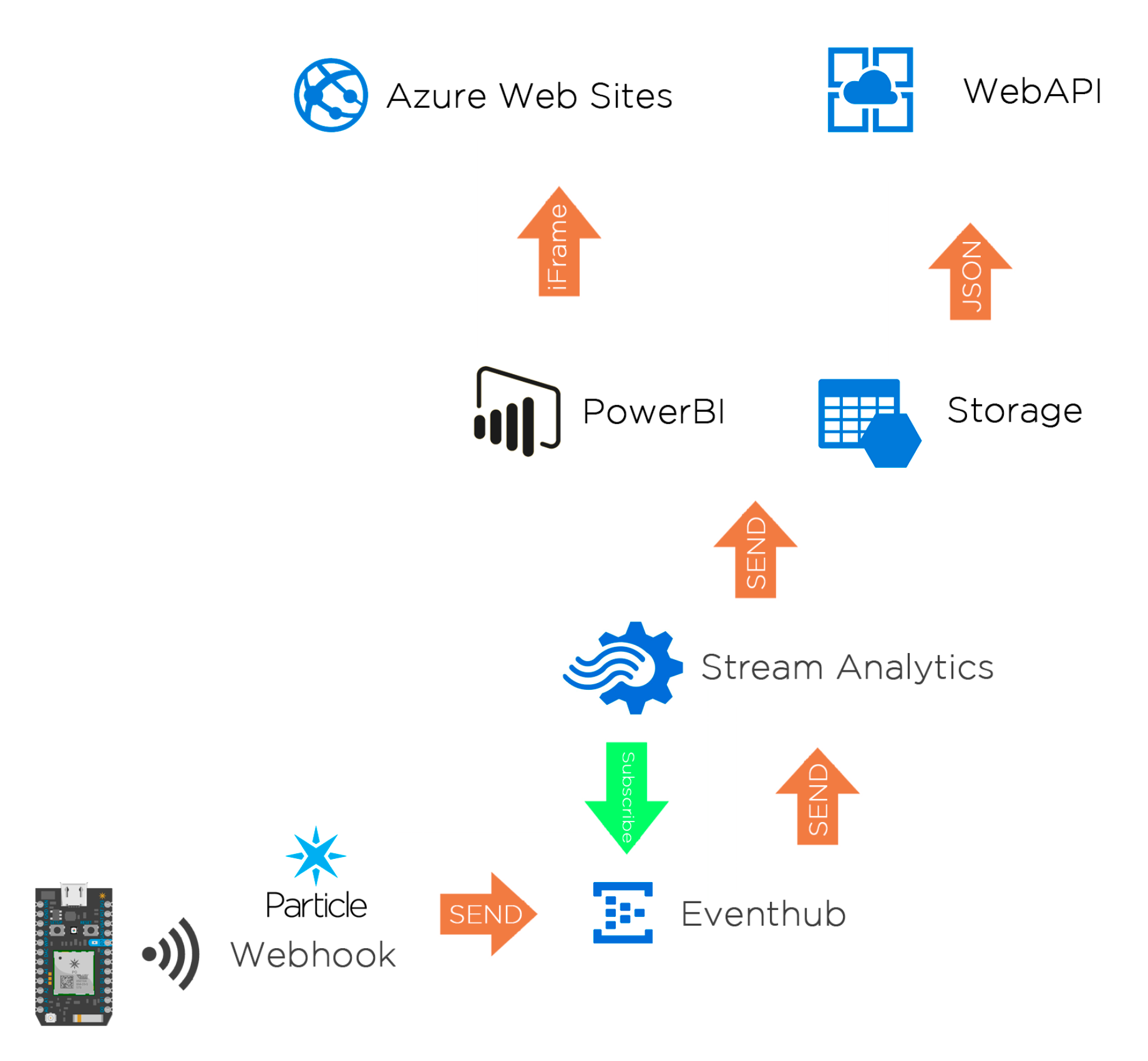
### *Namespace: pptlbh = Particle Photon Temperature Logger Benny Hoang*

Repo: https://github.com/BennyHoang/TemperatureLogger

# Utstyr

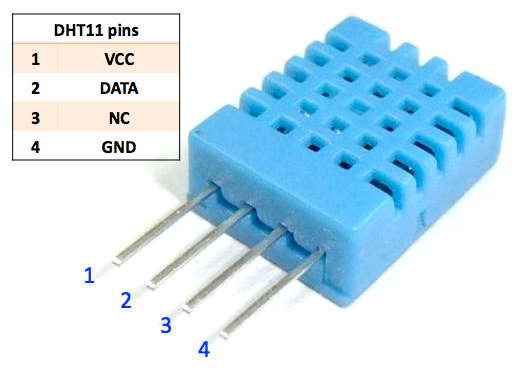
* Particle Photon
* DHT11 Sensor
* 10k Ω resistor
* Breadboard
* Wires
* Microsoft Azure Account
* Office 365 Account for PowerBI

