18-12-2018

David Sørensen - 201509701

Aarhus Universitet Herning

Mailbox notifier

E5IOT Internet of Things

# Projektbeskrivelse

Der skal i dette projekt laves et IOT-produkt, der kan være en opgradering af et lignende produkt, som ikke er koblet til internettet og udfører en meningsfuld opgave. Systemet skal være lavet af embedded hardware og software samt indebære en sensor og aktuator af valgfri type.

# Introduktion

Mailbox Notifier er en enhed, som kan informere en bruger om at der er post.

Systemet er tiltænkt at kunne monteres på en normal postkasse, som selvfølgelig er indenfor rækkevidde til at kunne oprette forbindelse til internettet.

Systemet vil med en sensor måle når der åbnet for brevsprækken, så systemet ved hvornår der er post. Der sker herefter to events, som informerer brugeren om at der er post:

* En fysisk indikation på postkassen, som viser at der er post
* En notifikation via. internettet, som kan fortælle at der er post.

Brugeren kan altså både via internettet altid vide hvornår der er kommer post, om brugeren er hjemme eller ude. Brugeren kan også fysisk se at der er post, f. eks på vej ud af hjemmet eller på vej hjem igen.

Indholdsfortegnelse

[Projektbeskrivelse 1](#_Toc532379741)

[Introduktion 1](#_Toc532379742)

[Krav 3](#_Toc532379743)

[Functional requirements 3](#_Toc532379744)

[Techical requirements 3](#_Toc532379745)

[Ubiquitous 3](#_Toc532379746)

[Event-driven 3](#_Toc532379747)

[State-driven 4](#_Toc532379748)

[Option 4](#_Toc532379749)

[Unwanted Behaviour 4](#_Toc532379750)

[System Design 4](#_Toc532379751)

[Photon-Mobiltelefon 4](#_Toc532379752)

[Photon – Servo 4](#_Toc532379753)

[Photon – Sensor 6](#_Toc532379754)

[Systems arkitektur 6](#_Toc532379755)

[Købt hardware 7](#_Toc532379756)

[Teori – Twilio 7](#_Toc532379757)

[Implementation og test 9](#_Toc532379758)

[Servomotor [2] 9](#_Toc532379759)

[Kontakt [3] 10](#_Toc532379760)

[SMS-service [4] 11](#_Toc532379761)

[Twilio service 11](#_Toc532379762)

[Webhook [4] 12](#_Toc532379763)

[Photon – Kode 13](#_Toc532379764)

[Test 14](#_Toc532379765)

[Verification 14](#_Toc532379766)

[Konklusion 14](#_Toc532379767)

[Referencer 16](#_Toc532379768)

# Krav

Systemet har et sæt af krav, og disse krav er opført i henhold til EARS (Easy to Approach Requirements Syntax) Requirement Capture modellen der inddeler krav i fem forskellige kategorier:

* Ubiquitous – Krav der altid skal være til stede
* Event-driven – Krav der skal ske ved bestemte hændelser
* State-driven – Krav der skal opfyldes ved en bestemt state
* Option – Krav der skal opfyldes når produktet skal styres af bruger
* Unwanted Behaviour – Krav der skal opfyldes når produktet har en uønsket opførsel

Der er på forhånd listet projektkrav, som er listet først:

## Functional requirements

1. The device must be able to connect to the internet
   1. Internet connection shall be via WIFI
   2. The device should preferably be able to connect to AU’s “AU Gadget network”
2. Your device must be able to read data from a connected sensor, local to the device
   1. a sensor can be anything that quantifies a physical measure, into an electrical signal, such as temperature, light, humidity, presence, movement, magnetism, pollution, etc.
3. Your device must be able to control an actuator
   1. An actuator can be anything that translates an electrical signal into a physical quantity, such as, motors, servos, valves, heaters, displays, lamps, etc.
4. Your device must be capable of using data from a web service, to augment “what it does”, this could be weather data, traffic data, stock prices, twitter feeds, emails, rss-feeds or something different.
5. Your software and hardware design must be shared
   1. You must create a public github account, and add relevant project files here
   2. Hardware documentation, schematics, datasheets and pcb layouts are to be uploaded in pdf format
   3. Software files are to be uploaded in raw source code format, e.g. .C, CPP, .h, .py, etc.

## Techical requirements

1. The technical platform can be a suited embedded platform of your choice, e.g. the Particle Photon, an ESP8266, a raspberry pi, beagle bone black or similar.
   1. The platform shall have Wifi connectivity
   2. The platform shall have available digital or analog I/O con connect sensors and actuators

## Ubiquitous

1. Systemet skal kun være vågne i et minut ad gangen.
2. Servomotoren skal have en strømforsyning på 4,8-6V
3. Reset skal foregå når systemet er tændt og ikke i sleepmode
4. Enheden skal gå i sleepmode når der intet er at foretage

## Event-driven

1. Når systemet starter, skal flag sættes til start-positionen
2. Når systemet vågner fra sleepmode, skal flag sættes til gotmail-position
3. Når systemet vågner fra sleepmode, skal der sendes en sms
4. Når systemet bliver resat, skal systemet starte på ny
5. Når der skal sendes en sms, skal det være til et specifikt telefonnummer
6. Platformen skal være i stand til at læse analoge data
7. Platformen skal være i stand til at skrive digitale data

## State-driven

1. Når systemet er i sleepmode kan den kun vækkes ved en reset eller når brevsprækken åbnes.
2. Når brevsprækken er lukket, skal kontaktens output være 0
3. Når brevsprækken er åben, skal kontaktens output være større end 0.

