Rapport Grupp 1

En ljus ide

Teknisk dokumentation  
HT18

Grupp 1  
Linus Kasper.   
Elias Nyqvist,  
Simon Ögaard Jozic

Teknik  
Joakim Flink  
Andreas Jillram  
Daniel Åkerlund

Innehåll

[1 Funktionsbeskrivning 3](#_Toc531169302)

[2 Systembeskrivning 4](#_Toc531169303)

[3 Detaljbeskrivning 5](#_Toc531169304)

[3.1 Delsystem 1 Frontend 5](#_Toc531169305)

[3.2 Delsystem 2 Armaturen (Solid Works) 6](#_Toc531169306)

[3.3 Delsystem 3 Wifimanager och Avståndssensorn 8](#_Toc531169307)

3.4 Delsystem 4 Backend ……………………………………………………………………..11

Bilagor

[Bilaga 1 Kod](#_Toc531168031) Frontend

[Bilaga 2](#_Toc531168032) Backend

**Bilaga 3 Wifimanager**

# Funktionsbeskrivning

Det första vi gjorde i projektet var att komma överens vilka funktioner vår lampa skulle ha, detta var det vi kom fram till.

1.Ändra ljusstyrka med en avstånsdsensor och en app.

2. I appen kan man också ändra balansen mellan varmt och kallt och stänga av både sensorn och de kalla och varma ledstripparna.

3. En walkthrough animation i hur man använder avståndssensorn.

4. Ett utryms effektivt sätt att frakta vår produkt

Tanken var att vi skulle göra en inloggnings skärm vilket vi har gjort med vi har dock inte hunnit få den att fungera då vi prioriterade annat.

Man kan via en avstånsdsensor, en mikrokontroller och några rader arduino-kod styra lampans ljus. Lampans “indirekta ljus” styr man via avståndssensorn och det ljuset som riktas mot lampans botten kan styras via en app.

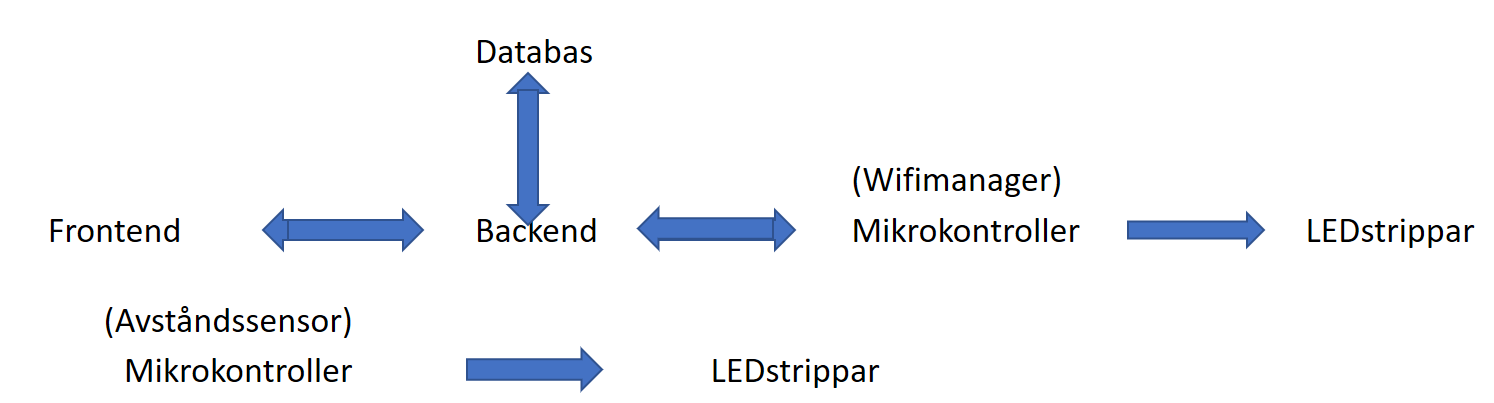
I appen kan man ändra temperaturen på ljuset, stänga av avståndssensor funktionen och stänga av de kalla och varma ledstripparna. Du kan även få en så kallad walkthrough i hur man gör om man vill dimma det indirekta ljuset.