# Arkitektur

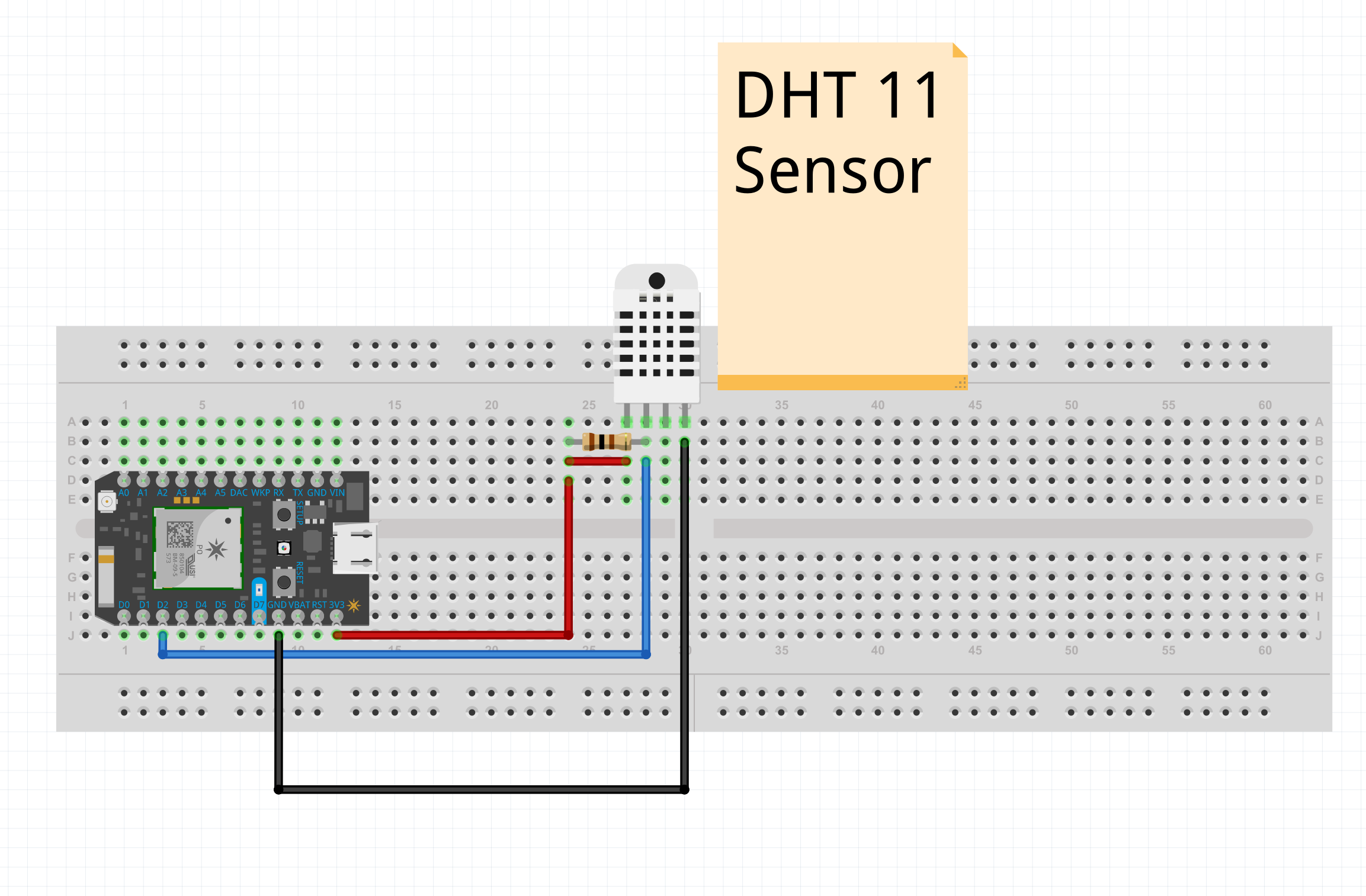


**Particle Photon**

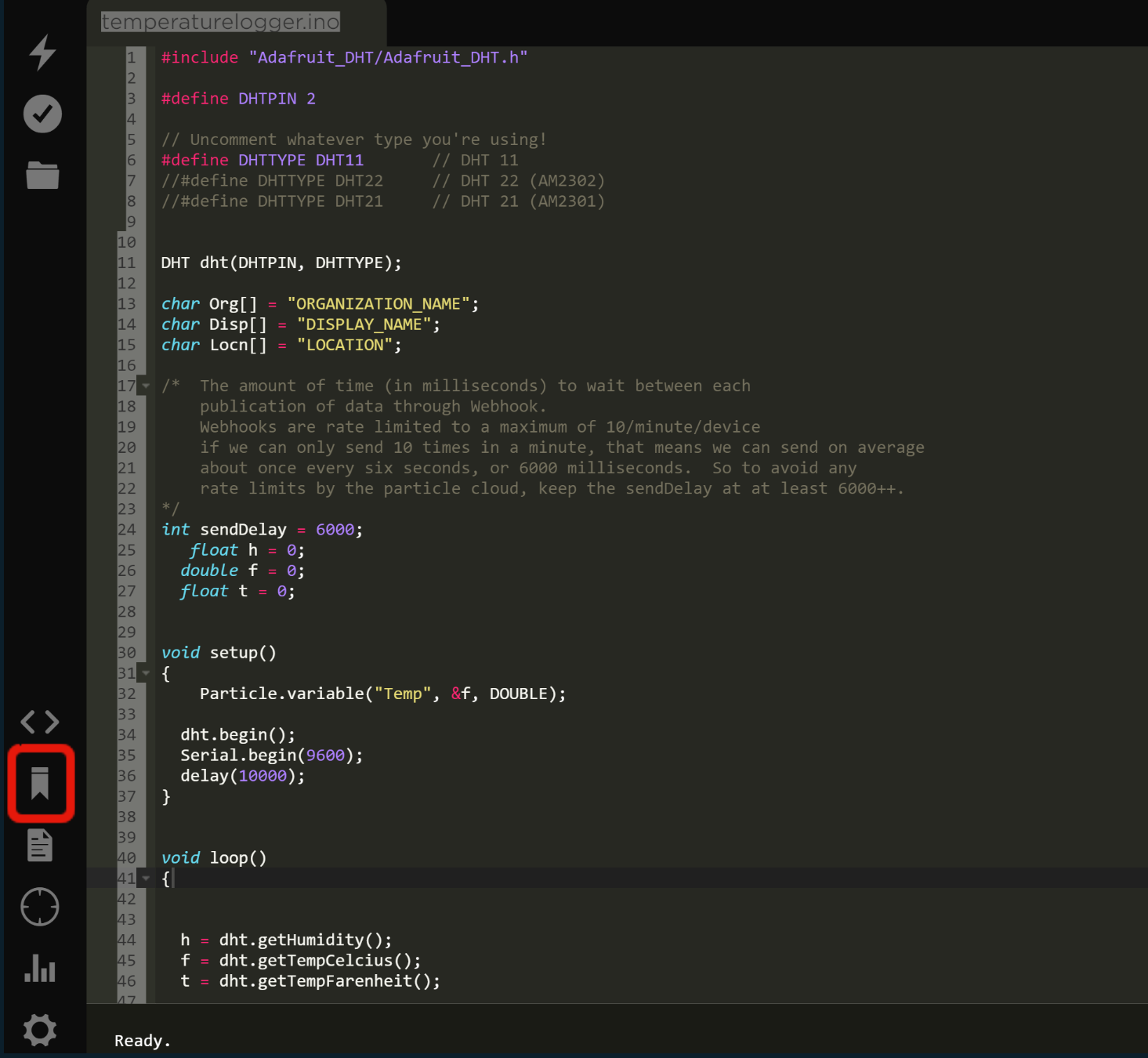
Hvordan Particle Photon er koblet til er ikke så forskjellig fra Arduino. En liten ting man bør være oppmerskom på er at Particle Photon bare 3.3V og ikke 5V. Bør kanskje være ekstra forsiktig her når du kobler til strøm slik at ingenting ryker. For temperatur/fuktmåleren (DHT11) betyr koblingene dette:



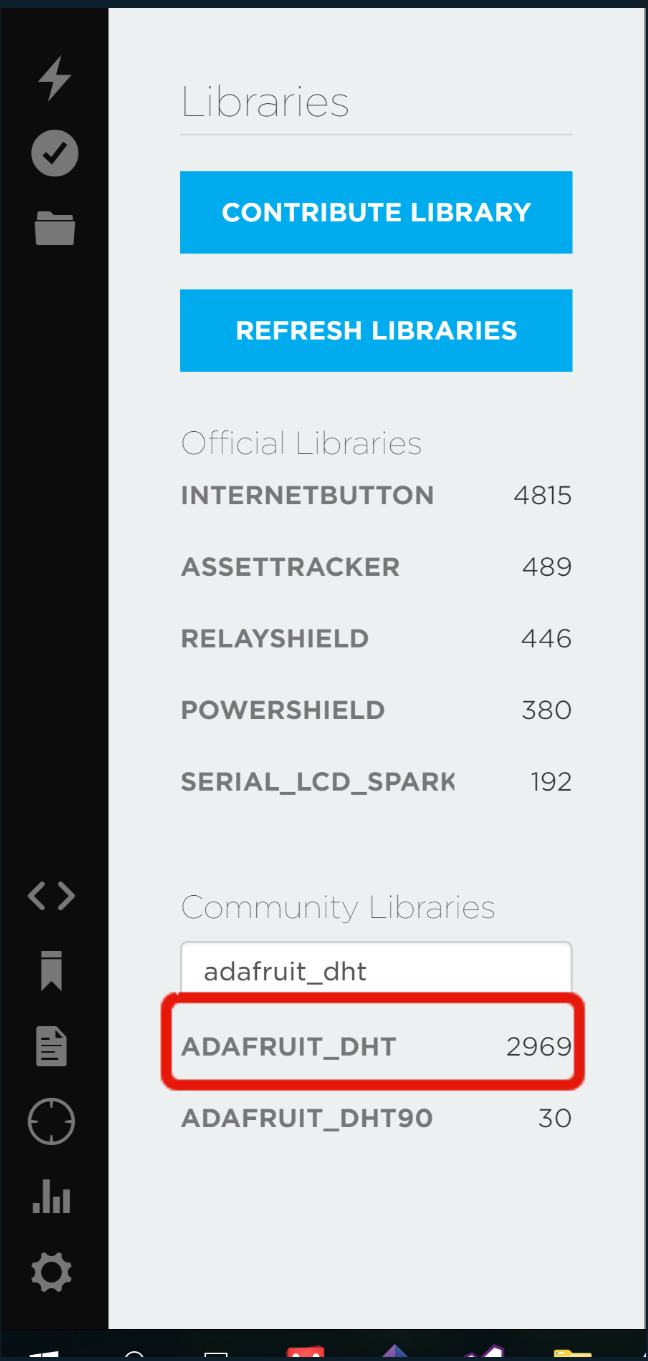
Koblingen til Particle Photon vil da se slik ut:



For å bruke DHT11 Sensoren må du ha biblioteket inkludert i prosjektet. Dette gjør du ved å klikke på ikonet som er merket i *build.particle.io.*



Søk etter “Adafruit\_DHT” og velg øverste bibliotek.



# Webhook

Ved å bruke particle sine webhooks, kan vi lett sende data fra Particle enheter til andre apper og services som eksisterer i nettet. I vårt tilfelle blir dette en måte å koble seg mot Azure og ta i bruk alle “out of the box” løsningene som allerede eksisterer der. På denne måten kan vi lagre sensordataene og visualisere dem i form av fine grafer eller brukes til machine learning for prediktativ analyse (Azure machine learning). Her er det helt opp til bruksområdet, dette viser bare hvor mange dører vi åpner ved å kunne bruke webhooks til eksterne apper.

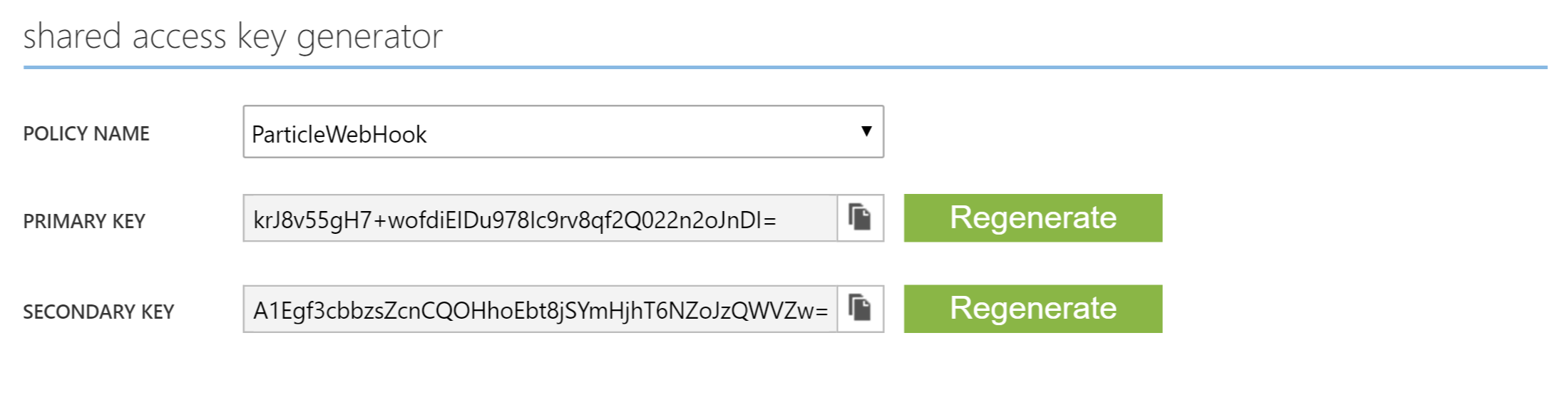
Webhooken er i JSON format med UTF-8 Char-encoding som gjør det veldig lightweight å sende frem og tilbake gjennom nettet, og lett for mennesket å lese.