## Option

1. Når reset knappen trykkes, skal systemet resettes
2. Når systemet resettes online, skal systemet resettes

## Unwanted Behaviour

1. Systemet må ikke kunne komme i en tilstand der vil sende sms’er for hyppigt
   1. Max 1 minut mellem hver sms

# System Design

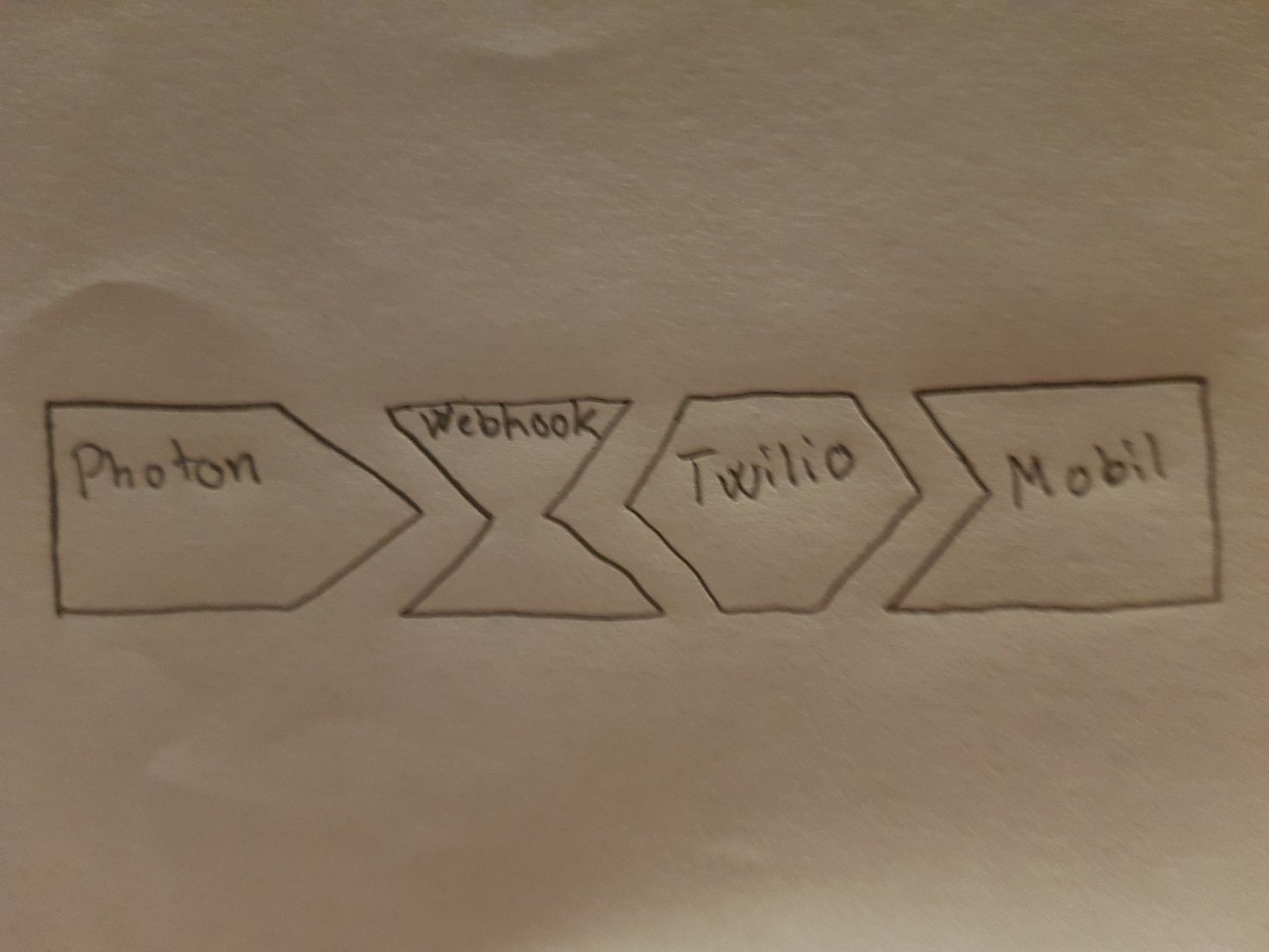
## Photon-Mobiltelefon

Systemet (Photonen) skal kunne kommunikere med en mobil i form af en SMS.

Particle enheder kan kommunikere med andet software gennem integration features, så som f.eks. en Webhook.

Twilio, er en service som kan sende sms’er, samt benytte andre kommunikationsformer.

Disse to mellemled har potentiale til at etablere forbindelse mellem Photonen og en mobil, så Photonen trigger Webhooken, som aktivere Twilio-servicen og sender en sms.



Figur 1 - Interface mellem Photon og mobil

## Photon – Servo

Systemet skal kunne styre en aktuator, som i dette tilfælde er en servomotor.