Hela armaturen ska vara printbara och kunden ska kunna köpa en mindre låda där delarna finns, för att sedan sammanfoga delarna till den klara lamparmaturen.

# Systembeskrivning

Vårt system fungerar på sätt att du kan styra det indirekta ljuset med hjälp av en avstånds sensor. Den är kopplad till en mikrokontroller som vi har programmerat med c++. Vi kan alltså med hjälp av handrörelse ändra ljusstyrkan på de varma och kalla ledstripparna. Dock så är denna mikrokontroller inte kopplad eller med i hela systemet. Den är utanför (se bild).

Vi kan också ändra ljusstyrkan på de varma och kalla LEDstripparna som går nedåt med hjälp av en app. System där fungerar på så sätt att om vi drar i en slider eller drar i en knapp i frontenden så skickas de värdena i Json format till backenden med en så kallad ”PATCH” funktion. Sedan från backenden så skickas värdena och sätts in i mysql kod så att värdena uppdateras i databasen. Sedan så kollar hela tiden wifimanager koden som ligger på mikrokontrollern om värdena ändras. Den gör då en ”GET” funktion som går genom backenden och backenden hämtar då värdena från databasen och skickar sedan tillbaka värdena till mikrokontrollern där de sedan kan användas till att dimma varm och kallt samt att stänga av ljuset och att stänga av sensorn.



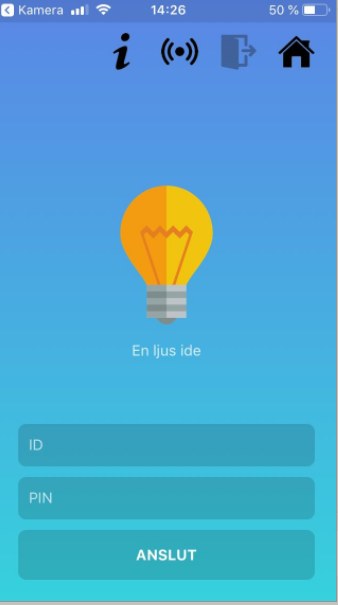
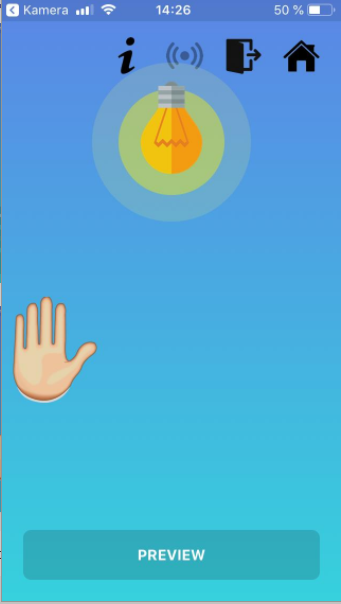
# Detaljbeskrivning

## Delsystem 1 Frontend

Frontenden är det man ser i projektet. Vi har valt att använda en app. En app där vi dels kan få en walkthrough på hur man använder vår avstånsdsensor samt en skärm där vi har sliders och knappar där vi ska kunna ändra styrka och förhållandet mellan varmt och kallt ljus. Det går också att stänga av och sätta på lampan om man så vill med en knapp.

Vi har 3 skärmar i vår app.

1 2 3

1, Det här är vår loggin skärm. Det är denna skärmen du hamnar på när du öppnar appen för första gången. Det fungerar dock inte att logga in då vi bara gjorde denna skärm för att lära oss mer och för att det är en snygg första skärm att hamna på. För att komma till skärm nummer 2 trycker du på huset.

(Se bilaga 1.A).

TouchableOpacity omringar Image (huset) så om du klickar (onPress) navigerar den dig till hemskärmen vilket är skärm 2 (ScreenHome). Bilden har en stylesheet och sourcen valde vi att ha en uri istället för require då det ska funka för alla oss tre att starta appen då vi har olika mobiltelefoner.

När du är på skärm 2 så ska du kunna ändra värden på varmt kallt och power. De värdena skickas sedan till databasen.

(Se bilaga 1.B).