Når vi bruker particle.publish(), kaller vi på “PublishToEventHub” for å sende JSON payload til Azure Servicebus. For å opprette denne kan du enten gå gjennom integrations i dashboard.particle.io eller bruke CLI.

# Azure

Skal ikke gå i dybden om hvordan Azure fungerer da det allerede finnes dokumentasjon på hvordan man lager ulike apps ute. Når Servicebus namespace er opprettet og Eventhuben har fått sine shared access policy, kan man finne det man trenger til å lime inn i sin webhook slik at de to appene får snakket med hverandre.

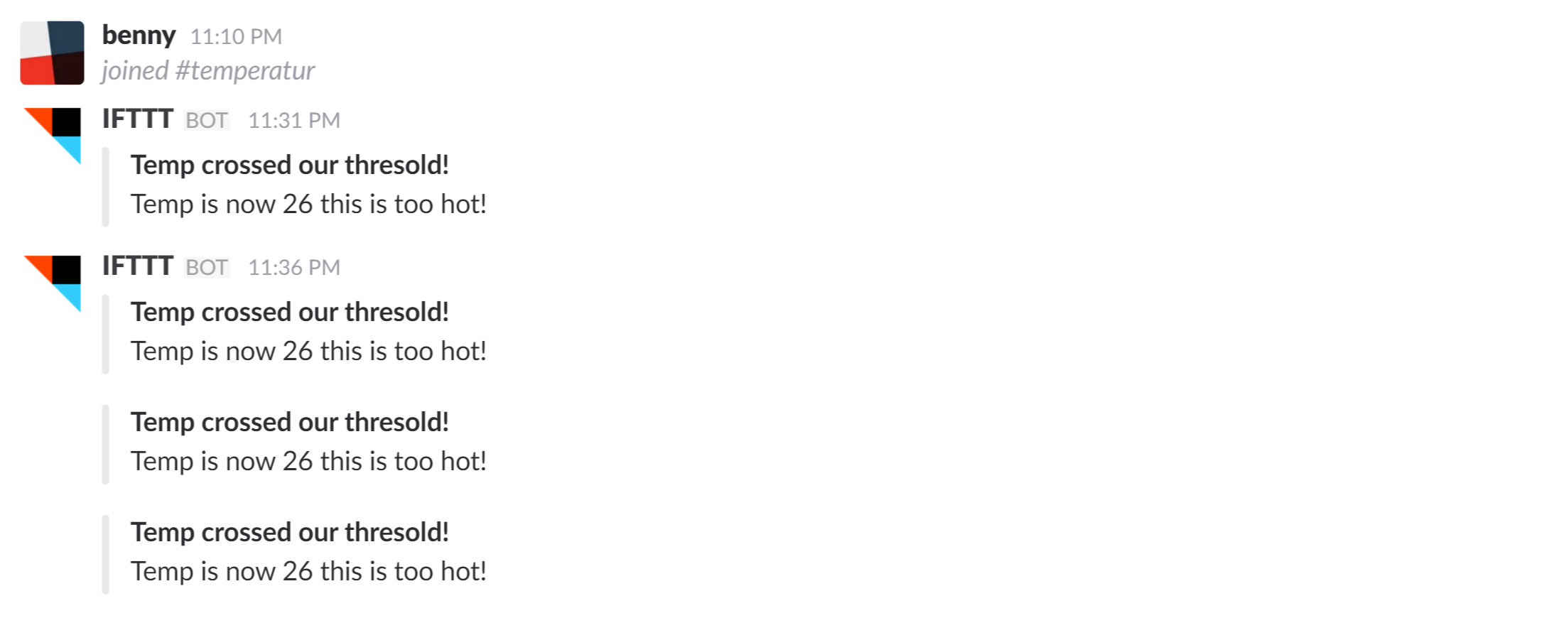


EventHuben gjør ikke noe annet enn å sende meldingene. I dette stadiet blir ikke temperaturen og andre data sendt inn lagret. Ved å bruke StreamAnalytics får jeg hele tiden hentet inn meldingene som blir sendt. StreamAnalytics kan da query input dataene og output i form av Database, PowerBI, CVC for excel osv. I mitt tilfelle blir det bruk av Azure Storage og PowerBI. Det brukes vanlig SQL for å output data (se StreamAnalytics query)

På denne måten kan jeg sende den dataen til PowerBI for å visualisere dataen i form av grafer. I PowerBI kan man embede grafene sine slik at man kan vise dem i nettsiden sin også.

# IFTTT





Denne blir brukt til å sende varsler til Slack kanalen når temperaturen er for høy. Hvor grensen går for temperatur kan lett justeres i IFTTT. Dette kan være feks hvis temperaturen er høyere enn 4 grader (samme gjelder grafene fra PowerBI). Sammenlignet med grafene i PowerBI er dette en lettere måte å få noe “instant” beskjed på. Grafene i nettsiden skal være ment for å se på mønster i større perspektiv og over lengre tidsperioder.