Servomotoren har følgende pins:

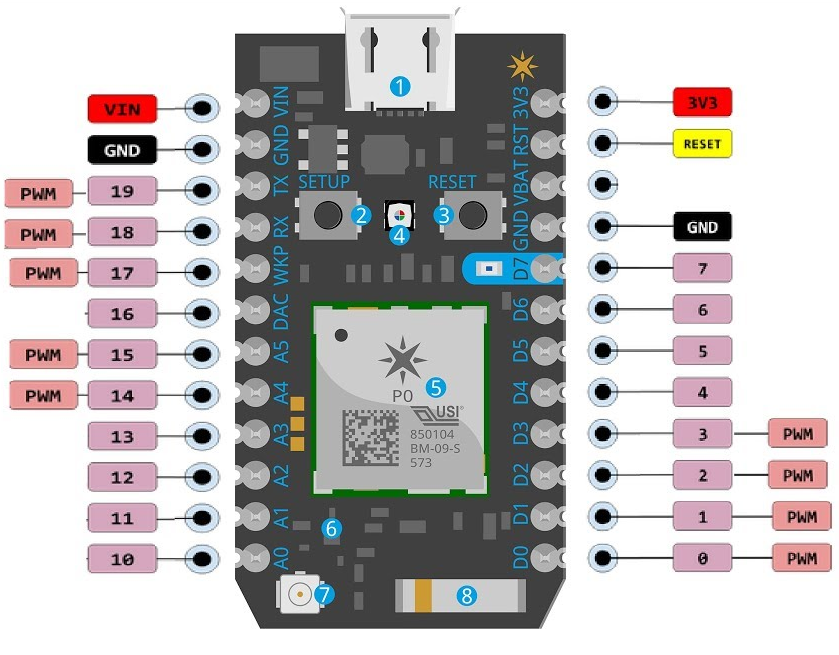
* Power
* Ground
* Control

Power skal have 4,8-6V

Ground skal forbindes til stel/jord/0V

Control skal forbindes til en PWM pin.

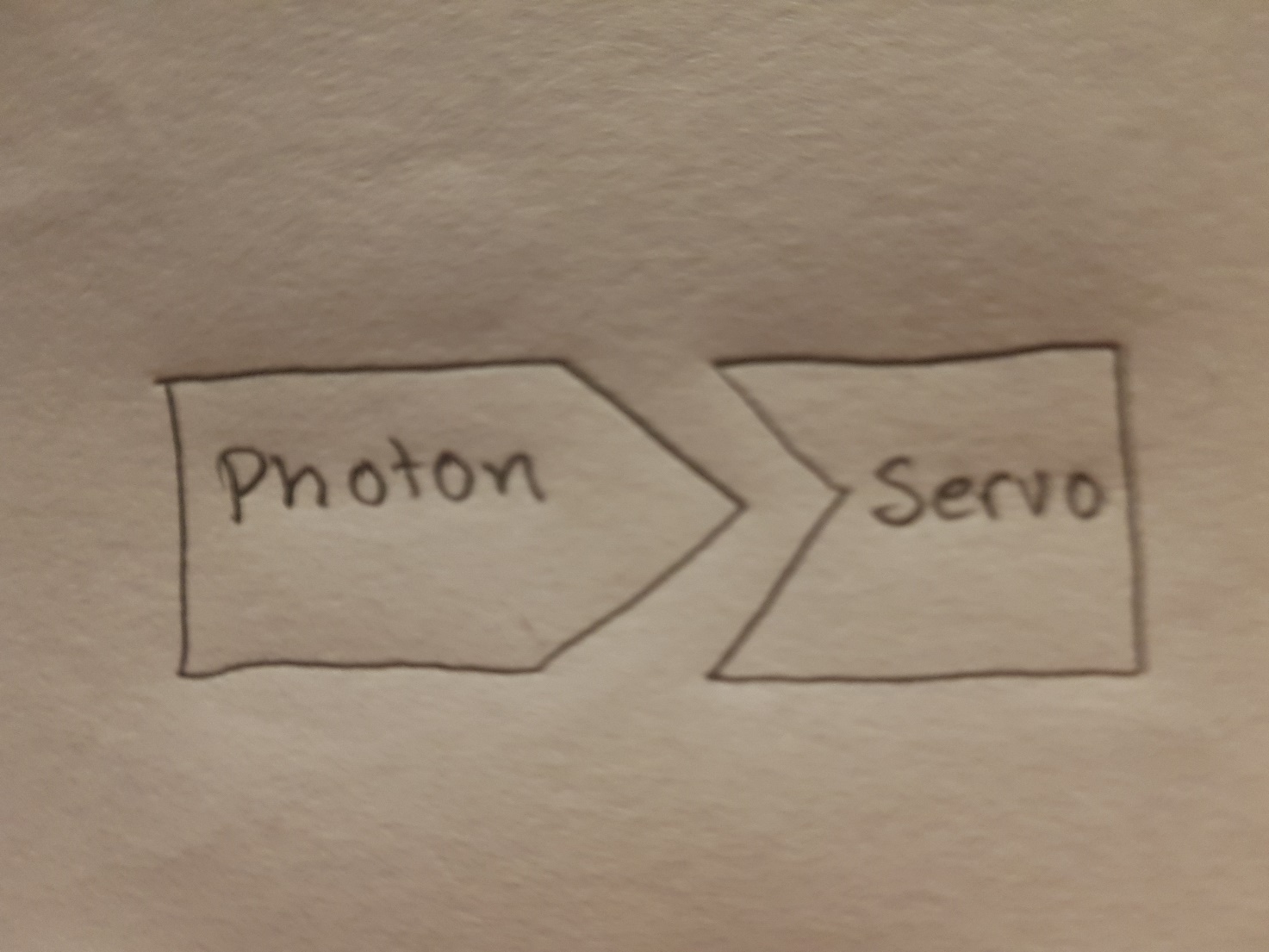
Photon har følgende pin’s der kan bruges til PWM



Figur 2 - PWM pins [1]

Servomotoren har som tidligere nævnt brug en strømforsyning, der kan give 4,8-6V og det kan Photon enheden ikke klare, derfor er der valgt at benytte et batteri der driver servomotoren. Der er benyttet et 9V batteri, som er nem at tilslutte et kredsløb. Dog er 9V for meget og derfor benyttes en 7805 spændingsregulator, der transformere spændingen til 5V.

