E5IOT- Rapport

Remote Roomba (RR)

Aarhus Universitet, School of Engineering
Elektronik, Herning

Gruppemedlemmer:

Malthe Brauer-Nielsen
201807554

Antal tegn:

Indholdsfortegnelse

Introduktion

Denne rapport er udarbejdet for projektet i faget E5IOT, og går ud på at udvikle et IOT produkt som ville kunne bruges i dagligdagen. Det udviklede IOT produkt beskrevet i denne rapport et det såkaldt Remote Roomba, hvis funktion er at gøre en ældre generations Roomba som ikke er trådløs, trådløs. Og derved give brugeren af systemet mulighed for at fjernstarte støvsugeren ved hjælp af sin mobil.

Projektbeskrivelse og Analyse

Remote Roomba ideen bygger på at tidligere generationer af iRobots Roomba robotstøvsugere, ikke kan tilkobles internettet. Da robotterne ikke kan forbindes via et kablet Ethernet, eller forbindes til et trådløst Wi-Fi netværk, så kan robotstøvsugeren derfor heller ikke startes "*remote*". Dette betyder at man kun kan starte robotstøvsugeren, når man er på lokationen hvor robotten er tilkoblet.

For den almindelige dansker, som har en ældre generations iRobot robotstøvsuger, betyder det at hver gang at robotten skal bruges, skal den manuelt startes. Ud fra dette kan der opstå forskellige problematikker der kan være irritable for brugeren ved brugen af robotten;

1. Ingen mulighed for at starte robotten, hvis man har forladt hjemmet og har glemt den.
2. Ingen mulighed for at starte om natten
3. Ingen mulighed for at sætte standard starttidspunkter op (Planlægnings muligheder)
4. Ingen mulighed for at tjekke om robotten faktisk køre, efter man har forladt hjemmet, hvor man har startet robotten.

I dette projekt er hovedfokusset det første nævnte problem, nemlig at robotten kan ikke startes remote, som hvis man glemte at starte den før man tog på arbejde. Samtidig ville der være mulighed for at udvide antallet af features på et senere tidspunkt, til at indebære problem 2 til 4.

Så det endelige System-To-Be, er et ekstra udstyr til støvsugeren der vil kunne fjernstarte robotten over internettet, via en mobil enhed uden at skulle være til stede ved start.

Projekt Specifikation

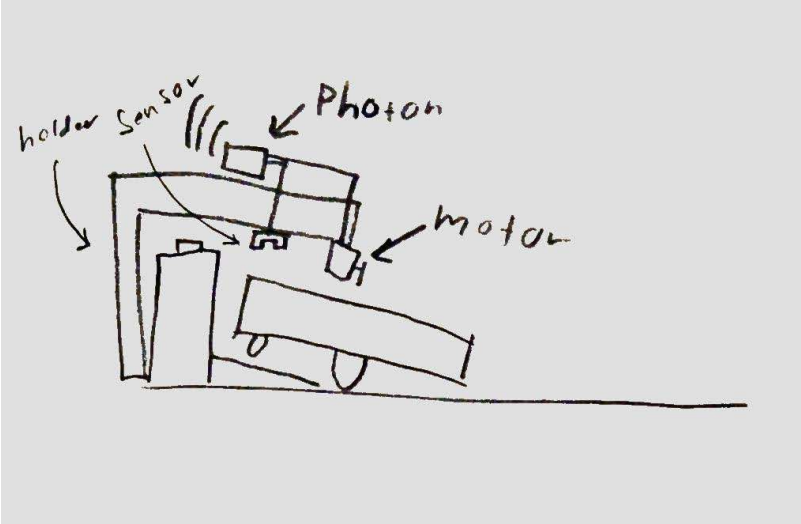
Til projektet er der givet nogle overordnet krav, som produktet / System-To-Be skal indeholde. Dette indebærer at der skal være en eller anden form for sensor, en form for aktuator og så at den kan forbindes til en webtjeneste. En sensor kan være alt der måler på noget fysisk og konvertere det til et signal der kan måles, såsom en knap eller temperatur sensor. En aktuator, kan være alt der konvertere et elektrisk signal til en fysisk reaktion, såsom en LED eller en motor.

I projektets System-To-Be, er der blevet udvalgt hver af disse ting, for at opfylde opgavens krav;

- Web opkobling – Particle board, model: Photon [1]
- Webtjeneste – IFTTT (IF This Then That) [2]
- Sensor – Ultra sonic sensor, model: HC-SR04 [3]
- Aktuator – Servo motor, model: SG90 [4]

Disse ting er blevet valgt, grundet at System-To-Be kommer til at være separat fra robotstøvsugeren, altså at den ikke er monteret på støvsugeren. Årsagen til dette, er at Robotstøvsugeren er designet til at havde en lav profil, så den kan komme ind under f.eks. skabe, sofaer og senge. Så ved at montere noget ovenpå støvsugeren, mistes denne lave profil og muligheden for at komme ind under ovennævnte ting. Derfor skal System-To-Be monteres på støvsugerens ladestation, da den altid vil returnere til denne efter kørsel. Dog

må System-To-Be heller ikke nedsætte ladestationens funktionalitet som at lade eller ved at gemme dens sensor. En skitse af System-To-Be kan ses på Figur 1.



Figur 1 Første skitse af System-To-Be

Formålet med den valgte sensor, er at det så er muligt at detektere, om robotstøvsugeren er til stedet i sin ladestation eller ej. Ved at detektere om robotten er til stede, kan det hurtigt bedømmes om servo motoren skal køre eller ej. Formålet med den valgte aktuator, er derfor, for at kunne trykke på den fysiske start knappen på toppen af robotstøvsugeren, for at sætte den i gang.