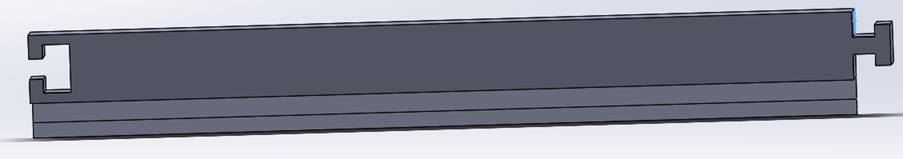
Först gör vi variabler för alla olika states (tillstånd) vi vill att slidersen ska ha. Sedan sätter vi in dessa i ett jason objekt som vi gör en variabel av. Varför vi gör en variabel är för att det ska gå att console logga den. Det vill vi kunna göra för att lättare felsöka den. Sedan tar vi urlen som vi har definerat tidigare. Sedam talar den om att det är en ”PATCH” funktion vi ska göra och att vi vill skicka i jasonformat. Sedan gör den om allt till json och console loggar det färdiga resultatet och skickar det. Sedan finns det också en funktion som fångar error ifall vi får det och talar om vad som är fel.

(Se bilaga 1.C)

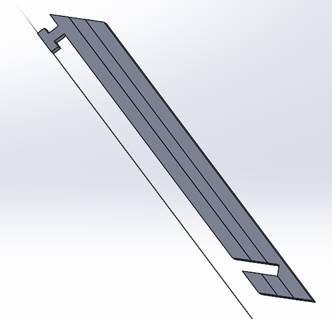
I vår view har vi en slider som vi har laddat ned. Vi börjar med att ge den en style. Vi säger att den ska hoppa ett steg i taget med ”step={1}”. Vi ger den ett maximalt värde till 100. Vilket betyder att den går från 0 till 100 och bara 1 steg i taget. Vilket motsvarar värden mellan 0-1024 egentligen. Vi säger också att värdet är lika med vad värdet på hot är. Sedan när vi slidar på den så ändrar den värdet hot till vad det nya värdet hot är. När vi sedan släpper den så går den till funktionen ovan och lägger in allt i json objektet osv. Det säger vi med

## Delsystem 2 Armatur

När vi gjorde armaturen valde vi att printa hela armaturen. Anledningen till det var att kunna sälja armaturen i ett litet paket. Paketet skulle vara enkelt att frakta då allt tomrum i armaturen skulle vara utanför ekvationen. Sammanfogningen gjorde vi universell så att en caddad del skulle ha samma sammanfogning som den nästa. Detta skulle göra det enklare för både kunden och oss då vi slipper räkna ut flera sammanfogningar. Det slutade med 5 olika delar som vi var tvungna att printa men antalet delar blev 24.



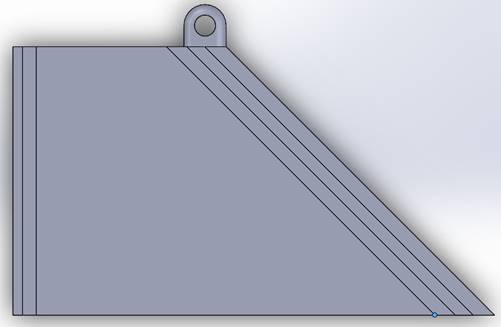
Detta är de två universella sammanfogningarna som vi använder i alla delar. Det vi behövde var ett hållbart sätt att sätta ihop två bitar utan lim eller tejp då vi inte fick använda det och inte häller printa hela armaturen samtidigt då det inte får platts.



Högst upp ser du en liknande sammanfognings del som ovan men den är på en 45 graders vinkel och ska sitta ihop med en annan del. Indenten du ser är gjord för att hålla upp vårt glas. För att denna delen av armaturen kommer sitta i en 45 graders lutning så tog vi av en liksidig triangel på båda ändarna.



Denna delen liknar den ovan men är mindre för att göra det möjligt att sammanfoga armaturen. Delen nedan är den som skapar strul när det gäller att sammanfoga armaturen då vinklarna är intill omöjliga att sammanfoga när ena änden är fast. Då gjorde denna del det enkelt att låsa armaturen.