Når servomotoren er forsynet med strøm, er det blot at forbinde den til en skrivbar PWM pin på Photonen, og Photonen er i stand til at styre servomotoren.



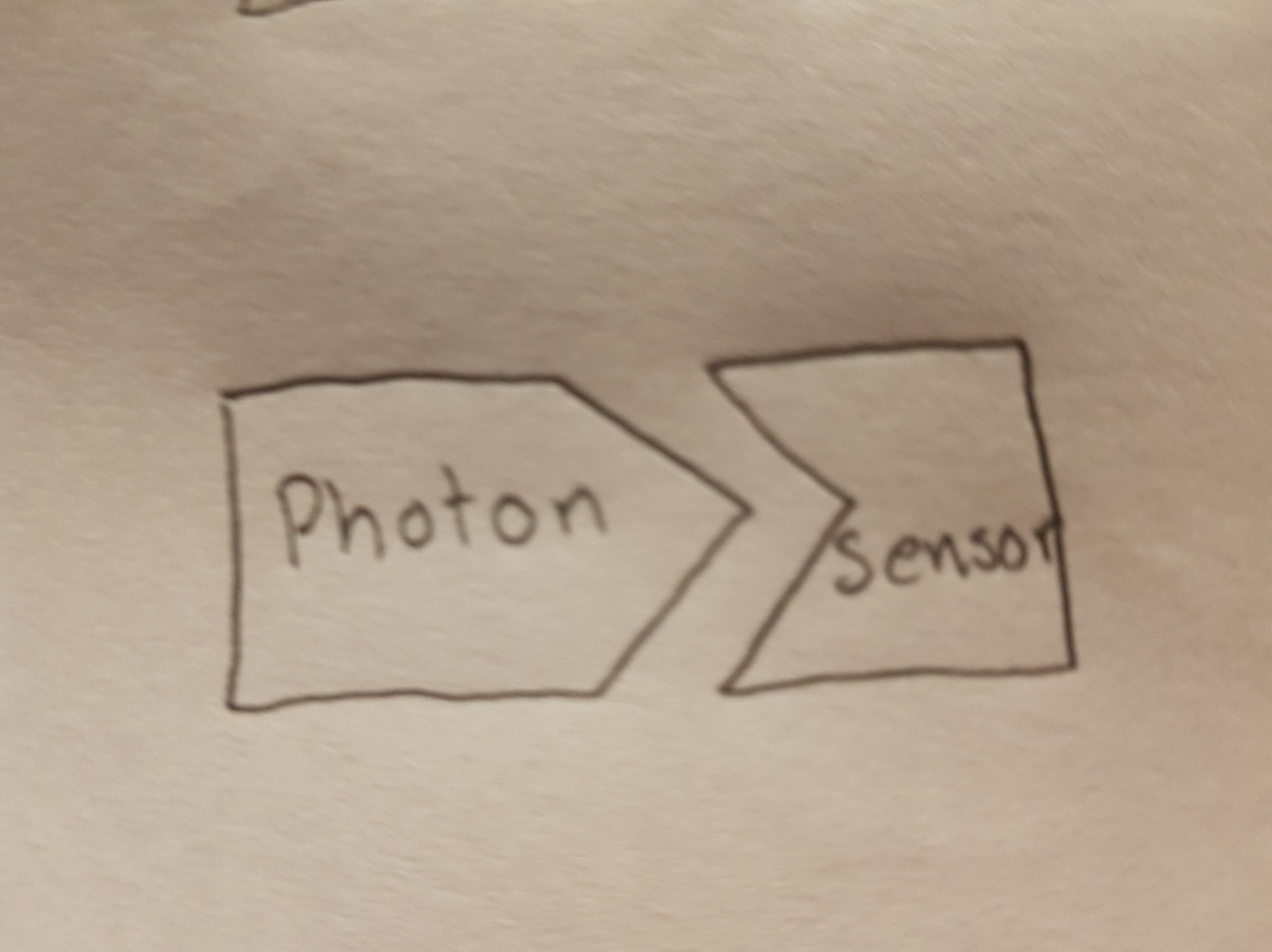
Figur 3 - Interface mellem Photon og servo

## Photon – Sensor

Systemet skal kunne læse fra en sensor, som i dette tilfælde er en kontakt.

Sensoren skal blot forbindes til en læsbar Analogpin på Photonen, og sensoren er i stand til at blive aflæst.