Alt kommunikation fra / til brugeren kommer til at foregå via Photon udviklingsboardet, som skal kunne modtage anmodninger sendt af brugeren og sende beskeder tilbage til dem. Disse anmodninger, køre via tjenesten IFTTT [2], der giver muligheden for at sende publish beskeder med et enkelt tryk på en mobil til Photon boardet via nettet og subscribe til andre beskeder.

Analyse af Web opkobling – Photon

Det valgte udviklings board er Particle Photon boardet [5]. Boardet kan kommunikere på 2,4GHz Wi-Fi der giver muligheden for at skabe en trådløs opkobling til internettet. Derudover er den også udstyret med 18 GPIO I/O pins der giver mulighed for analog og digital input output, men også til datakommunikation såsom UART.

Boardet har et typisk strømforbrug på 5V 80mA med Wi-Fi tændt. Når Photon boardet er forsynet via USB connectoren, så kan boardet output 4,8VDC med en max load på 1A i alt. De 18 GPIO pins på boardet er tolerante over for 5VDC input i digital mode, men kan kun output 3,3VDC og har en max belastning på 25mA

Analyse af Webtjeneste – IFTTT

Den valgte webtjeneste er IFTTT [2], der giver muligheden for at lave såkaldte ”Applets”. I disse ”Applets” kan man sætte programmer op, hvor at ved en bestemt interaktion, sker en eller anden form for reaktion. IFTTT [2] er integreret i mange forskellige tjenester og programmer, inklusiv Particle hvor man kan sende publish beskeder som reaktion.

Analyse af Sensor – Ultra sonic sensor

Den valgte sensor er en HC-SR04 produceret af ElecFreaks [3]. Denne sensor måler afstand ved hjælp af ultra lyd på 40kHz, og kan måle afstande fra 2 cm op til 400 cm (4 m) med en præcision på 3 mm.

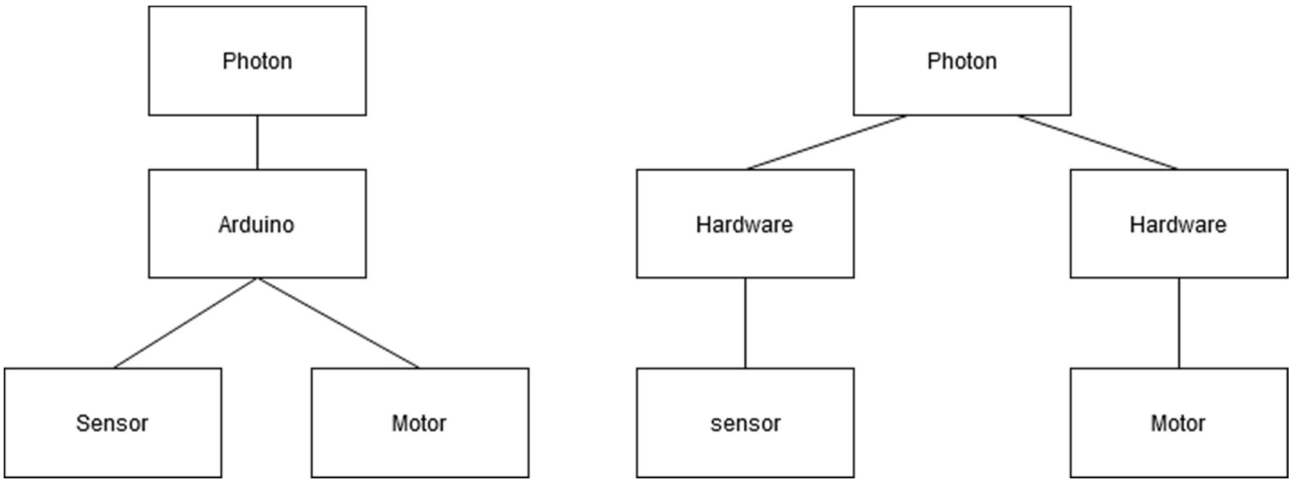
For at sensoren kan operere, skal den bruge en forsyningsspænding på 5V DC, og levere derved også et output på samme spændingsniveau.

Analyse af Aktuator – Servo motor

Den valgte aktuator er en servo motor SG90 [4]. Denne servo motor skal bruge en forsyningsspænding mellem 4,8VDC og 6VDC og en forventet 300-400mA ved høj belastning og 6mA i vente tilstand.

Servo motoren bruger et PWM-signal på samme niveau som forsyningsspændingen (~5V) for at kunne bevæge sig. Den kan rotere 180°, 90° i hver retning alt efter det udsendte PWM-signal som servo motoren modtager. En puls på 1 ms er længst til venstre (-90°), en puls på 1,5 ms giver den midterste position (0°) en puls på 2 ms giver en position længst til højre (+90°).

Ud fra det ovenstående betyder det, at der ved hardwaredelen i System-To-Be; enten skulle laves en driver til både sensor og servo motor, da Photon boarder kun kan output 3,3V og modtage 5V. Eller at der ville skulle bruges et ekstra eksternt board, såsom en Arduino, der kan håndtere 5VDC input og output, som sensor og motor skal bruge under kørsel af System-To-Be. Et overblik hvordan tilkoblingerne evt. kan sættes op er vist på Figur 2.



Figur 2 Forslag til hardware forbindelser med arduino eller ekstern hardware løsning

I dette projekt er der gået med løsningen til venstre, nemlig med en Arduino som driver til sensor og motor. Til dette formål, ville næsten alle Arduino boards kunne bruges, da de næsten alle sammen kan levere 5V output. Så til projektet er der blevet brugt en Arduino MEGA 2560, da dette board var tilgængelig i starten af projektet. Dog er dette udviklingsboard meget stort til formålet, og ville kunne erstattes med en Arduino Uno eller Leonardo.