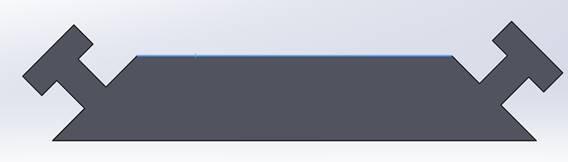


För att brädden på vår armatur var större än 140 mm krävdes det att vi delade upp sidan i två delar. De två delarna sammanfogas som de andra men har en ring där kedjan kan fästas och även sammanfogning i en lutning där den första delen sätts fast.

Denna del behövde vi göra ett få antal utav då varje hade små skillnader som vilket håll lutningen sitter i och vilken typ av fogningen är.



Detta är en av delarna som ledstripparna sitter på den gjorde vi för att få vinkeln på ljusets riktning, och har två likadana sammanfogningar så vi slapp göra två olika och kunde bara spegla delen



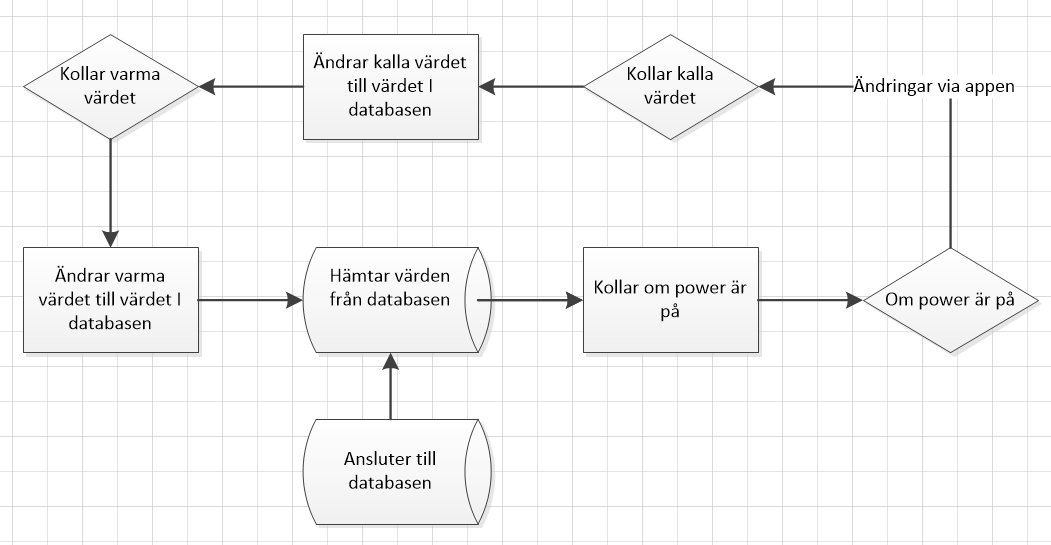
Denna del kunde vi då också spegla och räddade oss mycket arbete. Den håller också i ledstrippar men så att de är riktade direkt ner på marken för att få mer spridning på ljusets riktning.

Armaturens form vi valde har sammanfogat en indent i toppen av armaturen där arduino och kopplingen var tänkt att få plats, men utrymmet var mindre än förväntat och arduinon var ihopkopplat ganska stor så man ser tyvärr sladdar osv sticka ut på toppen. Vi valde bland annat formen till armaturen pågrund av att den gör att ljuset får en stor spridning men också för att den har en indent som vi hade tänkt lägga mikrokontrollarna vilket fungerade. Men det blev dock inte så snygg då man ser kablar och sådant.

Vi använde aluminiumtejp för att reflektera mer ljus både direktljuds och indirekta ljuset använde vi tejpen för att öka styrkan.

Avståndssensorn kunde vi sätta dit på sidan av armaturen då en gammal del var fel caddad och hade hål i sig där vi bara krävdes att slipa lite för att få in den.