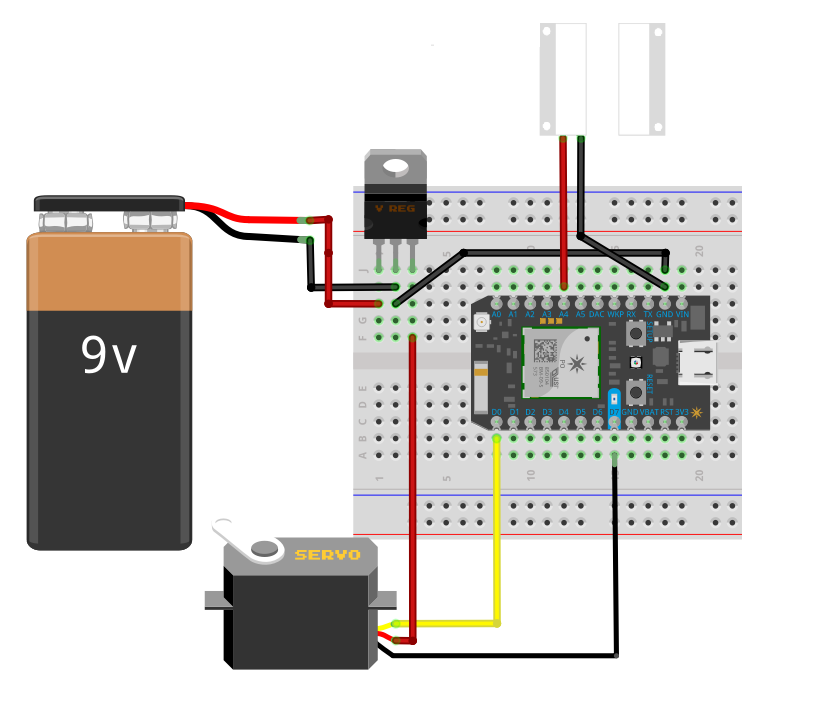
Så længe kontakterne er under 10mm fra hinanden regnes de for at være forbundet, hvilket vil sige at når brevsprækken åbnes skal denne afstand overskrides, for at kontakten udløses.



Figur 4 - Interface mellem Photon og sensor

## Systems arkitektur

Figuren nedenfor viser forbindelser mellem de forskellige dele, som udgør systemet.



Figur 5 - Schematic over kredsløb

## Købt hardware

Magnetisk kontakt: 20 kr.

<https://arduinotech.dk/shop/magnetisk-kontakt-aflang/>

Servomotor: 44 kr.

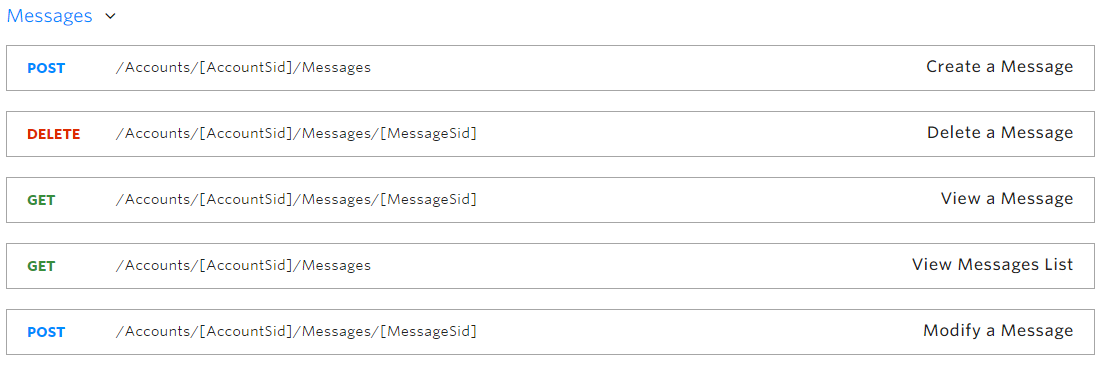
<https://arduinotech.dk/shop/servo-emax-es08ma-ii/>

# Teori – Twilio

For at forstå hvordan Twilio servicen fungere er der i dette afsnit dykket længere ned i Twilios API.

Som mange andre API tjenester benytter Twilio http-kommandoer til dataudveksling.

På Twilios hjemmeside kan man logge på sin konto og udforske SMS mulighederne for Twilios API. Udforske mulighederne og anvendelse.



Figur 1 - Et screenshot af Twilios hjemmeside der viser nogle af de muligheder "API Messages" tilfyder

For det her projekt er det kun relevant at udforske POST kommandoen, som handler om at lave en besked.