Requirements analysis

I projektet er der givet 2 forskellige kravs grupper, funktionelle krav og tekniske krav, hvor disse 2 grupper af krav flyder løbende sammen. Ud over disse krav, som var givet med opgavebeskrivelsen, så er der også blevet udarbejdet nogle ekstra krav, som er specifikke for det ønskede System-To-Be. Oversigten over kravene kan ses nedenunder, sammen med hvordan de tiltænkes at kunne blive løst;

Funktionelle krav:

Tabel 1 Funktionelle krav

ID	Krav	Forventet løsning	Anden kommentar
F1.0	The device must be able to connect to the internet	Vha. Photon boardet kan System-To-Be forbinde til internettet via Wi-Fi.	
F1.1	Internet connection shall be via WIFI	--ll--	
F1.2	The device should preferably be able to connect to AU's "AU Gadget network"	--ll--	Photon er blevet lånt af universitetet, og kan derfor forbinde til det givende netværk.
F2.0	Your device must be able to read data from a connected sensor, local to the device	Den valgte sensor vil være forbundet til en GPIO-pin, for at sensor data kan blive aflæst.	
F2.1	a sensor can be anything that quantifies a physical measure, into an electrical signal, such as temperature, light, humidity, presence, movement, magnetism, pollution, etc.	Sensoren bruger ultralyd til at aflæse afstand, og ud fra denne data udsende et elektrisk signal.	
F3.0	Your device must be able to control an actuator	Aktuatoren vil blive kontrolleret via output på en GPIO-pin.	
F3.1	An actuator can be anything that translates an electrical signal into a physical quantity, such as, motors, servos, valves, heaters, displays, lamps, etc.	Aktuatoren i dette tilfælde vil være en servo motor.	
F4.0	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.	Net tjenesten IFTTT [2] vil blive brugt, hvor en applet med deres "button widget" vil blive lavet.	Vil kunne starte processer på platformen via Wi-Fi.
F5.0	Your software and hardware design must be shared	Dette er delt via GitHub [6]	
F5.1	You must create a public github account, and add relevant project files here	--ll--	Relevant data og information vil blive delt i GitHub Repo'et [6]

F5.2	Hardware documentation, schematics, datasheets and pcb layouts are to be uploaded in pdf format	--ll--	Relevant data og information vil blive delt i GitHub Repo'et [6]
F5.3	Software files are to be uploaded in raw source code format, e.g. .C, CPP, .h, .py, etc.	--ll--	Relevant data og information vil blive delt i GitHub Repo'et [6]

Tekniske krav

Tabel 2 Tekniske krav

ID	Krav	Forventet løsning	Anden kommentar
T1.0	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.	Photon-platformen forventes at blive brugt sammen med Arduino-platformen	Dette krav dækker også krav F1.0-F1.2 i funktionelle krav.
T1.1	The platform shall have Wifi connectivity	--ll--	Ligner krav F1.1 i funktionelle krav
T1.2	The platform shall have available digital or analog I/O con connect sensors and actuators	Arduino-platformen har GPIO-pins der kan forbindes og kommunikere med sensor og aktuator.	

Specifikke krav

Tabel 3 Specifikke krav

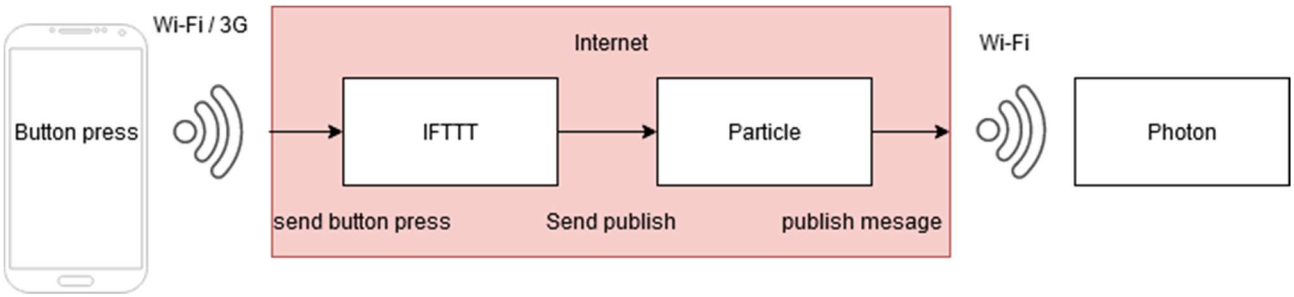
ID	Krav	Forventet løsning	Anden kommentar
S1.0	System-To-Be skal kunne starte Roombaen fra brugerens mobile enhed.	Ved at bruge IFTTT fås denne mulighed	
S2.0	System-To-Be skal ikke forringe Roombaen og dens ladestanders funktionalitet	Skal sættes på ladestanderen med åbne sidder, for ikke at gemme ladestanderen.	
S3.0	System-To-Be skal sende status retur til brugeren efter start.	IFTTT kan sende brugeren notifikationer.	

System Design

I System-To-Be er det blevet valgt, at Photon boardet [1] er den del, som kan forbindes til nettet, via en Wi-Fi forbindelse hvor particle platformen kan kommunikere med webtjenesten IFTTT [2]. I IFTTT [2], skal der laves to "Applets", en som kan publish en besked, som Photon boardet [1] kan subscribe til, og en anden til at modtage beskeder fra photonen.

