

Team kcj

Elektronisk dør lås med nøgle app

E5IoT projekt

Aarhus Universitet, School of Engineering
Elektronik, Herning



Lærer: Klaus Kolle og Morten O. Jakobsen.
Teammedlemmer:

Jan Kastbjerg Schiermer
201701970

Kasper Møbjerg Jensen
201703818

Christoffer Wesselhof
201808478

Indholdsfortegnelse

Introduktion	4
Introduktions youtube video	4
Github URL	4
 Projekt description.....	4
Overordnet.....	4
Argon platform.....	5
Thingspeak API.....	5
Expo App	5
Projekt styring	5
 Kravs analyse.....	6
Skole satte krav	6
Egne krav	6
 System design.....	7
Samlet	7
Argon platform.....	7
Kode	7
Kredsløb	9
Thingspeak API.....	10
Expo App	10
Master App	11
Gæste App.....	11
 Implementation.....	11
Argon platform.....	11
Kode	11
3d printning.....	13
Thingspeak API.....	21
Expo App	22
Master App	23

Gæste App.....	23
Test/Verifikation.....	25
Overordnet.....	25
Argon platform.....	25
Thingspeak API.....	25
Expo App	25
Konklusion.....	26
Figur 1 High level overview	5
Figur 2 high level sequence diagram	7
Figur 3 kode samhæng.....	9
Figur 4 argon kredsløb	10
Figur 5 high level thingspeak diagram	10
Figur 6: 3D tegning af del nr. 1.....	13
Figur 7: 3D tegning af del nr. 2.....	14
Figur 8: 3D tegning af del nr. 3 set fra bunden.....	14
Figur 9: 3D tegning af del nr. 3 set fra toppen.....	14
Figur 10: Arm fra servo-motor monteret i del nr. 3.....	15
Figur 11: 3D tegning af del nr. 4.....	15
Figur 12: 3D tegning af del nr. 5.....	16
Figur 13: 3D tegning af del nr. 6 zoomet ind på pindene i bunden	17
Figur 14: 3D tegning af del nr. 6.....	17
Figur 15: Del nr. 1 monteret over vrideren, samt del nr. 5 monteret på dørkarmen.....	18
Figur 16: Del nr. 2 monteret under låsecylinder dækslet, med feedback LED'er	18
Figur 17: Elektronikken monteret på del nr. 4 og del nr. 3 monteret på servo-motoren	19
Figur 18: Del nr 4 med elektronik monteret på del nr. 1	20
Figur 19: Del nr 6 monteret på del nr. 1	20
Figur 20 particle.io webhook template.....	21
Figur 21 thingspeaks mulighed for forskellige fields	21
Figur 22 thingspeaks webhook template.....	22
Figur 23 thingspeaks react app	22
Figur 24 Det grafiske vist på telefonen	23
Figur 25 Kode til GUI på telefonen.....	23
Figur 26 De to curl funktioner	23
Figur 27 Kode til GUI på telefonen.....	24
Figur 28 Det grafiske vist på telefonen	24
Figur 29 sendRequest funktionen	24
Figur 30 thingspeak test.....	25
Figur 31 overordnet test	26

Introduktion

I E5IoT skal der laves et internet of things projekt, men ved at der skal bruges en mulig IoTplatform, til at kunne udføre en meningsfuld opgave, hvori der bliver brugt diverse værktøjer som fx. WiFi, sensorer og aktuatorer.

Projektet beskrevet i denne rapport vil være omhandlende en elektronisk dør lås, hvor der bliver kreeret en "nøgle" app til denne lås.

Tanken bag dette projekt er at have en nem og effektiv metode til at kunne opfylde diverse mangler og ulemper en traditionel mekanisk nøgle+lås vil have, men dog er målet ikke at erstatte disse traditionelle midler, nogle af ulemperne kan være hurtig udskiftning af "nøglen", er så simpelt som og generere en ny kode, samt dele en "nøgle" vil være at bare skulle give en kode/api nøgle ud, mens det stadig vil være muligt og kontrollere hvem man lukker ind i sit hjem.

Introduktions youtube video

https://www.youtube.com/watch?v=OuilD-ZGZ1g&feature=youtu.be&ab_channel=jankastbjerg

Github URL

<https://github.com/IOT-teamKCJ>

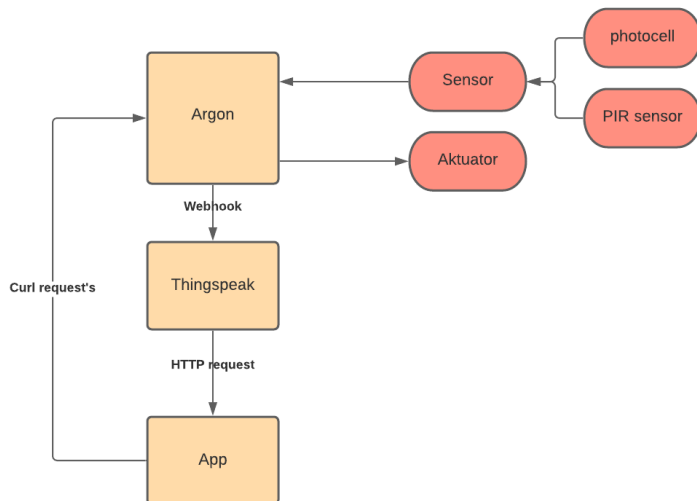
Projekt description

Overordnet

Projektet vil bestå af 3 nøgle elementer, samt vil der bliver brugt 2 platforme til projektstyring/deling:

- 1) argon platformen
 - a) Til at styre låsen
- 2) Thingspeak's API
 - a) Som bindeled og logger
- 3) App lavet med expo
 - a) Som "nøgle" + brugerflade
- 4) Github
 - a) Til projekt styring og deling
- 5) Discord
 - a) Til Projekt styring og deling

For at hele projektet vil skulle kunne hænge sammen skal disse dele kunne snakke sammen, hvilket vil blive gjort med forskellige protokol typer, som vil kunne ses på figur 1.



Figur 1 High level overview

Argon platform

Koden til argonen vil blive udviklet i VSC (visual studio code) SDK'en til particle, for derefter at blive deployet til platformen igennem SDK'en til argonen.

Argonen vil agere som styringen af aktuatoren, som vil være en motor til styring af dør låsen, og være modtager for sensor dataen, som vil kunne fortælle om døren er åben eller lukket, samt om der står nogen foran døren.

Argonen vil igennem events og webhooks være sat sammen med Thingspeaks API.

Thingspeak API

Thingspeak API'en vil stå for at logge diverse events, samt sende HTTP request's til appen igennem de integrerede funktioner der eksisterer på hjemmesiden, for at give notifikationer om dør oplåsnings request's.

Expo App

App'en vil være brugerens hovedplatform, og vil agere som "nøgle", ved at sende en andmodning til en bruger om de må komme ind ad en dør de ikke selv er ejer af.

App'en vil også agere som en kontrol platform til at styre sin egen dør lås, ved at kunne direkte låse døren op.

Projekt styring

Github og Discord bliver brugt til at dele filer, billeder, osv.. samt kunne have en form for version styring på projektet.