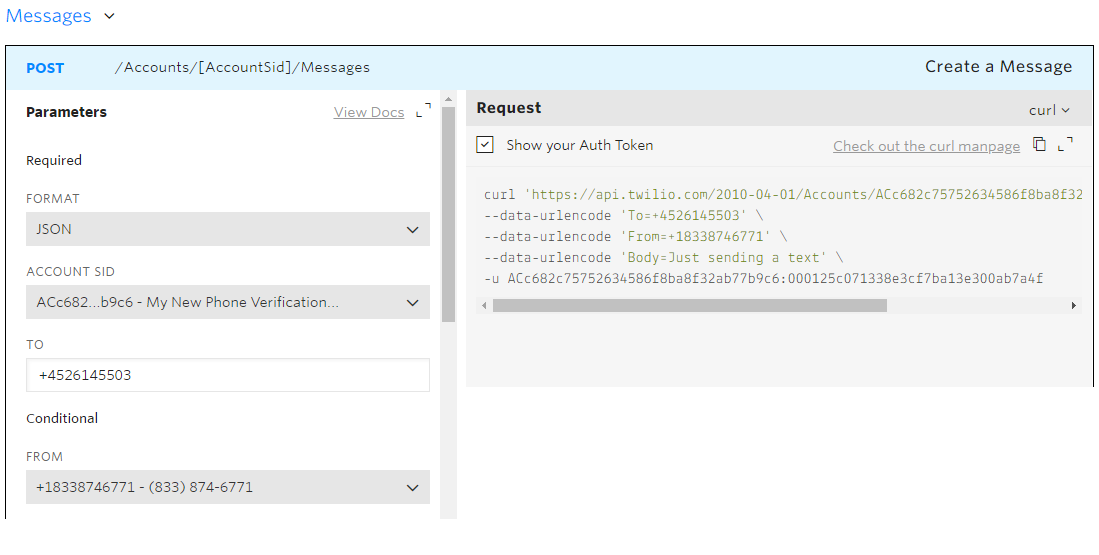
For at tilgå Twilios API benyttes som sagt http, med denne URL:

[https://api.twilio.com/2010-04-01/Accounts/<ACCOUNT\_SID>/Messages](https://api.twilio.com/2010-04-01/Accounts/%3cACCOUNT_SID%3e/Messages)

Som fortæller hvilken account der skal kaldes, og at det ønskes at tilgå servicen messages, altså der skal håndteres beskeder.

Der skal selvfølgelig nogle parametre med, som fortæller lidt mere om det her API kald.

* Format
* To
* From
* Body
* Authorasation Token



Figur 2 - Udvidet forståelse for "Create a Message" og kode generering

*Format* informerer om hvilket format dette kald skal læses i. Det ovenstående eksempel benytter JSON.

*To* informerer hvem denne besked skal sendes til.

*From* informerer hvilket nummer denne besked skal sendes fra, det skal være et nummer som er registreret til twilio-brugeren.

*Body* er selve den besked, som modtageren skal have.

*Authorasation Token* er en form for password der sikre at ikke alle kan bruge ens service.

På denne udforskningsside kan man nemt indtaste disse parametre og få autogeneret en kode der kan sende en sms. Man kan trigger API’en via kommandolinjen med curl kommandoen, eller med andre programmeringssprog, som f.eks. Python, PHP og C#. Hvilket betyder at man kunne trigger Twilio API’en (sende en SMS) hvis man havde brugt en mikrocontroller med internet i stedet for.

# Implementation og test

Implementeringen og test af hvert modul er beskrevet i hver deres afsnit.

Implementationen af kode er lavet i Particle Build (IDE) browser-portal. Nogle tests er lavet med Particle app’en.

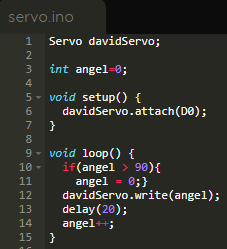
Der er lavet individuelle tests til hver af de enkelte moduler, og til sidst en samlet test til at afprøve samspillet mellem alle parter.

## Servomotor [2]

Servomotoren styres ved at regulere dutycycle, dog er det ikke nødvendigt at forholde sig til dette pga. de indbyggede biblioteker, der sørger for at vinklen, der ønskes at servomotoren skal have, blot er et tal (integer, float mm.), som skrives ud til servomotoren.

Til at styre en servomotor skal der skal først laves en servoklasse med typen *Servo.* Derefter skal servoklassen knyttes til en pin med kommandoen *servoklasseNavn.attach(pin).* Når man vil skrive til servomotoren, altså ændre vinklen bruger man *servoklasseNavn.write(angel in degrees)* kommandoen.

Nedenfor er implementeringen af testen, som først sætter vinklen til 0 og derefter øger vinklen med 1 integer-værdi. Denne test viser det range servomotoren skal være i stand til at udføre.



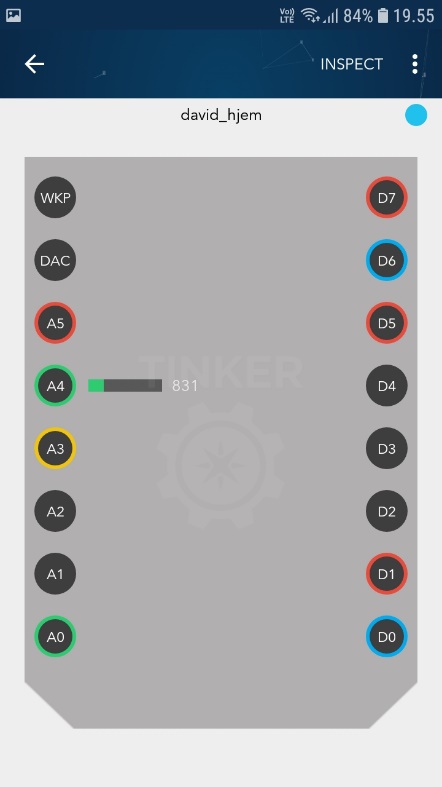
Figur 1 - Kode til styring af servomotor

Test af dette script fungerer som forventet; øger vinklen gradvist fra vinklen 0 til 90.