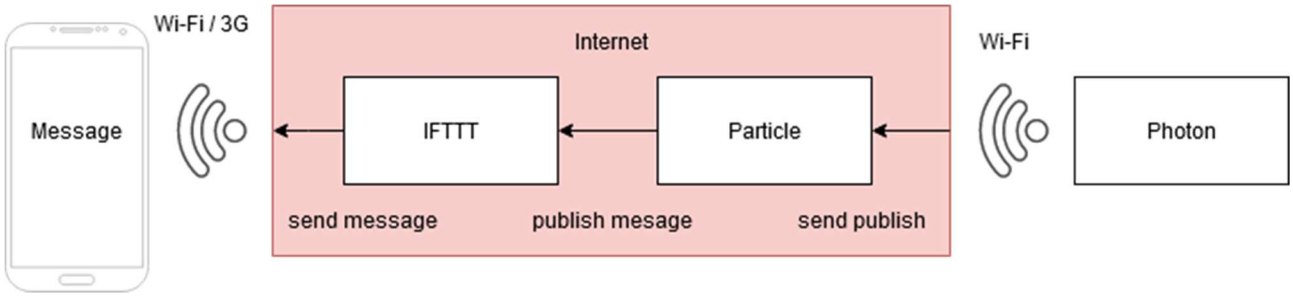
For at gøre det så nemt som muligt for brugeren at bruge System-To-Be'et, så skal den første "Applet" bruge deres "button" feature. Denne feature gør at brugeren får en enkelt knap, som ved interaktion

sender en publish besked fra brugerens telefon via IFTTT til Particle, som Photon boardet opfanger. Dette er illustreret på Figur 3.



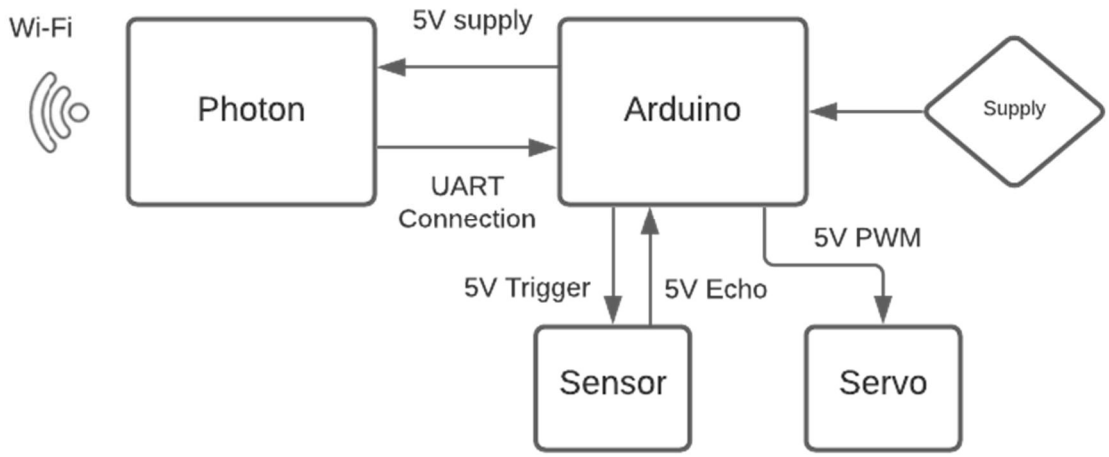
Figur 3 Brugerinteraktion for at publish besked til Photon

Den anden ”Applet” gør det modsatte af den første ”Applet”, som er beskrevet ovenover. Her er det IFTTT [2] som laver en notifikation på brugerens telefon, når Photonen publisher en besked som IFTTT [2] har subscribet på.



Figur 4 Giv besked til brugeren fra Photon

Ud fra den beskrevne hardware i de tidligere afsnit, så kan et interface diagram udarbejdes. Dette er vist på Figur 5. Heri er det blevet valgt at bruge UART til datakommunikation i mellem Photonen og Arduinoen som bruger en baudrate på 9600. Når Photon boardet reagere på en published besked fra IFTTT [2] som vist på Figur 3. Så vil boardet skulle kommunikere med Arduinoen over UART, for at starte processen med at starte Robotstøvsugeren. Interface diagram over hardwaren i systemet, er vist på Figur 5



Figur 5 Interface diagram over hardwaren

En af årsagerne til at denne løsning er blevet valgt over at lave et eksternt print til kontrol af sensor og servo. Var at dette gav en mulighed for at anvende noget af den viden lært fra tidligere semestre angående Co-design og seriel kommunikation i praksis.

Kommunikationsprotokollen som bliver brugt til at kommunikere mellem de 2 boards er UART. Ekstra til denne protokol skal det fastlægges hvilket data som sendes frem og tilbage i mellem de 2 boards, for at kommunikationen ikke fejler. Til dette vælges der forskellige integer / Heltals værdier, alt efter hvad der skal gøres, såsom at starte robotten. De valgte værdier og korresponderende aktioner er valgt ud fra hvilke scenarier der kan opstå, og er vist i Tabel 4 og Tabel 5 for data frem og tilbage mellem Photonen og Arduinoen;