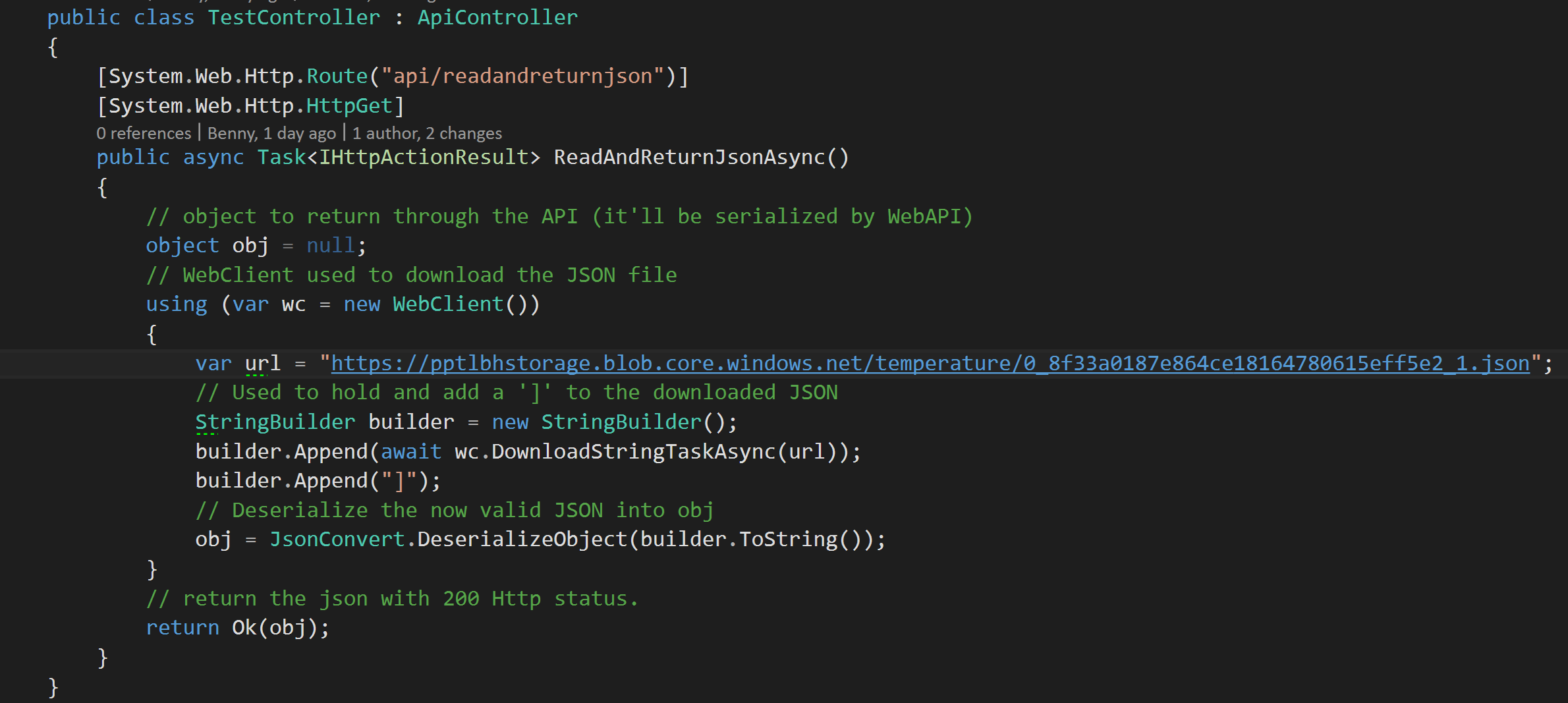
# Issues

Eventhuben i Azure tar litt lang tid før dataene fra Particle Photon blir publisert. Vi ser at det blir heletiden lagt til i Dashboard.particle.io. Prosjektet har tatt mye lengre tid enn forventet fordi det har tatt så lang tid for å få ting opp i skyen.

Har også prøvd å få til litt av hvert for å utforske hva muligheten er med Azure og particle photon. Dette førte til at produktet ser veldig halvferdig ut i og med at det ikke er noen direkte testing mot kjøleskap. Under prosjektet har jeg bare testet mot romtemperatur.