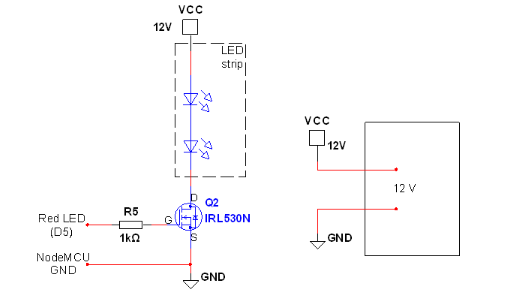
## Wifimanager och Avståndssensorn

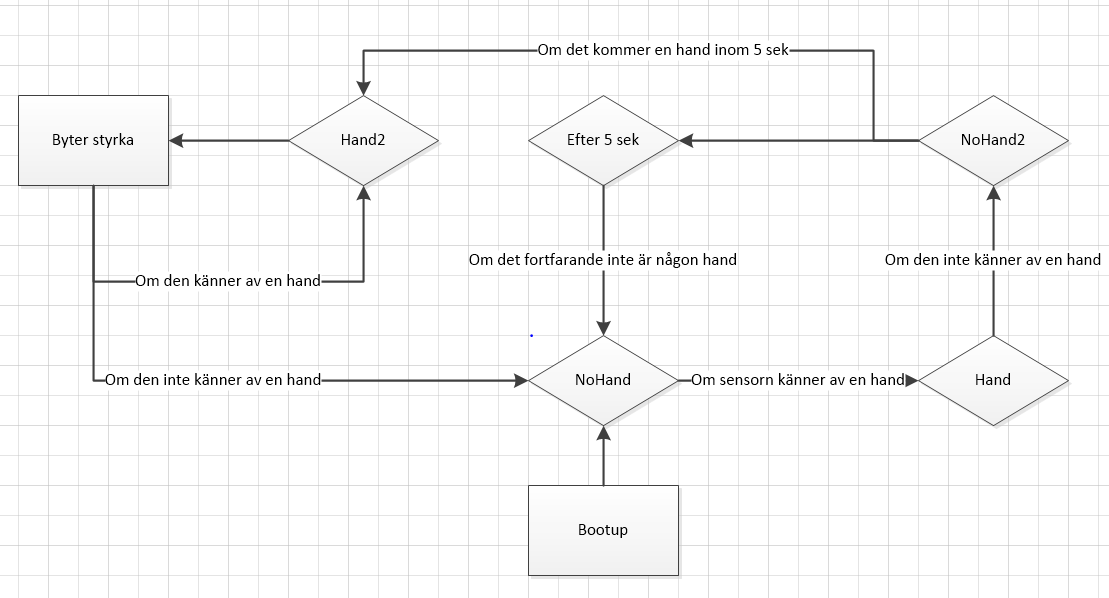


(se bilaga 3.A)

När man startar programmet ansluter man till databasen. Därefter så hämtas värderna där, och programmet checkar om booleanen ”power” är sann. Om den är sann, så kan man ändra värdet via appen (om boolen är falsk så stängs lamporna av). Om vi förutsätter att Power är true så kommer först det kalla värdet att kollas, och sedan att låsas och därmed uppdatera ljusstyrkan. Därefter så kollar programmet vad det varma värdet är, och ändrar sedan de varma lampornas ljusstryka.

Såhär kopplade vi med mikrokontrollern som vi laddade upp koden på,

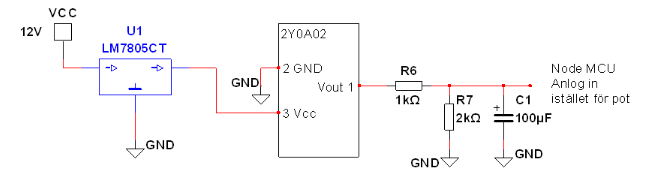




Så här såg det ut när vi ska ändra det indirekta ljuset med hjälp av en avståndssensor. När programmet startas så hamnar programmet i statet ”NoHand”. Här letar sensorn efter ett värde större än minimivärdet, dvs en hand mellan signalen. När den känner av detta går den till statet ”Hand”. Här görs motsatsen, den kollar efter att handen ska försvinna. Dessa funktioner har vi lagt in för att vi inte vill att någon ska ändra på värdet genom att t.ex stå eller gå genom lampan av misstag. För att konfirmera att man vill ändra värdet behöver man därför göra en viss handrörelse. Nästa state, NoHand2 är tidsbestämd via en forloop. Detta innebär att en hand måste komma in i sensorn inom 5 sekunder (detta gjorde vi som sagt för att kräva extra konfirmation). Om en hand kommer in i sensorn inom 5 sekunder så kan man med hjälp av höjningar och sänkningar ändra på värdet.