Kravs analyse

Skole satte krav

- 1) The device must be able to connect to the internet
 - a) Internet connection shall be via WIFI
 - b) 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
 - a) 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
 - a) 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
 - a) You must create a public github account, and add relevant project files here
 - b) Hardware documentation, schematics, datasheets and pcb layouts are to be uploaded in pdf format
 - c) Software files are to be uploaded in raw source code format, e.g. .C, .CPP, .h, .py, etc.
- 6) 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.
 - a) The platform shall have Wifi connectivity
 - b) The platform shall have available digital or analog I/O connect sensors and actuators

Egne krav

1. app:
 - a. app'en skal kunne styre en dørlås
 - b. gennem app'en skal der kunne gives en anmodning om at låse døren op
 - c. app'en skal kunne give en notifikation til brugeren, hvis nogen gerne vil ind ad ens dør
2. argon:
 - a. argonen skal kunne styre dør låsen med en aktuator
 - b. argonen skal kunne aflæse en photocelle
 - c. argonen skal kunne aflæse en PIR sensor
 - d. argonen skal kunne modtage curl request's
 - e. argonen skal kunne kommunikere med thingspeak
3. thingspeak:
 - a. thingspeak skal kunne logge data

- b. thinspeak skal kunne sende http's request's til expos cloud, for at app'en kan generere notifikationer
4. github:
 - a. alt kode skal ligge på github

System design

Samlet

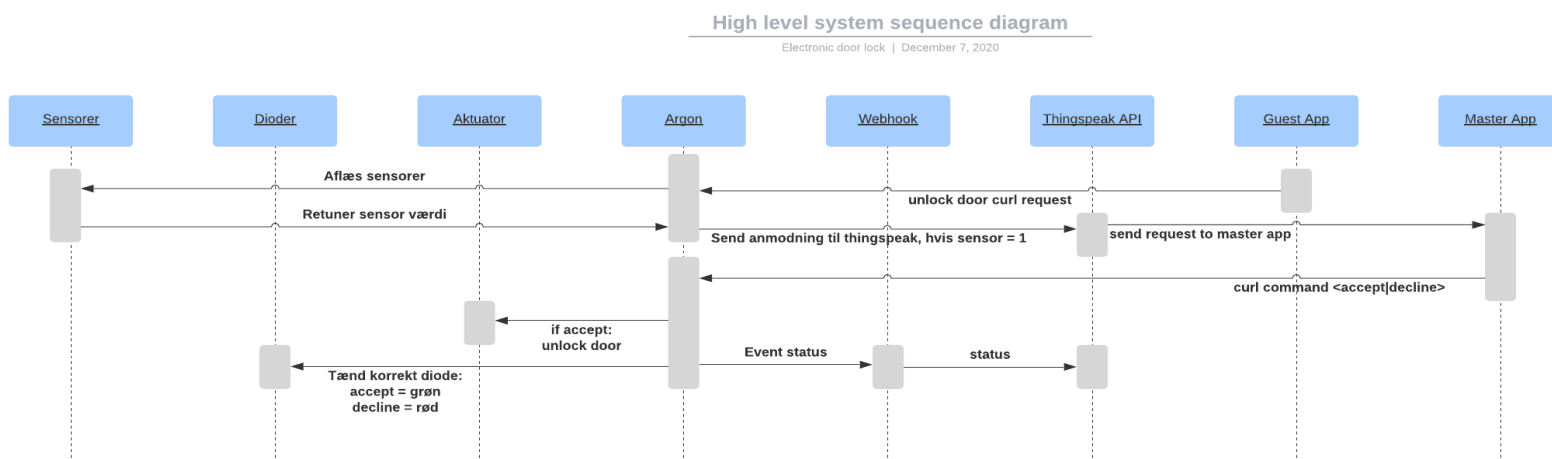
Systemets flow vil foregå som vist på figur 2.

En gæst vil sende en anmodning, om at komme ind af en dør, som vil blive sendt til argonen, som vil tjekke PIR sensoren, om nogen står foran døren. Hvis der står nogen foran døren, vil anmodningen blive sendt videre til thingspeak gennem webhooks, og hvis ikke vil anmodningen dø ud.

Thingspeaks API, vil herefter sende andmodningen til ejerens app i form af en notifikation, ved hjælp af expos cloud og https post request's, hvor ejeren så vil kunne acceptere eller decline anmodning på app'en.

Efter ejeren har trykket accept eller decline i appen, vil der blive sendt en besked til argonen, som så enten vil åbne døren, eller prøve at lukke døren (i tilfælde af at døren allerede er lukket, vil decline ikke have nogen synlig effekt på aktuatoren).

For at kunne se om ens anmodning bliver accepteret eller declinet, vil der fra argonen være 2 dioder, som enten lyser rød (ved decline) , eller lyser grøn (ved accept). Dioderne vil kun være tændt i X sekunder.



Figur 2 high level sequence diagram

Argon platform

Kode

I koden vil der være følgende funktioner:

- Setup()
 - Til at opsætte:
 - Dioder, til at vise status
 - Aktuator, til at kunne åbne låsen
 - Sensor, til at kunne sense om døren er åben/lukket
 - Variabler, for at variabler/status kan ses i particles cloud API
 - Funktioner, for at der kan sendes kommandoer igennem particles API
- Loop()
 - Bruges ikke til dette program
- App_request()
 - Til at aflæse PIR sensoren
- servoAngle()
 - til at kunne styre aktuatoren
 - tager kommandoerne:
 - toggle: sætter aktuatoren til 180°, og derefter 0°
 - lock: sætter aktuatoren til 0°
 - unlock: sætter aktuatoren til 180°
- master_command()
 - tager curl kommandoer/request's, til at styre argonen
 - tager kommandoerne:
 - Accept: åbner døren, tænder en grøn diode, og sender et status event
 - Deny: sætter døren til at skulle låse, og tænder en rød diode
 - "ikke accepteret kommando": sender et status event
- doorIsOpen()
 - returnerer true hvis døren er åben
 - returnerer false hvis døren er lukket
- readSensor
 - aflæser sensor værdien

Master_command(allow) giver diverse status events, alt efter diverse situationer. De forskellige status events bliver sendt til thingspeak, hvor det er muligt at se den returnerede status kode, der gør det muligt at aflæse hvordan ens kommando er blevet modtaget/afviklet.

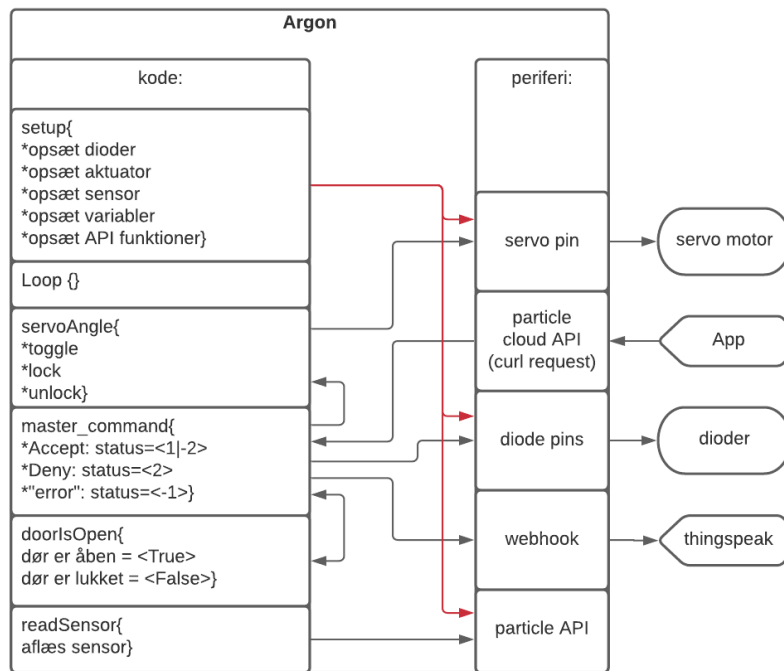
Statuskoder:

- 2: Deny kommandoen blev afviklet
- 1: Accept koden blev afviklet
- -1: forkert kommando blev givet
- -2: Accept kommandoen blev givet, men døren var allerede åben

I tilfælde af statuskode -2, vil LED'erne skifte imellem grøn og rødt lys i 2 sekunder.

Når en enmodning bliver accepteret, vil der være delay på 10 sekunder til at åbne døren, hvorefter den vil låse døren igen. I tilfælde af at døren er åbnet, vil der køre et loop som venter på at døren bliver lukket igen, hvorefter ved loop afslutning, vil døren låse sig igen.

Designet for hvordan koden hænger sammen kan ses på graf form i figur 3.