Grunnen til at jeg brukte PowerBI er fordi jeg lett kunne finne graf over hva jeg ville ha og legge ut i nettsiden. Med Web-Apiet er det også samtidig mulig å lage grafer utenom det også da vi har dataene i JSON format [www.pptlbhwebapi.azurewebsites.net/api/readandreturnjson](http://www.pptlbhwebapi.azurewebsites.net/api/readandreturnjson)

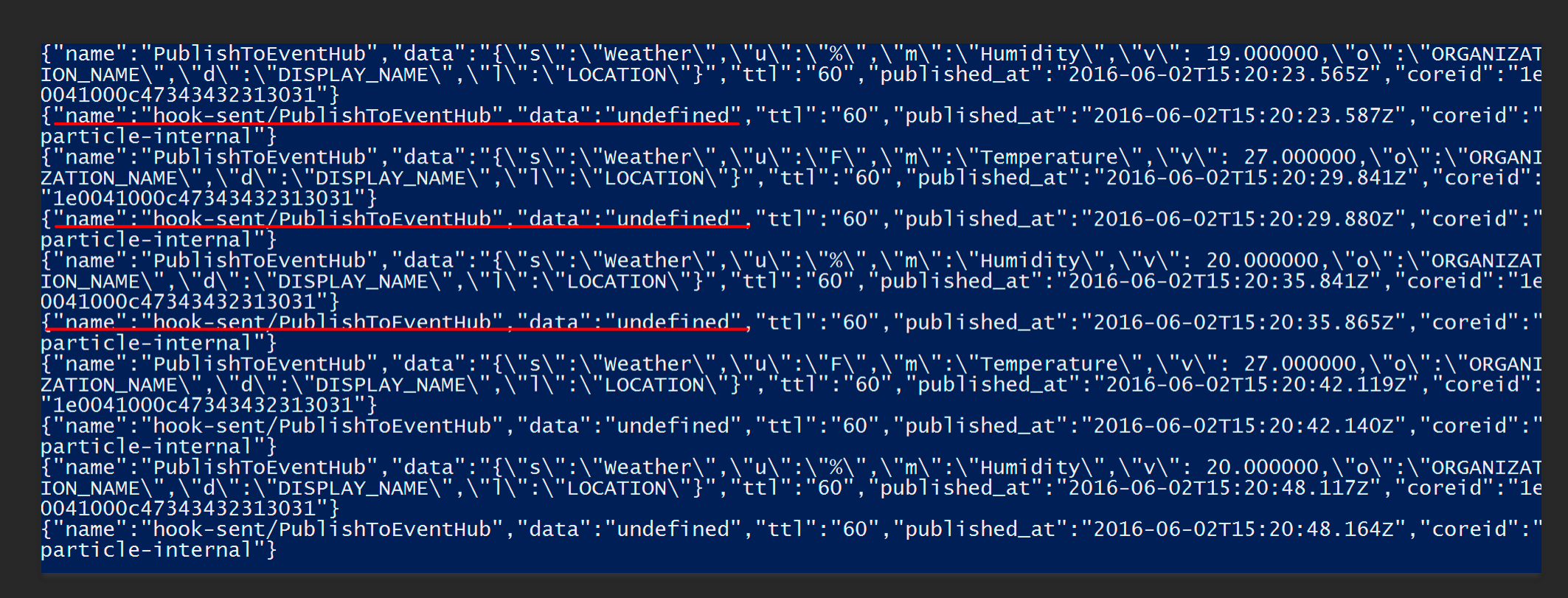
Måten dette blir gjort på er at jeg leser igjennom en storage blob som ligger i Azure. På slutten må jeg også append “]” fordi dette er en fil som fortsetter å få flere rader av data fra stream analytics.