(Koden finns i github).

Sähär kopplade vi på mikrokontrollern där vi laddade upp koden på,



## Backend

Det är vår backend som fungerar som mellanhanden mellan allting. Ingen annan del i projektet är ihopkopplad med varandra. Med alla delar är ihop kopplade till backenden. Vi börjar med frontenden, vi får där från användaren in värden som via en PATCH funktion skickas i jsonformat till backenden. Den skickar det sedan till databasen där de nya värdena sparar. Sedan så gör mikrokontrollern en GET funktion där den vill hämta värdena i tabellen. Då skickar den en GET request till backend, den hämtar då värdena i databasen och skickar sedan värdena till mikrokontrollern.

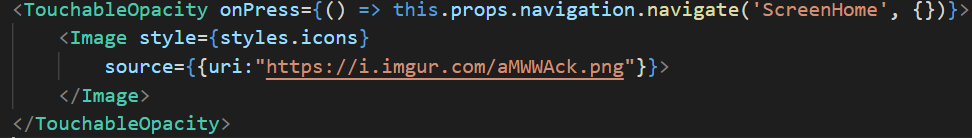
(Se bilaga 2.A).

Detta är vår patch funktion. Vi gör först en array där vi skriver att vi tar bodyn från req vi får in och tilldelar den name, cold, hot osv. Sedan console loggar vi arrayen för att lättare kunna felsöka. Sedan gör vi en funktion där vi skriver i MySQL kod att vi vill uppdatera de olika kolumnerna med respektive body vi fått in för de kolumnerna i requesten. Obs här måste man komma ihåg att man börjar räkna från 0 och inte 1 i arrays. Sedan skriver vi att om det gått fel ska den skriva ut fel annars ska den skriva ut resultatet.

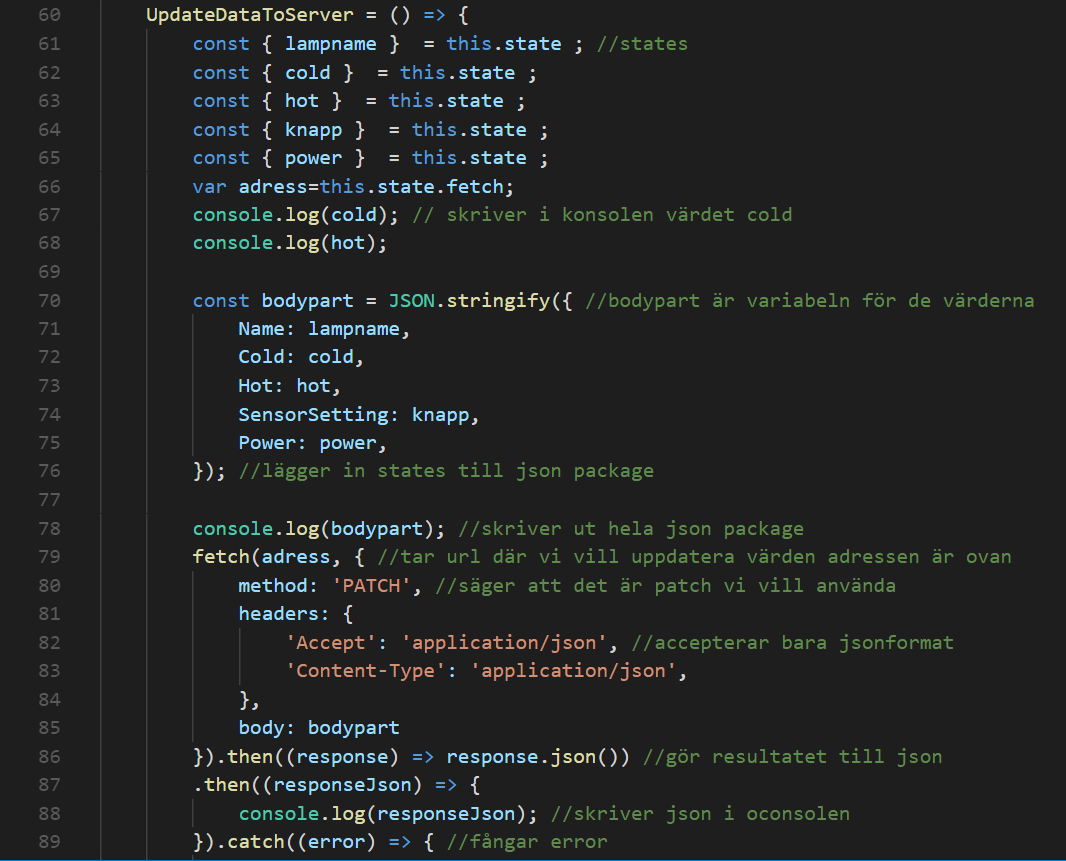
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Base | Url | Function | Body parameter | Description | Returned |
| /products | / | GET |  | Hämtar alla värden | String {} alla namn, meddelande |
| /products | /A | GET |  | Hämtar alla värden från ”A” | String {} hämtar ”A”, meddelande |
| /products | / | PATCH | String, int, boolean  {Name:  Cold:  Hot:  Power:} | Uppdaterar ”A” | String meddelande |