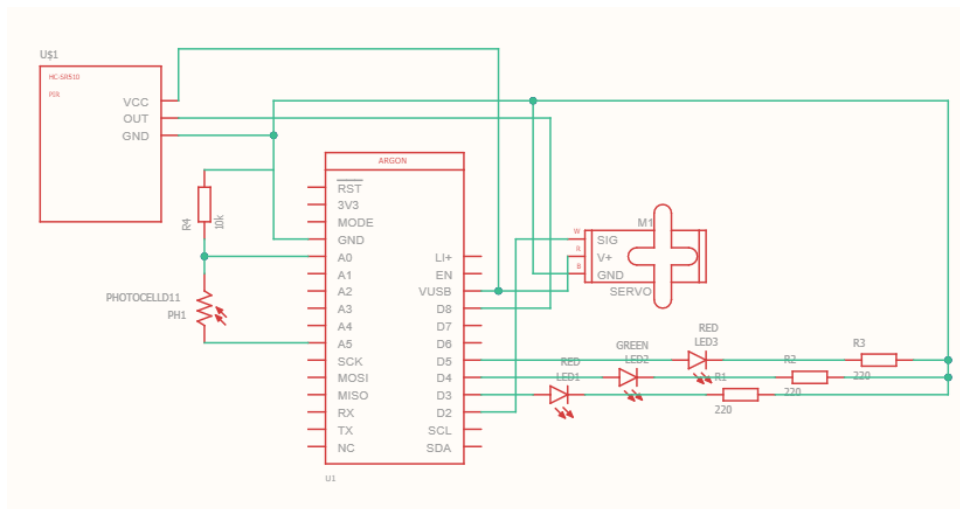


Figur 3 kode samhang

Kredsløb

Komponenter i kredsløbet:

- 3 dioder
- 4 modstande
- 1 PIR sensor (U\$1)
- 1 Servomotor (M1)
- 1 photocelle



Figur 4 argon kredsløb

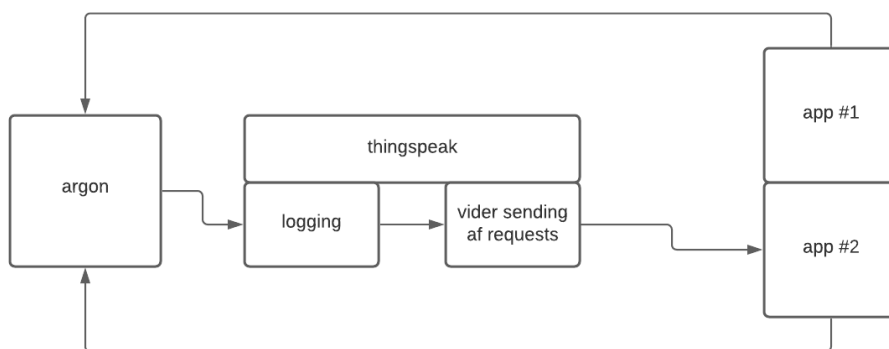
Thingspeak API

Thingspeak vil blive brugt som logger, samt vil det blive brugt til at levere request fra gæste anmodninger, til ejeren af en given lås, i form af notifikationer.

Logging delen af projektet vil blive gjort ved at kreere et event på argonen, for herefter at lade en webhook på particle.io opfange eventet, som så vil sende det videre til thingspeaks API.

Request håndteringen vil foregå ved at telefon Appen sender en curl request til argonen, som sender den videre til thingspeak, hvorefter thingspeak, igennem sine egne app's, er blevet sat op til og skulle sende disse request's videre til expo's cloud (et mobil framework), som vil generer en pop-up notis på telefonen.

High level samhæng kan ses på figur 5.



Figur 5 high level thingspeak diagram

Expo App

Appdelen af projektet har for at gøre det nemt idet mobil app udvikling ikke er hovedfokus for dette fag lavet i to apps en master app, som kan styre døren, og en gæste app, som kan spørge om adgang. Begge apps bliver udviklet ved brug af Expo på computeren samt Expo appen på telefonen. Det er en tunnel forbindelse der oprettes mellem computer og telefon således at mens man udvikler

vil koden blive pushet til telefonen som så kan vise hvordan ens kode bliver forstået. Selve koden bliver udviklet i Visual Studio.

Master App

Master appen har til opgave at styre døren, dette kan gøres som ejer af døren der kommer hjem og vil ind, ved at trykke "accept". Denne apps andet ansvar er at tage imod notifikationer fra folk der spørger om adgang.

Gæste App

Gæste appen har til opgave at give brugeren mulighed for at spørge om adgang, ved at sende et request gennem en curl kommando. Hvis alt er som det skal være, altså at PIR sensoren kan se nogen vil en notifikation komme hos master appen som så kan låse døren op for en.

Implementation

Argon platform

Kode

Servo:

I hensyn til servo motoren til låsen, har Particle biblioteket inbyggede funktioner til dette. man starter med at lave et servo objekt:

```
// create servo object to control a servo
// a maximum of eight servo objects can be created
Servo myservo;
```

Efter servo objekt kreering kan man sætte en pin til dette objekt:

```
// setup actuator
myservo.attach(D2);
```

Hvorefter det er muligt at skrive grader til servoen:

```
myservo.write(0);
```

Master kommandoer:

For at sende en master kommando kan man bruge curl, hvor strengen til at kunne sende argumenter til en funktion vil i dette projekts tilfælde være (i tilfælde af windows vil det være curl.exe <kommando>:

```
curl https://api.particle.io/v1/devices/<DEVICE-ID>/Master?access_token=<ACCESS-TOKEN> -d "arg={access:<deny or allow>}"
```

master kommandoerne vil blive modtaget i JSON format, hvor det er muligt at kopiere den

modtagne JSON streng til en JSON variabel, og iterere igennem JSON objekterne:

```
//insert JSON string into JSON object, and iterate through object
JSONValue master_msg = JSONValue::parseCopy(data);
JSONObjectIterator iter(master_msg);

while(iter.next())
{
```

Dog for at det er muligt at aflæse dataen skal JSON objektets iter streng formateres, hvorefter det er muligt at aflæse data navn og værdi:

```
// make it possible to read JSON string
String command = iter.value().toString().data();

// go through JSON string
if (iter.name() == "access")
{
    if(command == "Accept") {
```

Herefter agerer master_command funktion på forskellige måder, alt efter hvad "access" værdien er (Accept, Deny, eller en ikke accepteret kommando), ved hjælp af if statements.

Måden den agerer på, er ved hjælp af:

- digitalWrite
- Particle.publish
- myservo.write
- delay
- doorIsOpen

```
// open door
digitalWrite(power,HIGH);
digitalWrite(lightSensorLED, HIGH);
delay(200);
if(!doorIsOpen()) {
    Particle.publish("lock_hook", "1", PRIVATE);
    Particle.publish("door is open", PRIVATE);
    digitalWrite(greenLED, HIGH);
    myservo.write(180);
    delay(10000);
    while(doorIsOpen())
    {
        delay(10);
    }
    myservo.write(0);
    Particle.publish("door is closed", PRIVATE);
    digitalWrite(greenLED, LOW);
    digitalWrite(lightSensorLED, LOW);
    digitalWrite(power,LOW);
    return 0;
}
```

doorIsOpen:

Denne funktion fungerer ved hjælp af if statements og particles "analogRead" funktion:

```
// function for reading door status
bool doorIsOpen() {
    if(analogRead(photoresistor)>100)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

3d printning

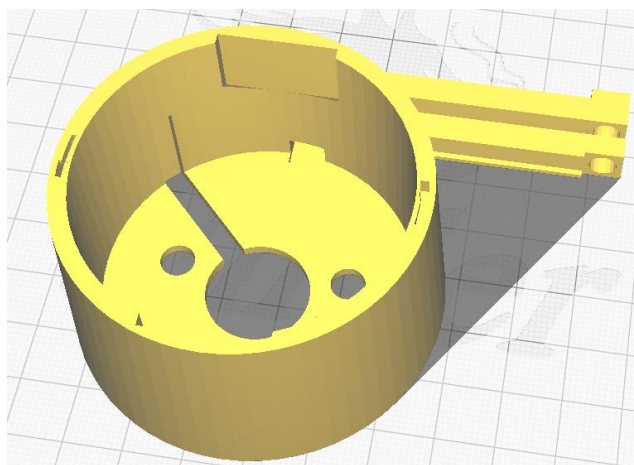
3d modellen til prototypen består af 6 dele, og er designet i Autodesks gratis værktøj Tinkercad¹.

3d modellen er designet sådan at den kan monteres på en dør med et ASSA låsesystem, med vridere på indersiden af døren, 3d modellen bruger de eksisterende skruer i låsesystemet, og kræver derfor ikke yderligere materialer at montere, med undtagelse af del nr. 5 som skal monteres med et stykke dobbeltpåklæbende tape.

Del nr.1 er bunden til indersiden af døren, den monteres under vrideren, der er et stort hul i midten af bunden sådan at vrideren kan dreje uhindret, og to mindre huller som passer til skruerne som holder hele låsesystemet sammen. Derudover er der en slids i bunden og siden, sådan at ledninger fra ydersiden kan komme uhindret forbi vrideren og op til argon-enheden.

I øverste højre hjørne er holderen til dørsensoren, som består af en fotocelle og en LED. Her er der lavet et hul i bunden af huset hvor ledninger fra dørsensoren kan komme uhindret forbi vrideren og op til argon-enheden. På figur 15 kan del nr. 1. ses monteret under vrideren og med dør sensoren monteret.

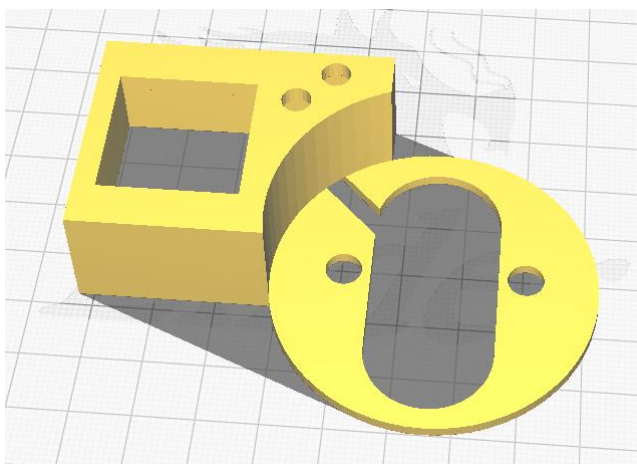
I toppen af del nr. 1 er der 2 indhak til at montere del nr. 4, som er holderen til servo-motor og argon-enheden. I toppen ses også to slidser, som er til at fastmontere del nr. 6, som er låget. Se figur 18 for at se del nr. 4 monteret i del nr. 1



Figur 6: 3D tegning af del nr. 1

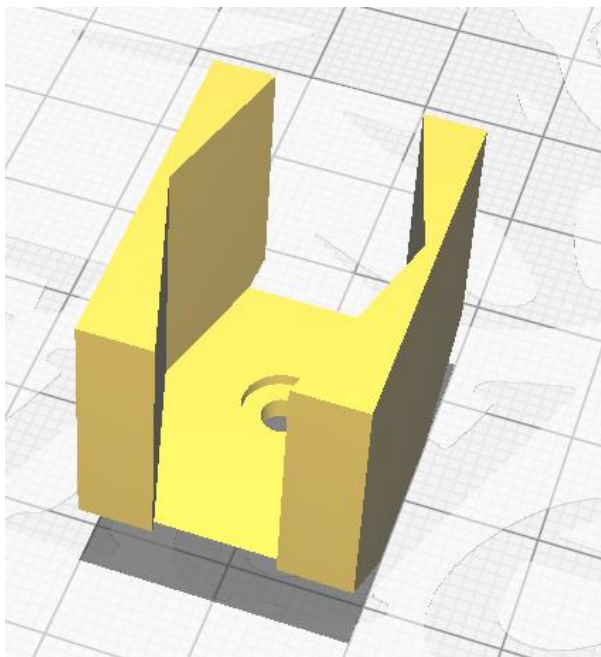
Del nr. 2. er delen til ydersiden. den monteres under dækslet som sidder uden om låsecylindren. I bunden er der et stort aflangt hul som passer over låsecylindren, samt 2 små huller til skruerne som holder hele låsesystemet sammen. Der er også en slids i bunden sådan at ledningerne til de 2 feedback LED'er som sidder i det lille hus i hjørnet med 2 huller til montering af LED'erne. Her er også lavet plads til en pir sensor af typen HC-SR501, som skal detektere om der står en person uden foran døren. Del nr. 2 ses monteret på figur nr. 16.

¹ <https://www.tinkercad.com>

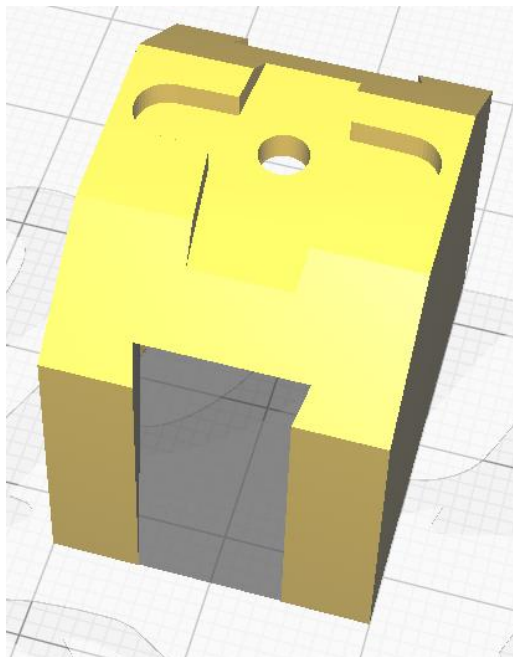


Figur 7: 3D tegning af del nr. 2

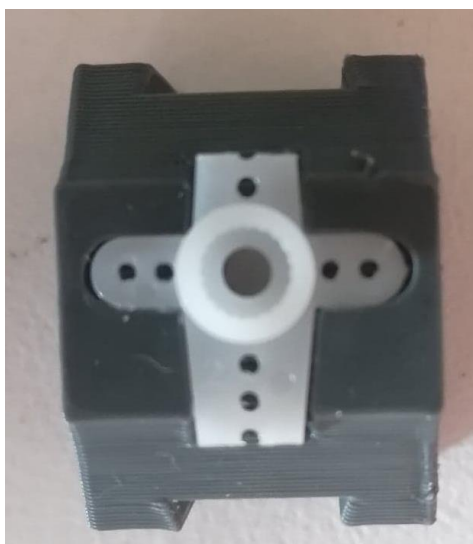
Del nr. 3 er holderen som sidder rundt om vridergrebet, som monteres direkte på servo motoren. En tilskåret udgave af armen fra servo motoren, klikkes direkte i delen, se figur 5, derefter kan delen skrues direkte på servo motoren med den skrue som følger med servo-motoren. På figur 17 kan del nr. 3 ses på servo-motoren, som er monteret i del nr. 4



Figur 8: 3D tegning af del nr. 3 set fra bunden.

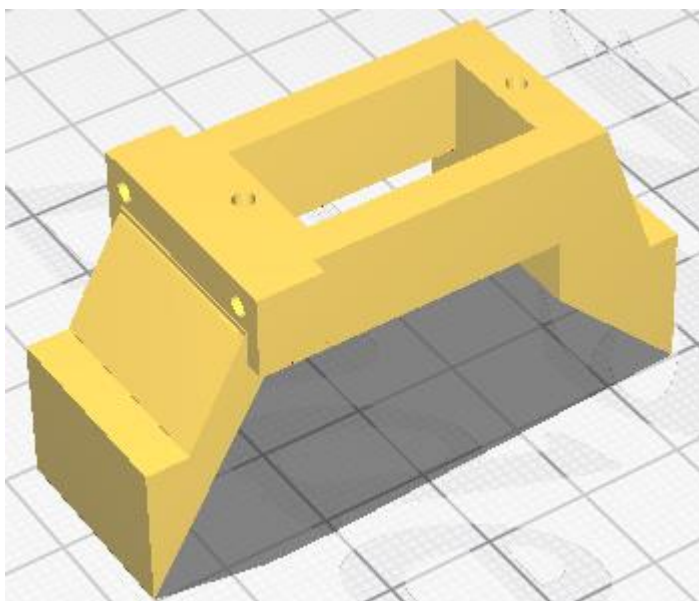