Tabel 4 Data fra Photon til Arduino

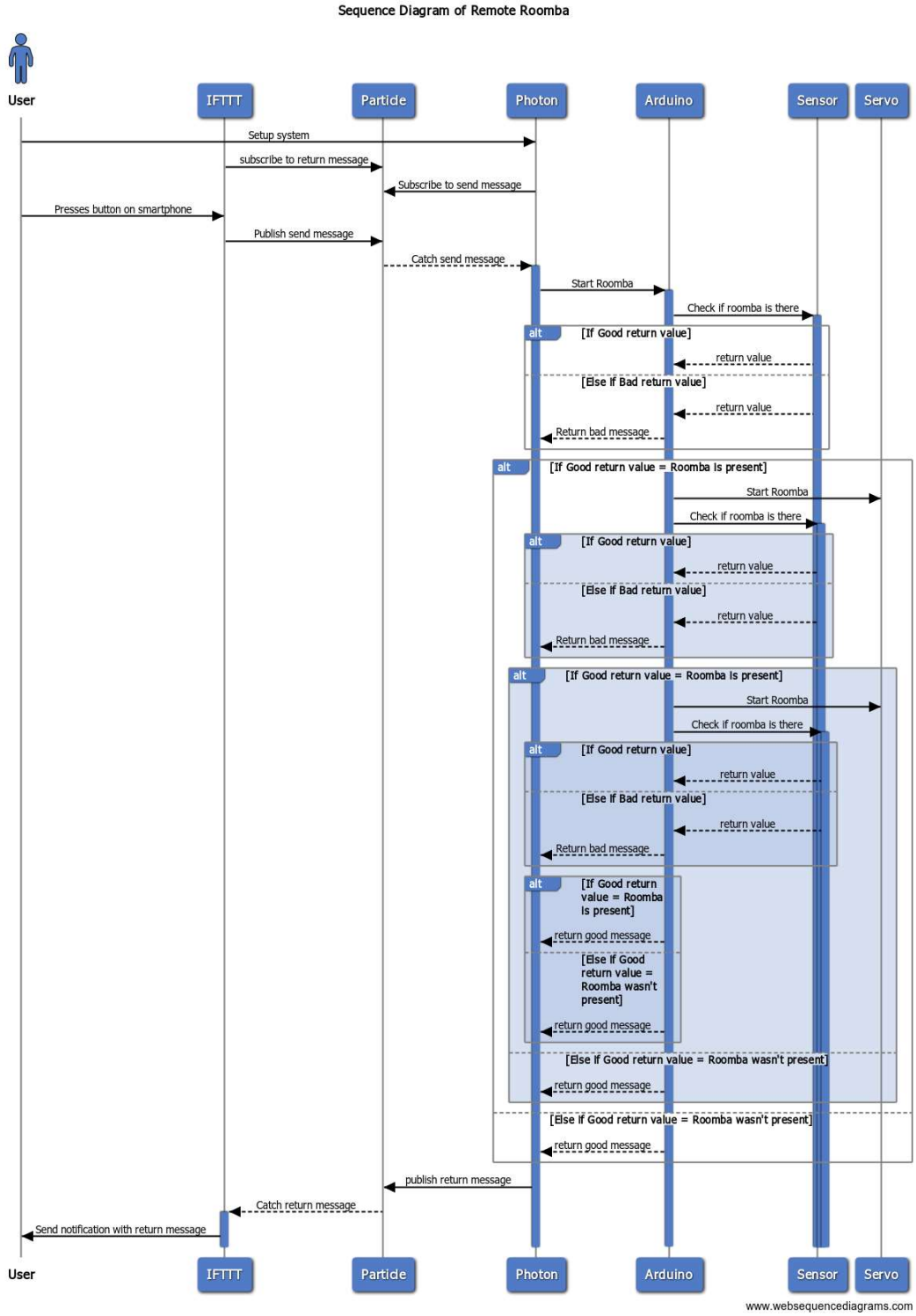
Integer / heltal	Aktion	Værdi
1	Start Roomba	1

Tabel 5 Data fra Arduino til Photon

Integer / heltal	Scenarier	Retur værdi
1	Robotten er startet	1
2	Robotten er prøvet startet	2
3	Robotten er der ikke	3
4	Sensor fejl / Anden fejl	4

Med data fra Photon til Arduino, kan der kun sendes en kommando, og det er den med at starte robotten. Dog er der 3 forskellige scenarier der kan opstå vedr. robotten, hvor den enten starter normalt, ikke vil starte, eller at den ikke er til stede. Derudover, er det også lagt ind at der kan opstå en uventet fejl, som derved også kan håndteres og returneres til brugeren. Denne analyse over de forskellige scenarier, har også lagt grunden til sekvensdiagrammet for systemet sammen med analysen over den generelle opsætning af

systemet, og bruger interaktioner. Sekvensdiagrammet for systemet kan ses på Figur 6 og viser det flow der er fra at brugeren trykker på knappen på hans telefon, til at beskeden kommer tilbage igen.



Figur 6 Sekvens diagram for System-To-Be

Implementering

Både Photonens og Arduinoens kode blev udviklet i C++, og bygger om det samme princip, hvor ”main()” funktionen er gemt nede i core biblioteket. Og i stedet implementeres en ”setup()” funktion, hvor alt initialiseres, såsom pins og subscribes. Samt en ”loop()” funktion hvor selve ens implementering er lavet, denne funktion er en uendelig løkke, og køre derfor for evigt, indtil boardet slukkes eller en fejl opstår.

Photon

Implementeringen i Photonen, består som ovennævnt af ”setup()” og ”loop()” funktionerne. Derudover, så er der implementeret yderligere 4 funktioner, som er vist nedenunder;

1. Modtager funktion til UART-kommunikation
2. Afsender funktion til UART-kommunikation
3. Modtager funktion til det subscribet emne på particle platformen
4. Afsender funktion til at publish data på particle platformen

Disse 4 funktioner, er hoveddelen i photon programmet. I ”setup()” sættes subscrib kaldet op, som kalder modtager funktionen, og derefter sender en UART besked afsted. I ”loop()” funktionen bruges UART modtageren, og Particle publish funktionerne. Her ventes / polles på UART-modtager funktionen, indtil der kommer et input, hvor efter der sendes en publish afsted ud fra den modtaget besked.