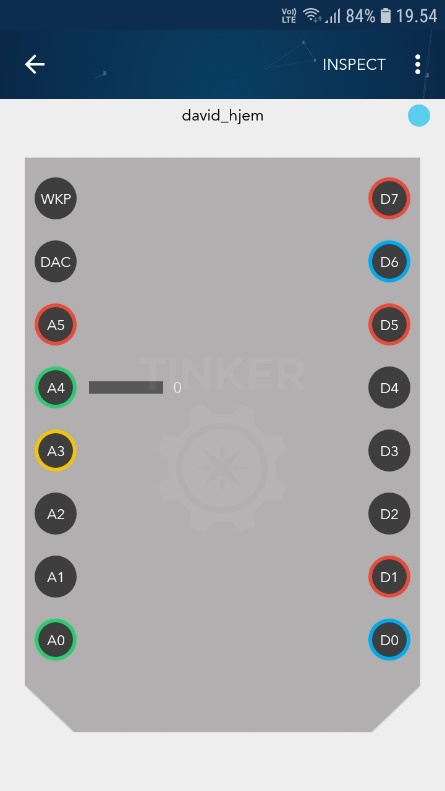
Til det endelige program skal servomotoren blot udføre to vinkler 0 og 90. 0 grader til at starte med og 90 grader når der er kommet post.

## Kontakt [3]

Kontakten er testet med Particle app’en, hvor der er læst fra pin A4, som er den læsbare pin fra kontakten (den anden ledning går i GND). Når de to dele til kontakten er forbundet, er outputtet 0, men når de fjernes fra hinanden, aflæses der en betydelig højere værdi fra den analoge pin.



Figur 2 - Kontakten er afbrudt



Figur 3 - Kontakten er forbundet

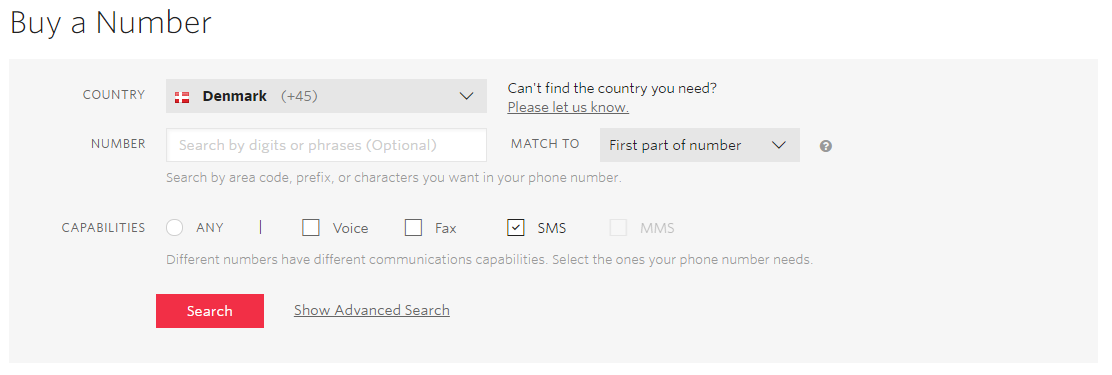
Ud fra dette kan der konkluderes at der kan skelnes mellem forbundet og afbrudt forbindelse. Denne kontakt skal til det endelige program vække Photon’en fra sleepmode, enten ved at detektere at denne analoge værdi øges eller falder.

## SMS-service [4]

Implementation af SMS-servicen er lavet i samspil mellem forskellige services. Følgende steps er udført for denne implementation.

1. Køb af telefonnummer, som skal bruges som afsender.
   1. Købt hos Twilio.com
2. Opsæt Webhook, som mellemled mellem Twilio og Photon
3. Kode der aktiverer Webhook’en og sender en SMS

### Twilio service

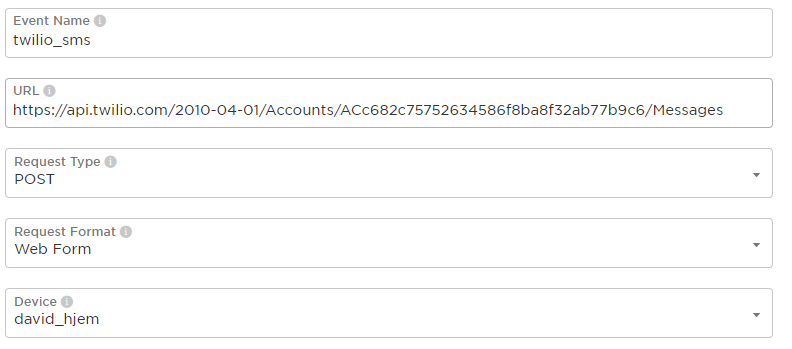


Figur 4 - Køb af telefonnummer

Efter at have oprettet en Twilio-konto, kan man med en trial konto oprette et telefonnummer. Der er lidt begrænsninger ved en trial konto, men det er muligt at anskaffe et nummer der kan klare sms’er. Dog følger der i sms’en en lille trail reklame med (kan ses i et senere afsnit).

### Webhook [4]

Der er blevet oprettet en Webhook integration på console.particle med følende data, som gør det muligt at sende data fra Particle enheder til en anden web service.



Figur 5 - Opsætning af Webhook

Event navn: Der er valgt et navn til denne Webhook integration.

URL: En URL på den API-service hvor data skal sendes til når den her Webhook bliver udløst. URL’en indeholder Twilio Account SID, så den korrekte konto findes.