Figur 9: 3D tegning af del nr. 3 set fra toppen



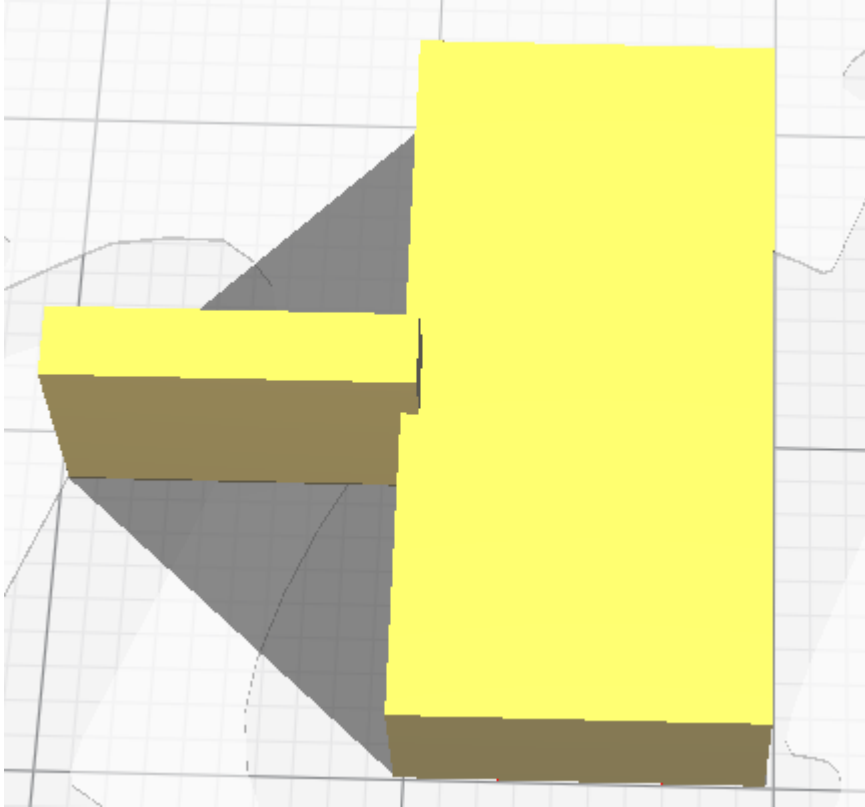
Figur 10: Arm fra servo-motor monteret i del nr. 3.

Del nr. 4. er delen som holder al elektronikken. I toppen er et hul hvori servo motoren skal sidde, der er 2 huller til at fastgøre servo-motoren med skruer. I siden er der 2 huller hvor argon-enheden skal monteres. På figur 17 ses del nr. 4 med servo-motor påmonteret del nr. 3 og argon-enheden med prototype breadboard monteret. Delen med elektronik og del nr.3 monteret, monteres i del nr. 1, sådan at del nr. 3 har fat om vrideren, hvilket kan ses på figur nr. 18



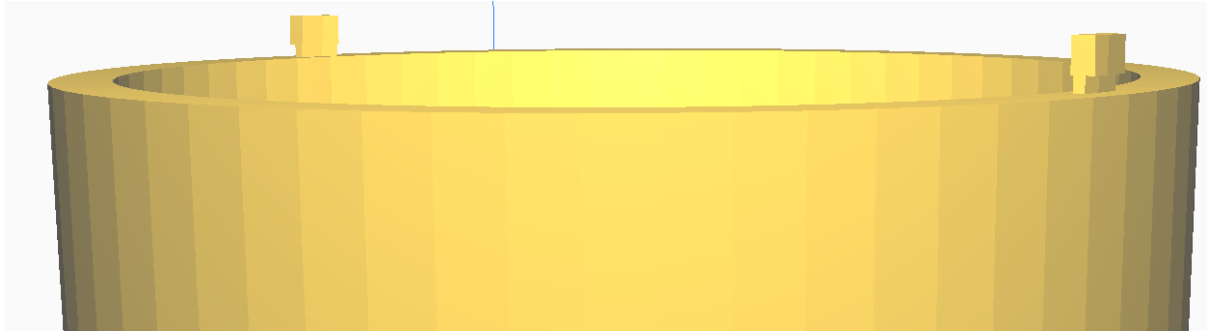
Figur 11: 3D tegning af del nr. 4

Del nr. 5 er delen som lukker af for lyset til fotocellen. Det er den eneste del som kræver noget eksternt for at montere den. Nemlig et stykke dobbeltklæbende tape. Den ses monteret på dørkarmen, på figur 15.

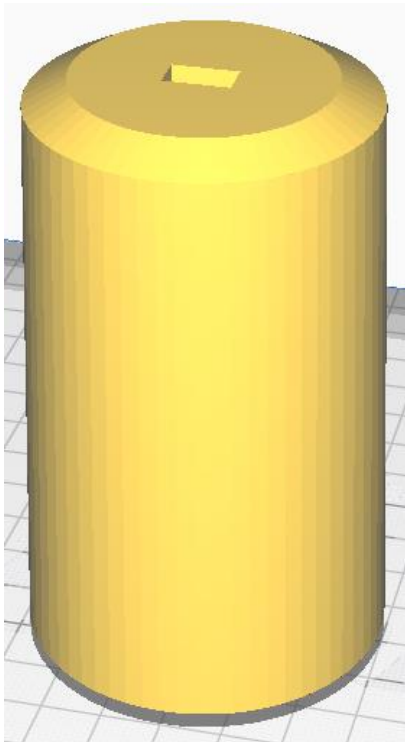


Figur 12: 3D tegning af del nr. 5.

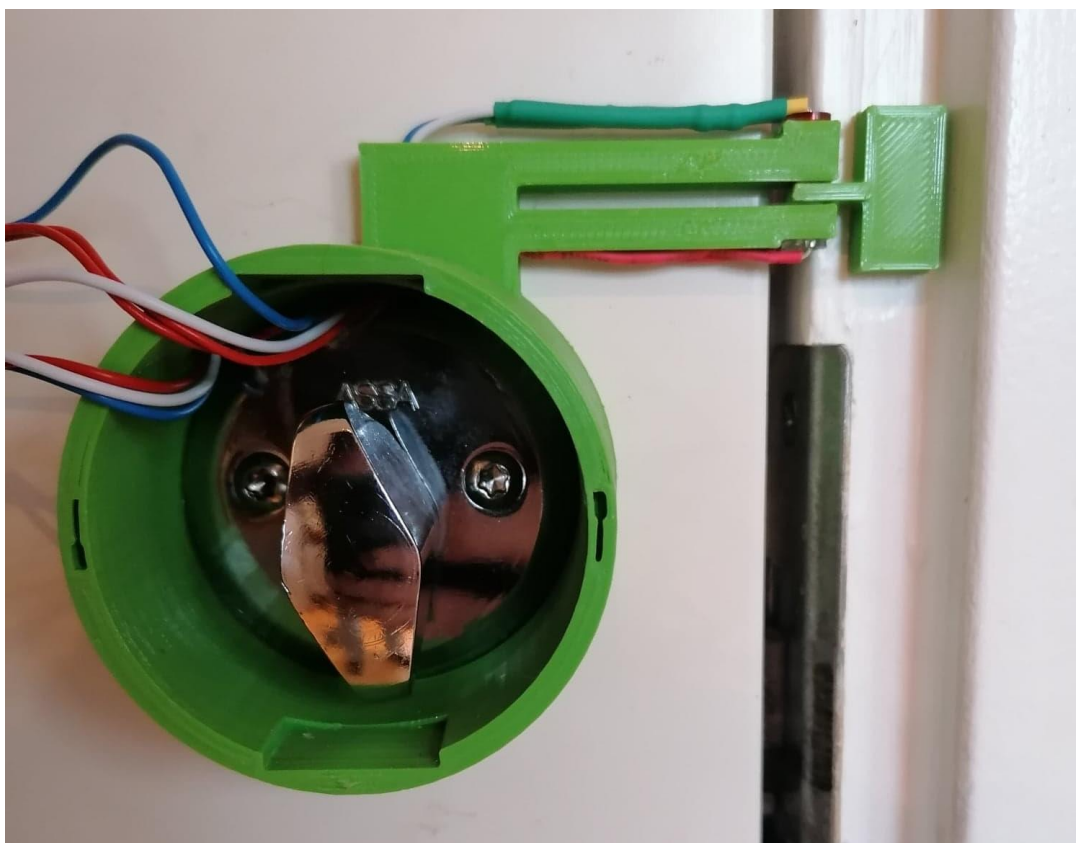
Del nr. 6 er låget som indkapsler hele elektronikken, der er to pinde i bunden, se figur 6, som kan monteres i slidserne i toppen af del nr. 1, og låses fast ved at dreje låget med uret. I toppen er et hul, som passer til et mini USB stik, som forsyner strømmen til elektronikken. Del 6 ses monteret på del nr. 1 på figur 19.



Figur 13: 3D tegning af del nr. 6 zoomet ind på pindene i bunden



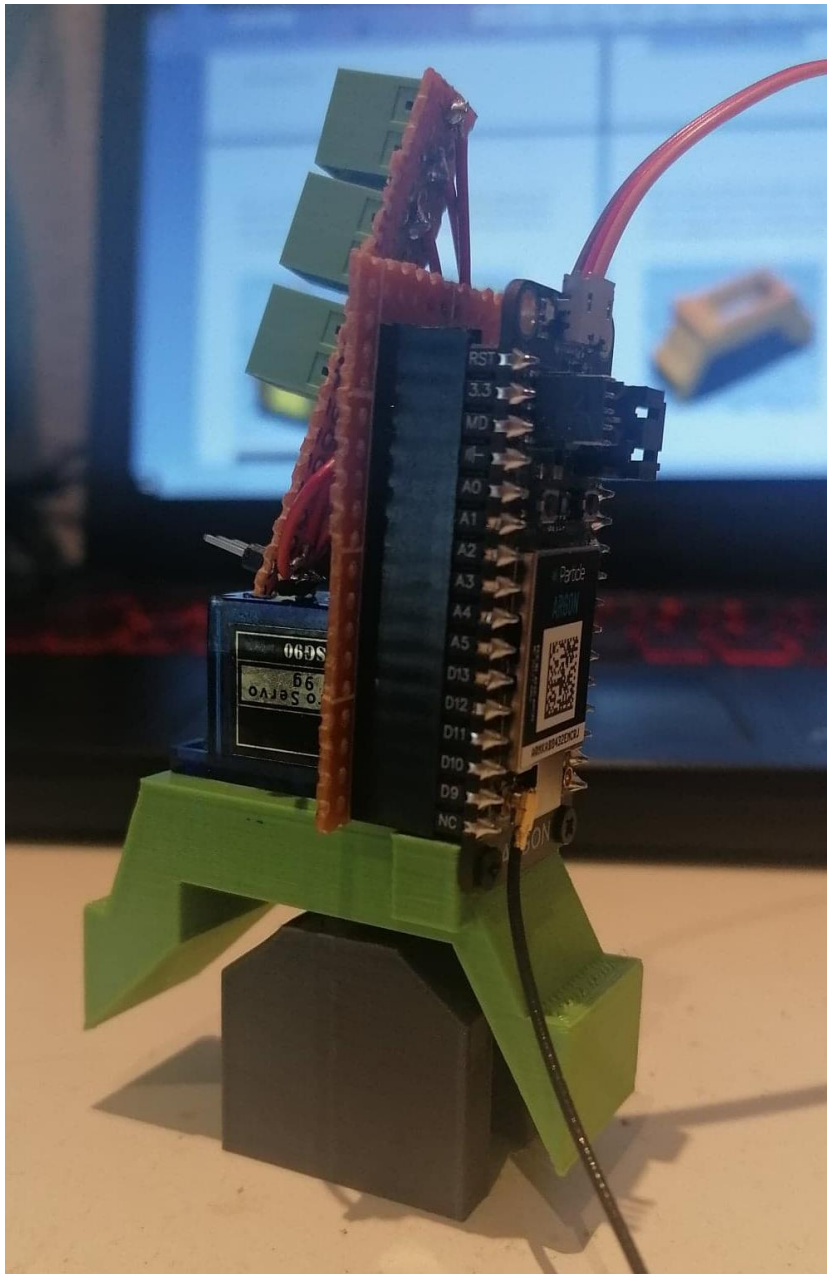
Figur 14: 3D tegning af del nr. 6



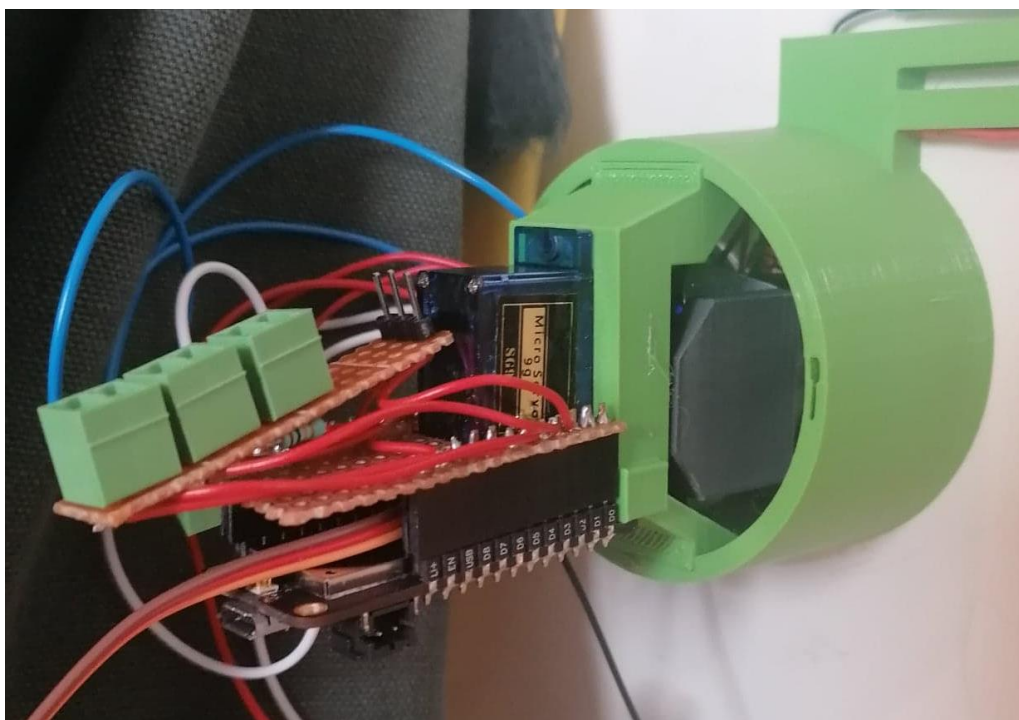
Figur 15: Del nr. 1 monteret over vrideren, samt del nr. 5 monteret på dørkarmen



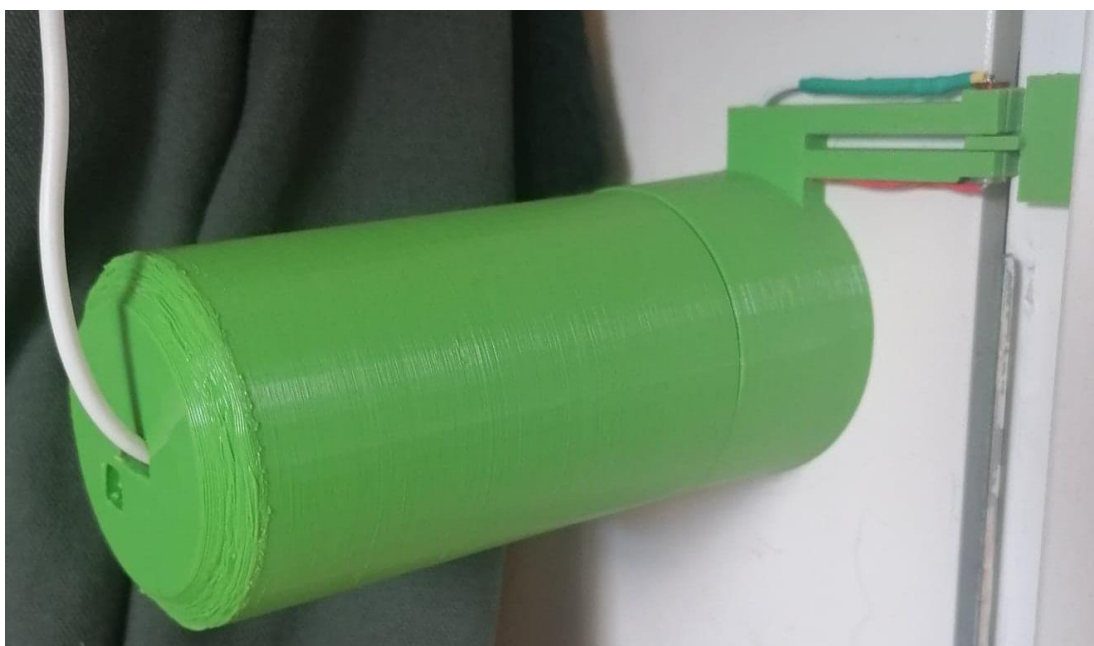
Figur 16: Del nr. 2 monteret under låsecylinder dækslet, med feedback LED'er



Figur 17: Elektronikken monteret på del nr. 4 og del nr. 3 monteret på servo-motoren



Figur 18: Del nr 4 med elektronik monteret på del nr. 1



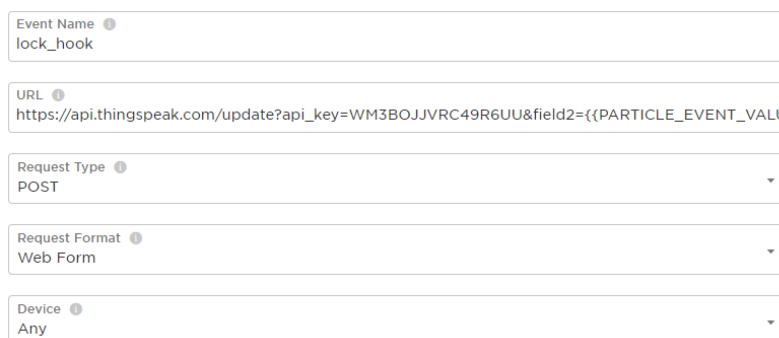
Figur 19: Del nr 6 monteret på del nr. 1

Thingspeak API

Logging:

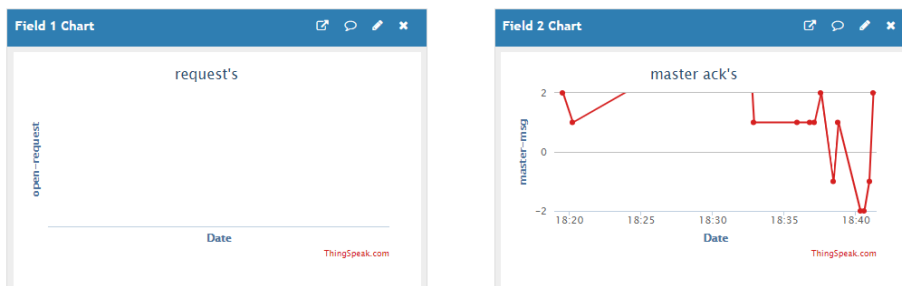
For at få thingspeak logging til at fungere, skal der først sættes en webhook op på particle.io, som tager fra de genererede events. At sætte en webhook op, er forholdsvis simpelt på particle.io, da der er eksisterende templates til dette. I webhook templateen skal der sættes:

- hvilket event skal der reageres på
- URL'en til thingspeaks API, sammen med ens API nøgle, parsing af event data, og hvis nødvendigt, hvilket field man vil sende til
- Hvilket request format man har brug for, dog i vores tilfælde skal vi sende til API'en og bruger derfor en POST request



The screenshot shows the Particle.io webhook configuration interface. It includes five input fields: 'Event Name' with the value 'lock_hook', 'URL' with a long URL containing an API key and a field ID, 'Request Type' set to 'POST', 'Request Format' set to 'Web Form', and 'Device' set to 'Any'.