På Figur 7 kan selve UART modtager funktionen ses. Her tjekkes der først om serielporten er i tilgængelig, og hvis den er det, læser den på porten. Her indlæses en karakter / char ad gangen, og gemmes i en buffer indtil at en ”\n” modtages. Efter dette så konverteres de modtaget karakterer til en integer i funktionen ”Process_received()”, og værdien gemmes.

```
62  // UART Receiver.
63  void uart_receive()
64  {
65    delay(1000);
66    // Read data from serial
67    while (Serial1.available())
68    {
69      if (readBufOffset < READ_BUF_SIZE)
70      {
71        char c = Serial1.read();
72        if (c != '\n')
73        {
74          // Add character to buffer
75          readBuf[readBufOffset++] = c;
76        }
77        else
78        {
79          // End of line character found, process line
80          readBuf[readBufOffset] = 0;
81          process_received();
82          readBufOffset = 0;
83          waiting = 0;
84        }
85      }
86      else
87      {
88        Serial.println("readBuf overflow, emptying buffer\n");
89        readBufOffset = 0;
90      }
91    }
92  }
```

Figur 7 UART Modtager funktionen

Efter at havde modtaget en besked, stoppes der kortvarigt med at blive læst på UART’en, for at publish en besked til particle platformen. Et udklip af dette kan ses på Figur 8, hvor den modtaget værdi gemmes

lokalt, og køres igennem en switch statement med 4 cases for de fire forskellige scenarier vist i Tabel 5 Data fra Arduino til Photon, hvor der bliver published en besked med en streng for det givende resultat.

```
117  void publish_message()
118  {
119    int x = receivedValue;
120    Serial.printlnf("Recieved %d from Arduino", x);
121
122    switch (x)
123    {
124    case 1:
125      Particle.publish("Roomba", "Roomba deployed.", PUBLIC);
126      break;
```

Figur 8 Udklip af Publish funktionen fra Photonen

Når photonen reagere på et emne den er subscribet til på particle platformen, så kalder den med det samme UART afsender funktionen, som kan ses på Figur 9. Her bruges en simpel funktion til at afsende en af mulighederne vist på Tabel 4 over UART forbindelsen til Arduinoen.

```
94   // UART send
95   void uart_send(int data)
96   {
97     Serial.printlnf("Send Data; %d\n", data);
98     // Send data over UART (Serial1) to Arduino.
99     Serial1.printlnf("%d", data);
100  }
```

Figur 9 UART afsender funktionen, kaldes fra published_received funktionen

Arduino

Ligesom på Photonen, så er der både en ”setup()” og en ”loop()” funktion på Arduinoen. Ud over dem, så er der 4 andre funktioner, som bruges til at interagere med sensor, servoen og UART’en, og kan ses nedenunder.

1. Funktion til at interagere med sensoren
2. Funktion til at interagere med servo motoren
3. Modtager funktion til UART-kommunikation
4. Afsender funktion til UART-kommunikation

Som før på Photonen, så polles der på UART-modtageren indtil der modtages noget, og da begge platforme bygger på C++, og har mange af de samme core-funktioner, så er UART-modtageren ens på Photonen og Arduinoen. Samt bruger UART afsenderen også de samme funktioner med ”Serial.printlnf” til at sende beskeder. Men afsenderen på Arduinoen bruger samme switch form som på Figur 8, for at returnere status.

Men når Arduinoen modtager noget på UART’en fra photonen, så køres hele tjekket igennem som kan ses på sekvensdiagrammet på Figur 6. I denne proces bruges både sensoren og servo motoren, til at tjekke efter Roombaen og starte den.

På Figur 10 kan man se selve kode logikken til brugen af sensoren. Her bliver sensoren kørt 10 gange, da der til tider kommer fejl målinger, så for at undgå disse tages 10 målinger, hvor gennemsnittet beregnes til sidst. Men først, så nulstilles trigger pinnen, som er den der bruges til at sende signalet afsted med. Derefter så sættes pinnen høj i 10 mikrosekunder for at sende et lydsignal afsted. Derefter så venter processen på at modtager pinnen går fra at være lav til at være høj. Når dette sker, så starter en timer, som

måler rejsetiden for lydsignalet som stopper igen når pinnen bliver lav igen. Med denne tidsmåling, kan afstanden beregnes og ligges til tidligere målinger.

```
147   for (int x = 0; x < 10; x++) {
148     // Clears the trigPin condition
149     digitalWrite(trigPin, LOW);
150     delayMicroseconds(2);
151     // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
152     digitalWrite(trigPin, HIGH);
153     delayMicroseconds(10);
154     digitalWrite(trigPin, LOW);
155     // Reads the echoPin, returns the sound wave travel time in microseconds
156     duration = pulseIn(echoPin, HIGH);
157     // Calculating the distance
158     measureddistance += duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
159     delay(100);
160   }
161   measureddistance = measureddistance / 10;
```

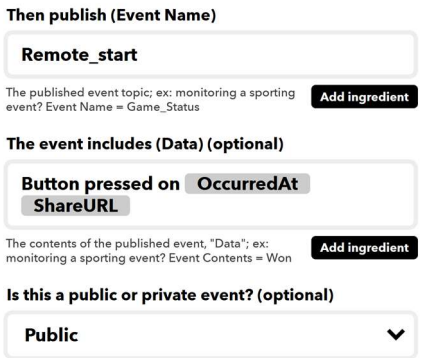
Figur 10 Logikken til sensoren

Ud fra den gennemsnitlige afstand som blev beregnet, kan der så til sidst blive vurderet om Roombaen er til stede eller ej. Måden dette gøres på, er ved at tjekke om afstanden er højere eller lavere end 3 cm, hvor disse 3 cm manuelt er blevet målt da robotten var i ladestationen.

Til servo motoren, var der et indbygget bibliotek som gjorde brugen af motoren meget simpelt. Da det kun er vinklen motoren skal dreje til der skal gives. I denne motors tilfælde er dette en vinkel imellem 0 og 180 grader som kan gives til funktionen. For at komme ud over problemer med at robotten lagde sig i en deep sleep når den venter, og derfor først skal vågne efter tryk på start. Så blev funktionen implementeret med at antallet af tryk fra servo motoren kunne bestemmes, da der kan opstå scenarier hvor funktionen skal kaldes igen. I de tilfælde hvor den skal kaldes igen, kan robotten være vågnet fra deep sleep, så for mange tryk kan resultere i at robotten stoppes igen.