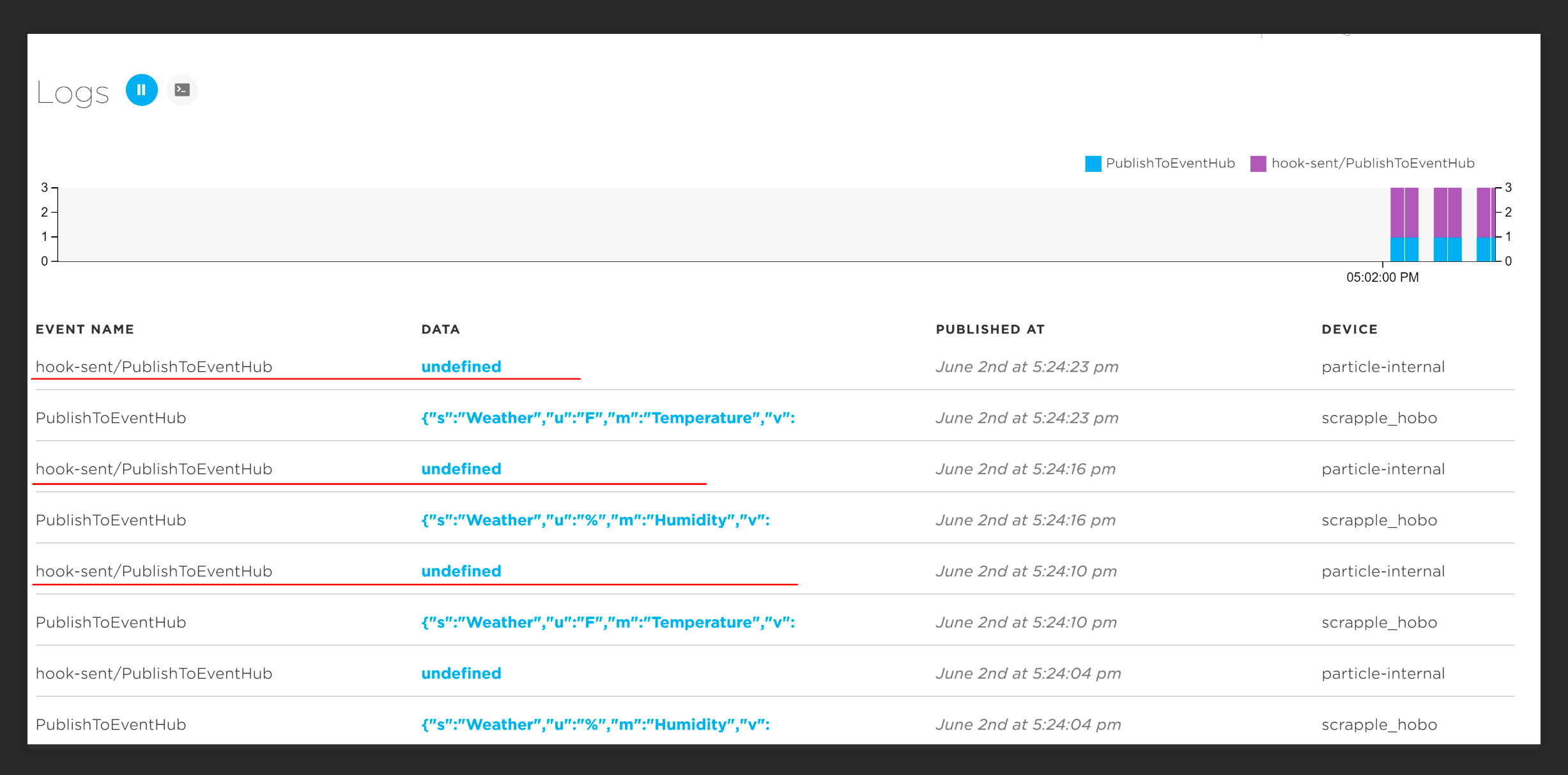
Fordelen her kunne også være å spesialisere ulike spørringer for dataene. Nå tar det veldig lang tid å lese filen og skrive den ut. Dette kan være feks at den henter ut den siste temperatur verdien.

Ved å kjøre “particle subscribe mine” vil du kunne se om webhooken blir publisert til Azure eller ikke. Dette kan du se også gjennom Dashboardet til Particle.

**CLI**

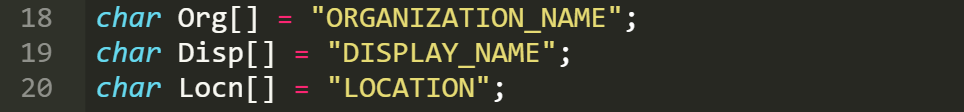


**Particle Dashboard**



Hvis den ikke dukker opp når du debugger, hjelper det å slette webhooken og lage den på nytt.

Meningen med å disse med i payloden er for å kunne skille på devices som er koblet til.



I grafene skal det være mulig å filtrere bort hvilken display\_name som har publisert til event-huben. Dette er enda ikke testet ut da jeg bare hadde en particle photon å teste med (blir spennende å se om det kommer opp data fra andre enheter når sensor prøver ut). Dette er gitt at Disp[] er initialisert til en annen tekst i de andre enhetene.

# Konklusjon

Det var veldig spennende å jobbe med particle photon sammenlignet med en arduino. Du har mye større plass til minne og den har Wifi, som gjør det mulig å gjøre masse kule ting med det du allerede kan av webutvikling. Når prosjektet er så spennende og kult, blir det større sjanse for å bli overambisiøs som jeg ble. Dette førte til at jeg mistet litt fokus på det faktiske produktet. Fikk aldri tatt den til en ordentlig test i kjøleskapet, men ble til at jeg heller testet alle ulike områder. Til senere tid blir det veldig lett for meg å bare endre på et par parametere så skal den være tilpasset et kjøleskap case.

## Video: <https://www.youtube.com/watch?v=hI3lukMubqc>

# Kilder

<https://docs.particle.io/guide/tools-and-features/webhooks/>

<https://docs.particle.io/reference/cli/>

<https://azure.microsoft.com/en-us/documentation/services/event-hubs/>

<https://azure.microsoft.com/en-us/documentation/services/stream-analytics/>

<https://azure.microsoft.com/en-us/documentation/services/storage/>

<https://azure.microsoft.com/en-us/documentation/articles/web-sites-dotnet-rest-service-aspnet-api-sql-database/>