Figur 20 particle.io webhook template



Figur 21 thingspeaks mulighed for forskellige fields

Vidersending af request's:

For at kunne videre sende request's skal thingspeak's app's sættes op.

For at kunne lave en form for webhook igennem thingspeak, skal der først oprettets en GET request template i thingspeaks app "ThingHTTP", tingene der skal udfyldes her, er generelt det samme som ved particle.io's webhook, dog skal man selv udførde headers og en body.

Name:	notification test
API Key:	19FKRv80335E474
Regenerate API Key	
URL:	https://exp.host/--/api/v2/push/send
HTTP Auth Username:	
HTTP Auth Password:	
Method:	POST
Content Type:	
HTTP Version:	1.1
Host:	
Headers:	Name: Content-Type Value: application/json
Body:	[{"to": "ExponentPushToken[Kij4x4JvAmXNfs9veuN Wwxj]","title": "hello","body": "app test"}]
Parse String:	
Created:	2020-11-10 1:17 pm

Figur 22 thingspeaks webhook template

Dog er det ikke helt optimalt at bruge denne app alene til vores projekt, hvilket er grunden til thingspeak's react app også bliver brugt, hvilket gør at thingspeak kan agere som en webhook.

React appen fungerer ved at man definerer en handling i tilfælde af et event, hvor eventet er hvis der sker noget på en channels feed.

Til dett projekt er det blevet defineret at hvis channel feeden "open-request", modtager en integer (hvilket sker hvis der bliver trykket "Accept" på vores app), så skal den sende requesten videre, til den URL man har defineret i appen ThingHTTP, som så vil være expo's cloud.

React Name	React test
Condition Type	Numeric
Test Frequency	On Data Insertion
Condition	If channel lock_hook_feed (1225216)
field	1 (open-request)
	is greater than
	0
Action	ThingHTTP
	then perform ThingHTTP notification test
Options	<input type="radio"/> Run action only the first time the condition is met <input checked="" type="radio"/> Run action each time condition is met

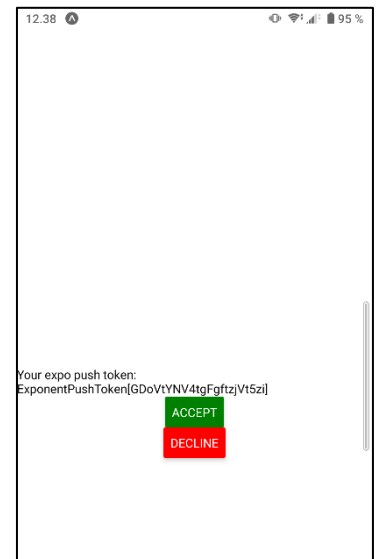
Figur 23 thingspeaks react app

Expo App

Til implementering af koden benyttes Expo's build kommando, som sender koden op på Expo's server som så compilerer koden til .apk filer som så kan hentes ned igen. For at afvikle koden på telefonen herefter installerer man en apk-installer på telefonen som så kan installere appsne.

Master App

I master appen skrives der kun kode i filen App.js. Dette gøres fordi selve master appen er relativt lille og simpel. Der benyttes nogle biblioteker som installeres igennem npm install, som opdaterer Expo mappens package.json som fortæller hvilke libraries der bruges, samt installerer modulerne ind i mappen node_modules. I en app ses der forskel på implementering af funktionalitet og selve GUI'en vist på telefonen. På figur 25 ses koden der genererer det grafiske på figur 24 til højre. Det kan også ses at de to button objekter har en onPress metode, de kalder hver sin funktion, hhv. acceptCurl og declineCurl.



Figur 24 Det grafiske vist på telefonen