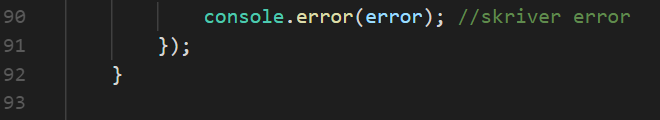
Bilaga 1 Kod Frontend

A, Navigation

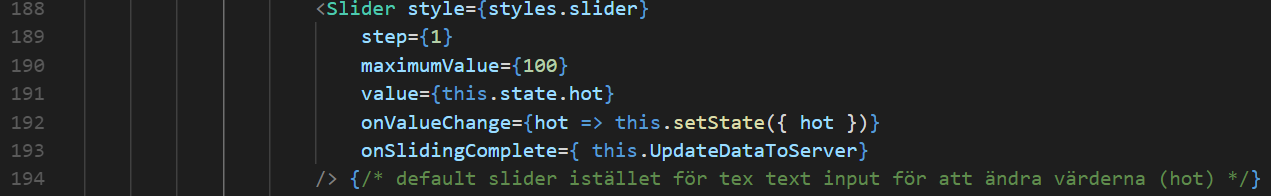


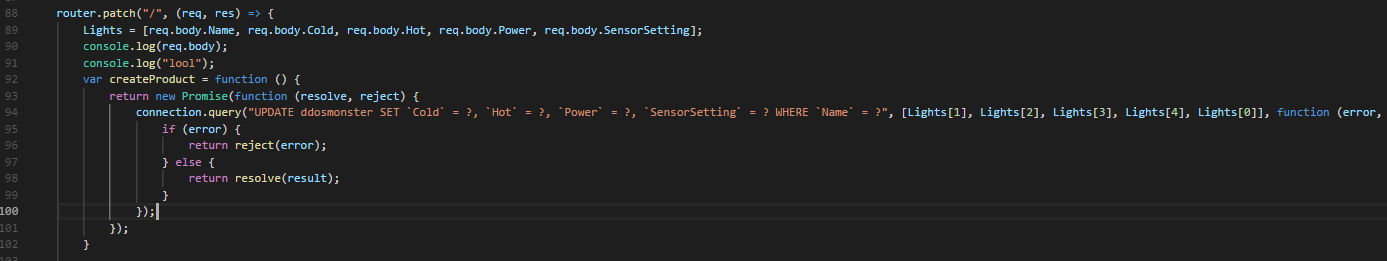
B, Uppdatera värden





C, Slider



Bilaga 2 Backend A. PATCH funktion   
  


Bilaga 3 Wifimanager   
A. Wifimanager   
  