IFTTT

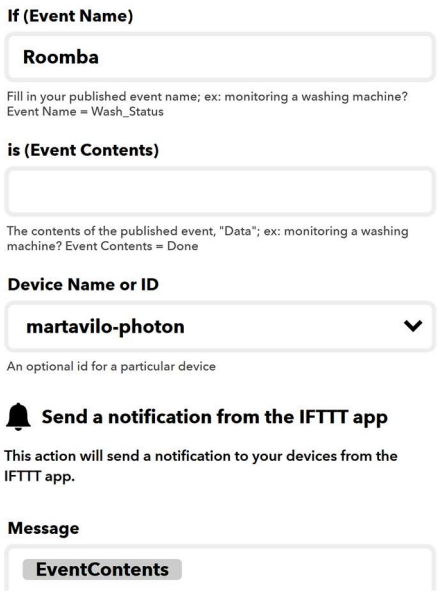
IFTTT tjenesten blev koblet sammen med den particle bruger hvor photonen var oprettet, og der blev Implementeret to applets. Den ene applet kan ses på Figur 11, og giver brugeren en visuel knap på hans / hendes mobil som kan trykkes på. Når denne knap trykkes på, så laver den en publish med det givende event navn på particle platformen.



Figur 11 Knap applet som publisher event på particle

Den anden applet kan ses på Figur 12, og er den applet der giver en notifikation når der bliver published et event fra photonen. Her sættes hvad event navnet er, evt. specifik data som kom med eventet og hvilken enhed der skulle havde lavet dette publish. Da eventet kan indeholde 4 forskellige strenge af data, så

tjekkes indholdet af dataene ikke for at gøre denne ”Applet” universel. Til selve notifikationen gives den modtaget data som notifikationstekst.



Figur 12 Notifikations applet

Test/Verifikation

Til test og verifikation af system-To-Be er kravene i afsnittet ”Requirements analysis” blevet gennemgået, for at sikre at systemet opfyldte alle krav. Hertil er alle funktionelle krav og tekniske krav blevet opfyldt og bekræftet løbende i løbet af opgaven, igennem analyse og design af systemet.

De specifikke krav nævnt i Tabel 3 Specifikke krav, er dog funktionaliteter som skal testes for at bekræfte at de fungere efter hensigten. Til Dette er der blevet udført en såkaldt Big Bang integrations test, hvor hele systemet testes samlet. I denne test, bliver alle funktionaliteter testet for at sikre at de opstillede krav er opfyldt.

Til dette blev der udført en række manuel test beskrevet i Tabel 6. Heri er der beskrivelsen af testen, krav for at testen kan blive udført korrekt, test trinene og så om testen fungerede og blev godkendt.

Tabel 6 Testcases for Big Bang Integrations test

Test ID	*Testbeskrivelse*	*Test requirements*	*Udførelses trin*	*Forventet resultat*	*Godkendt (Y/N)*
T1.0	Brugeren trykker på knappen på sin mobile enhed, venter et kort øjeblik hvor efter servo motoren begynder at bevæge sig.	1 Robotten er i ladestationen.	1. Trykker på knappen på mobilen 2. observere motoren efter bevægelse	Servoen bevæger sig	Y
T2.0	Brugeren får en notifikation på sin mobil, omkring at robotten blev forsøgt startet, men stadig er til stede.	1 At der er blevet trykket på knappen på mobilen. 2 At robotten stadig er i ladestationen	1. Trykker på knappen på mobilen 2. observere motoren efter bevægelse	Der kommer en notifikation med teksten *”Roomba present but*	Y

		efter servo motoren har kørt.	3. Efter motoren har kørt afventer en notifikation. Der kan gå op til 5 minutter 4. Aflæs notifikationen	*not deployed."*	
T3.0	Brugeren får en notifikation på sin mobil, omkring at robotten blev startet.	1 At der er blevet trykket på knappen på mobilen. 2 At robotten er i ladestationen før der trykkes på knappen.	1. Trykker på knappen på mobilen 2. observere motoren efter bevægelse 3. Efter motoren har kørt afventer en notifikation. Der kan gå op til 5 minutter 4. Aflæs notifikationen	Der kommer en notifikation med teksten *"Roomba deployed."*	Y
T4.0	Brugeren får en notifikation på sin mobil, omkring at robotten ikke er i ladestationen.	1 At der er blevet trykket på knappen på mobilen. 2 At robotten ikke er i ladestationen før der trykkes på knappen	1. Trykker på knappen på mobilen 2. observere motoren efter bevægelse 3. Efter motoren har kørt afventer en notifikation. Der kan gå op til 5 minutter 4. Aflæs notifikationen	Der kommer en notifikation med teksten "Roomba was not present."	Y
T5.0	Robotten og ladestationen kan stadig se hinanden med System-To-Be monteret på ladestationen.	1 At robotten køre i mod ladestationen	1. Start robotten pegende i mod ladestationen. 2. Lad den køre i mod ladestationen 3. Observer 4. Udfør test step 1-3 flere gange med andre indgangsvinkler til ladestationen	Robotten køre i mod ladestationen, stoppe og vender om.	Y

De udførte test og deres resultater, er blevet video dokumenteret og kan ses her:

*INDSÆT LINK HER

Konklusion

Målet med opgaven og System-To-Be, var at udvikle og implementerer et system, som gjorde det muligt at fjernstarte en ældre generations Roomba, som ikke selv kan forbinde til nettet. Ved hjælp af IFTTT platformen og particle platformen, er det blevet muligt at oprette en 2 vejs kommunikation. Hvor en bruger kan sende anmodninger afsted til System-To-Be, og få en respons tilbage fra System-To-Be omkring dens status.