```
<View style={styles.viewStyle}>
  <Text>Your expo push token: {expoPushToken}</Text>
  <Button color='green' style={styles.buttonStyleAccept} title={"Accept"} onPress={() => acceptCurl()}></Button>
  <Button color='red' style={styles.buttonStyleDecline} title={"Decline"} onPress={() => declineCurl()}></Button>
</View>
```

Figur 25 Kode til GUI på telefonen

Funktionerne kan ses på figur 26 nedenfor. Funktionerne kører en asynkron metode fra axios biblioteket. De laver en post request på en url og sender et argument. Det er måden telefonen kommunikerer med Particle Argon Boardet.

```
async function acceptCurl() {
  try {
    await axios.post("https://api.particle.io/v1/devices/argon/accept", {arg={access:Accept}})
    console.log("Accepting...")
  }
  catch (error) {
    console.log(error)
  }
}

async function declineCurl() {
  try {
    await axios.post("https://api.particle.io/v1/devices/argon/decline", {arg={access:Deny}})
    console.log("Declining...")
  }
  catch (error) {
    console.log(error)
  }
}
```

Figur 26 De to curl funktioner

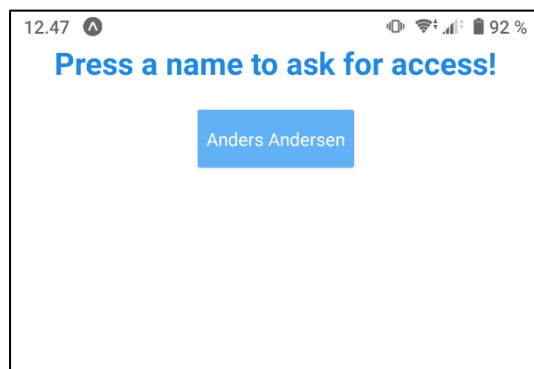
Gæste App

I gæste appen bliver der oprettet en enkelt knap med titlen "Anders Andersen". Knappen har igen en onPress metode som kalder funktionen sendRequest, som igen er en post request med en url og et

argument. Koden til knappen kan ses på figur 27 nedenfor.