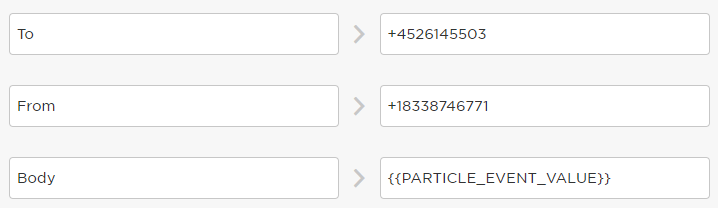
Request type: Valgt til at være POST, som sender data ud til Twilio servicen.

Request format: Valgt til at være Web Form, som er det format Webhook’en vil videregive data.

Device: Navn på den enhed, der kan benytte denne integration.

**Dynamisk Data**

Disse data bliver sendt afsted sammen med Webhooken



Figur 6 - Opsætning af dynamiske data

To - Til hvilket nummer – Brugers nummer

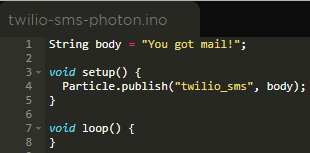
From – Fra hvilket nummer - Twilio-nummeret

Body – Indholdet som skal sendes i SMS’en, som er den

Dette er data som Twilio-servicen kan forstå.

## Photon – Kode

Koden er meget simpel, den skal blot trigger den Webhook, som går under navnet ”twilio\_sms”. Den tekst der skal sendes som sms, er argument to, som i dette tilfælde er en string, der har følgende data: ”You got mail!”.



Figur 7 - Kode der sender en sms

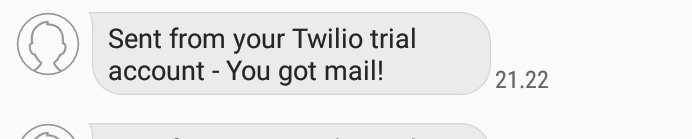
[5] Funktionen *Particle.publish* bruges til at publish (offentliggøre) en event til lyttere og i dette tilfælde Webhooken. Der er muligt også at sende nogle parametre med. Første parameter er navnet på den event der skal der skal udløses (Webhooken: *twilio\_sms*), den anden parameter er data der kan sendes med. Den fulde syntax er her:   
*Particle.publish(String eventName, String data)*

Det er muligt at undlade denne dataparameter, og blot specificere den ønskede sms-tekst i Webhooken i feltet *Body*. Men body er en particle\_event\_value, som netop er denne parameter data, som sendes med.

Det er vigtigt at funktionen til at udløse sms’en ikke forekommer for ofte og slet ikke i en evig loop, da det kan crashe Twilio-services.

### Test

Testen udføres blot ved at flashe testkoden til ens Particle Photon. SMS’en bliver modtaget efter få sekunder og ser således ud:



Figur 8 - Sms modtaget fra Twilio

Som det kan ses, er den body der blev sendt med Publish funktionen, sendt med i SMS’en. Og som nævnt tidligere følger der også en automatisk starttekst med, pga. trail kontoen.

# Verification

Efter at have testet og alle dele hver for sig er næste step at samle alle funktionerne til et samlet system. Systemet skal kunne styre en servomotor, modtage data fra en kontakt, udløse Webhooken.

Til det samlede system er der dog tilføjelser. Systemet sættes i sleepmode, og reagere på en faldende værdi af kontakten. Det vil mindske strømforbruget af systemet.



Figur 9 - Kode til systemet

# Konklusion

Der er lykkedes at lave en prototype af et system, der kan informere en bruger omkring når der er post, ved brug af en kontakt, som sensor, servomotor, som aktuator i samspil med en Particle Photon, som læser fra sensor og styrere aktuator og kommunikere med brugeren ved hjælp af internettet, og sender SMS’er der fortæller både hvornår systemet at klart, og hvornår der er modtaget post. Systemet kan også give brugeren besked om post ved et *flag* på postkassen, som indikere post ved en vertikal position.  
Der er lavet en offentlig Github konto til alt dokumentation omkring dette projekt, som kan ses på følgende URL: <https://github.com/stepperen/IOT>

Udvikling af systemet har været spændende og meget lærerigt især i form at API-lære og samspil mellem web-services, og bare det at kunne sende en SMS med et program, har været en win-oplevelse for mig.

# Referencer

|  |  |
| --- | --- |
| [1] | J. Ellis, »youtube,« August 2017. [Online]. Available: https://www.youtube.com/watch?v=oGOPE2gaRtE. [Senest hentet eller vist den oktober 2018]. |
| [2] | ukendt, »electronicwings,« [Online]. Available: https://www.electronicwings.com/particle/servo-motor-interfacing-with-particle-photon. [Senest hentet eller vist den Oktober 2018]. |
| [3] | rsaiyid, »instructables,« [Online]. Available: https://www.instructables.com/id/Magnetic-Door-Sensor-and-Arduino/. [Senest hentet eller vist den Oktober 2018]. |
| [4] | P. Kamp, »twilio,« September 2018. [Online]. Available: https://www.twilio.com/docs/sms/tutorials/how-to-send-sms-messages-particle-photon. [Senest hentet eller vist den September 2018]. |
| [5] | particle, »particle,« [Online]. Available: https://docs.particle.io/reference/device-os/firmware/photon/. [Senest hentet eller vist den November 2018]. |