Kravene givet i projektoplægget for systemet er blevet, igennem analyse og implementering, inkorporeret og besvaret i System-To-be. Samt har en Big Bang integrations test bevist at systemet fungerer efter hensigten og ikke forringer Støvsugerens eller ladestationens funktionalitet.

Fremtidigt arbejde

Med System-To-Be på plads og fungerende, er der stadig mange mulige features / forbedringer som kan blive implementeret.

En af de helt store, som ikke nåede at blive implementeret er reducering af strømforbruget. Som systemet er lige nu, så er både Photon og Arduino konstant tændt, og poller på UART modtage funktionen. En måde som kunne hjælpe med at reducere strømforbruget, ville være at bruge sleep funktionaliteten i de 2 boards.
I Arduinoens tilfælde, ville dette være at bruge interrupts til at vække boardet fra sleep når der sker noget på UART forbindelsen.
I photonens tilfælde, ville dette være at undersøge om boardet tjekker efter tidligere published beskeder hvis wifi slukkes og tændes igen, da den så ville kunne vækkes i et givent interval for at tjekke efter tidligere beskeder. Dette vil selvfølgelig gå ud over respons tiden på at starte støvsugeren, som skal tages med op til overvejelse.

Andre features som kunne hjælpe brugeren af systemet ville være; at Arduinoen sender notifikationer afsted til brugeren efter at Roombaen har kørt nogle gange, for at minde brugeren om at han / hun skal huske at tømme støv beholderen på støvsugeren. Eller gøre børsten rent fra hår og andet skidt som sætter sig, og kan forringe rengøringen. Disse 2 ville skulle komme med forskellige intervaller, da børsten hyppigere skal gøres rent end at beholderen skal tømmes. En anden notifikation som kunne blive sendt, ville også være når robotten kommer retur, så brugeren ved at rengøringen er afsluttet.

Andre features som ville være oplagt at implementere ville være at give brugeren muligheden for at planlægge hvornår støvsugeren skulle køre. Dette ville kunne gøres ved at IFTTT aflæser en Google kalender, med kalenderbegivenheder med hvornår støvsugeren skulle køre. Dette kunne f.eks. være om natten, eller et andet standart tidspunkt på dagen, så brugeren ikke selv skal huske at starte den.

Bibliografi

[1] Particle, »Particle.io,« N/A N/A N/A. [Online]. Available: https://docs.particle.io/datasheets/wi-fi/photon-datasheet/. [Senest hentet eller vist den 22 09 2020].

[2] IFTTT, »IFTTT,« IFTTT, N/A N/A N/A. [Online]. Available: https://ifttt.com/home. [Senest hentet eller vist den 05 10 2020].

[3] ElecFreaks, »Sparkfun,« N/A N/A N/A. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. [Senest hentet eller vist den 05 10 2020].

[4] Tower Pro, »http://www.ee.ic.ac.uk/,« N/A N/A N/A. [Online]. Available: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf. [Senest hentet eller vist den 22 09 2020].

[5] Particle, »Particle Docs,« 16 08 2020. [Online]. Available: https://docs.particle.io/datasheets/wi-fi/photon-datasheet/. [Senest hentet eller vist den 05 10 2020].

[6] M. Brauer-Nielsen, »E5IOT - GitHub,« 13 09 2020. [Online]. Available: https://github.com/Gubimand/E5IOT. [Senest hentet eller vist den 13 09 2020].

[7] ReQtest, »ReQtest,« ReQtest, 25 10 2018. [Online]. Available: https://reqtest.com/requirements-blog/requirements-analysis/. [Senest hentet eller vist den 22 09 2020].

[8] Sparkfun, »Sparkfun,« N/A N/A N/A. [Online]. Available: https://www.sparkfun.com/products/15569. [Senest hentet eller vist den 22 09 2020].

Bilag

[1] Opgavebeskrivelse

Purpose

The purpose of this project is to apply certain theoretical topics, from the course, onto a specific project. The project runs through the entire semester. You must design an IoT artefact, composed of embedded hardware, embedded software, sensors, and actuators.
You IoT artefact shall be capable of performing a meaningful task, better, that a similar device, that is not connected to the internet.
That could be anything from "a smart plant watering systems, that takes into account the weather forecast, sunshine hours, etc, " to a " streaming music device that plays music depending who is present" or perhaps an intelligent industrial sensor, or something completely different.

Documentation

A written project report must be handed in, including as minimum

- Introduction
- Project description
- Requirements analysis
- System design
- Implementation
- Test/Verification
- Conclusion

The project report must be uploaded to digitalEksamen, in pdf format. In the report you are welcome to include links to videos, that contain supporting information, such as demonstrating your artefact is working, selected verification tests executed or other relevant supporting information. The pdf report must contain the primary information.

Teamwork

You are allowed to work in teams of up to 3 persons. In your project documentation, is must be clearly stated what topics the individual team members have worked with

Requirements

Functional requirements

1 The device must be able to connect to the internet

1.1 Internet connection shall be via WIFI

1.2. 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

2.1 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

3.1 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

5.1 You must create a public github account, and add relevant project files here

5.2 Hardware documentation, schematics, datasheets and pcb layouts are to be uploaded in pdf format

5.3 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.

1.1. The platform shall have Wifi connectivity

1.2. The platform shall have available digital or analog I/O con connect sensors and actuators

Timeframe

week 37 - approval of your project description and design requirements
week 38-43-47- Short presentations of the project progress
week 50 - upload of your report, final source code and documentation to github etc.

At the exam

The exam is oral
At the exam you are expected to give a short (maximum 5 minuttes) presentation of your project.

Since we have been reading the report, you should focus on presenting the key findings, or selected interesting topics at the exam, perhaps a novel way of solving a technical problem.

You can expect the project report to weigh app. 50% of your exam grade, and question in the remaining curriculum to weigh the remaining 50%