```
render() {  
  return (  
    <View style={styles.container}>  
      <Text style={styles.header}> Press a name to ask for access! </Text>  
      <Button info style={styles.buttonStyle} onPress={() => this.sendRequest()}>  
        <Text style={styles.buttonText}> Anders Andersen </Text>  
      </Button>  
    </View>  
  );  
}
```

Figur 27 Kode til GUI på telefonen



Figur 28 Det grafiske vist på telefonen

Funktionen `sendRequest` kan ses nedenfor på figur 29. Dette er sådan telefonen spørger om adgang, som går til Particle Argon Boardet som så siden håndterer dette. Hvis Argonen kan se nogen vha. PIR sensoren sender den en notifikation til Master Appen, som så kan give adgang eller afvise.

```
sendRequest = async () => {  
  try {  
    await axios.post("https://api.particle.io/v1/devices/argon/trigger", {arg={test:test}})  
    console.log("Sending Request...")  
  }  
  catch(error) {  
    console.log(error)  
  }  
}
```

Figur 29 `sendRequest` funktionen

Test/Verifikation

Overordnet

Som vist i introduktions videon, virker hele systemet.

Argon platform

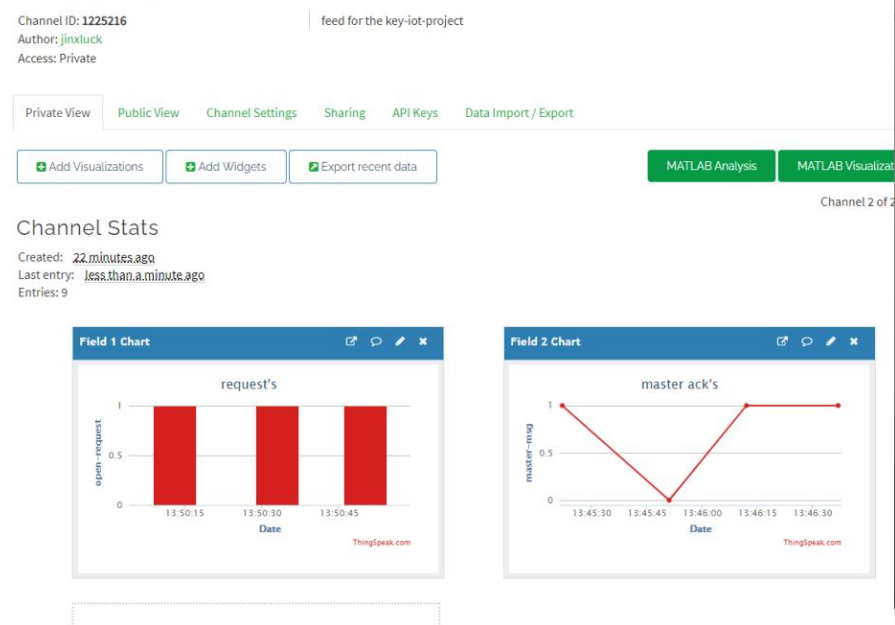
Som vist i youtube video'en nedenunder virker koden til argon platformen.

<https://youtu.be/iGIVc9r3MAQ>

Thingspeak API

På figur 30 kan der ses at thingspeak kan logge events

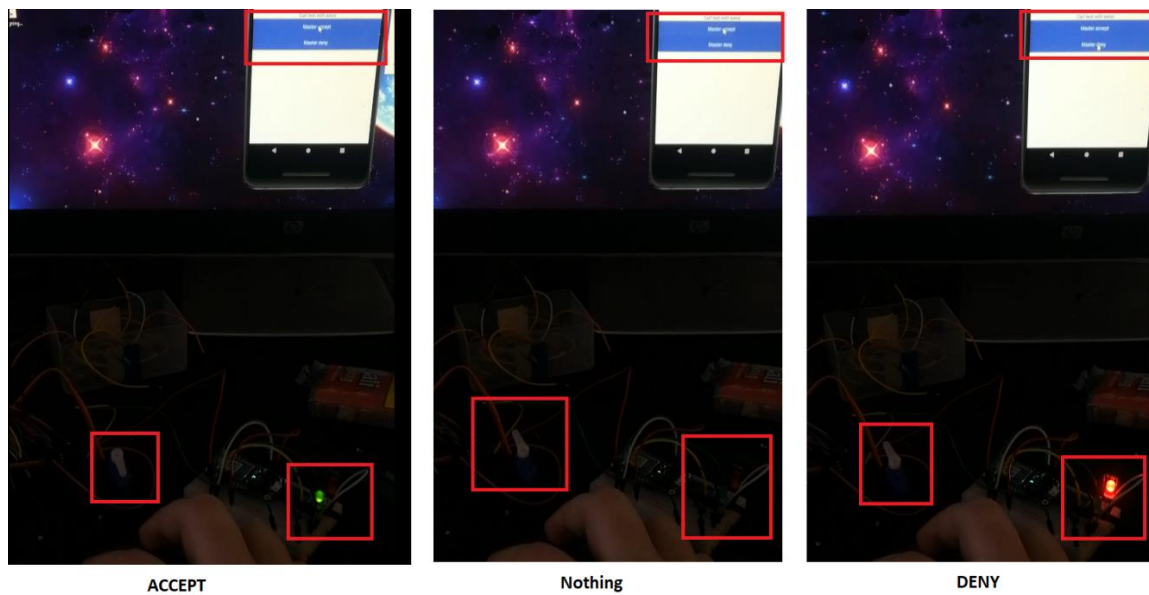
lock_hook_feed



Figur 30 thingspeak test

Expo App

Som set på figur 31 kan der sendes beskeder fra appen, til argonen hvor aktuatoren bliver aktiveret, samt at dioderne virker



Figur 31 overordnet test

Konklusion

Der kan konkluderes at kravene sat under krav afsnittet, fint kan opfyldes, samt at der uden problemer kan laves et IoT projekt, der medtager diverse API's, cloud services, webhooks, og kode sprog (react-native og cpp).

Ud fra dette vil der menes at projektet er færdig gjort til et koncept niveau, dog er der ændringer der ville skulle laves, hvis det skulle tages videre fra koncept til prototype.