Ljusidé

Målgrupp för rapporten: Era spetskompisar (En läsare som har samma kunskaper som er men som inte gjort det ni har gjort.

Omfattning: Läsaren ska kunna göra en fungerande kopia av er konstruktion och förstå er lösning.

Grupp 3 – Erik, Edvin och Oliver

Teknisk dokumentation  
HT18

Grupp 3  
Erik Spector, Edvin Sjögren och Oliver Witzel.

Teknik  
Joakim Flink  
Andreas Jillram  
Daniel Åkerlund

Innehåll

[1 Funktionsbeskrivning 3](#_Toc531169302)

[2 Systembeskrivning 4](#_Toc531169303)

[3 Detaljbeskrivning 5](#_Toc531169304)

[3.1 Delsystem 1 (byt namn) 5](#_Toc531169305)

[3.2 Delsystem 2 (byt namn) 5](#_Toc531169306)

[3.3 osv 5](#_Toc531169307)

Bilagor

[Bilaga 1 Kod](#_Toc531168031)

[Bilaga 2](#_Toc531168032)

Figurer

[Figur 1 Figurtext 3](file:///G:\Ämnen\Datorkunskap\Textdokument\Repetition%20Word\Facit\Repetition%20av%20Word%20-%20formaterad.docx#_Toc531167890)

[Figur 2 Figurtext 4](file:///G:\Ämnen\Datorkunskap\Textdokument\Repetition%20Word\Facit\Repetition%20av%20Word%20-%20formaterad.docx#_Toc531167891)

# Funktionsbeskrivning

Vi har konstruerat en lampa som har en mängd olika funktioner. Vår ”huvudfunktion” är att lampan kan befinna sig i två olika lägen, som taklampa och som bordslampa. Det finns fyra plattor där led-stripsen befinner sig, en på varje sida av ”takplattan”. När lampan befinner sig i ”bordslamp-läge”, kommer dessa ”led-stripsplattor” hänga neråt (se bild åt höger). När vi istället lägger lampan i ”taklamp-läge”, kommer plattorna ligga nästan horisontellt jämtemot golvet. Man byter läge genom att dra ”takplattan” upp eller ner. För att fästa lampan i ett av lägena, använder man sig av så kallade ”stopp-pinnar”.

Detta multifunktionella system gör så att man bara behöver en lampa, för att fylla olika funktioner. Att ha denna funktion är utmärkt när man t.ex. vill flytta lampan från en arbetsyta, till en allmän yta.

Med hjälp av fyra hål i ”takplattan”, kan lampan också producera indirekt ljus. Lampan bidrar alltså till det allmänna ljuset på platsen. Självklart producerar lampan också direkt ljus, för att t.ex. lysa upp ett skrivbord.

En annan funktion vi har är att lampan kan styras med hjälp av en app. I appen kan man sätta av och på lampan, ändra ljusstyrkan med hjälp av en ”slider” och välja om man vill ha kallt eller varmt ljus, eller båda! Att stänga av och på lampan är smidigt, det är bara att trycka på ”switch” knappen. Med hjälp av ”slider” kan man ändra styrkan på lampa i procent, 0 – 100. Längst ner finns tre knappar, ”Cold”, ”Warm” och ”Both”. Man kan bara ha en av dessa knappar igång samtidigt. Genom att ändra inställningarna på lampan och sedan trycka på ”Update” knappen, kan man ändra lampans ljus i realtid.

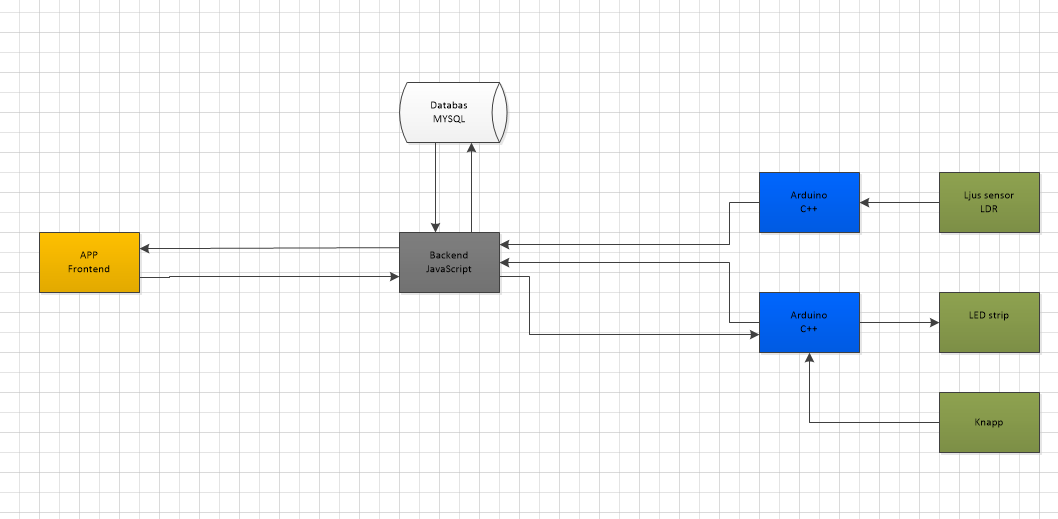
Ytterligare en funktion vi har i appen är ”Power Saving Alert”. Detta betyder att om det är tillräckligt ljust ute, alltså att lampans ljus inte är nödvändigt, kommer appen upplysa användaren. Dock så kommer den inte att skicka dessa meddelanden hela tiden, utan bara skicka när systemet känner att det är nödvändigt. Denna funktion sparar både på elkostnaderna, och miljön.

Vi kan också använda oss av en fysisk knapp för att stänga av och på lampan. Denna sitter uppe på ”takplattan”. Man använder sig av denna när mobil-appen inte är tillgänglig.

# Systembeskrivning

I vårt system finns det fyra ”delsystem”, Backend, Frontend, Databas och Arduino.

Frontend är det som användaren ser, i detta fall appen. Själva estetiken, och en del av logiken, finns i Frontenden. Den använder sig främst av att ändra data i databasen, men också att hämta data för uppvisning. I frontend ligger också logiken för ”Power Saving Alert”, som är en av våra främsta funktioner. Den både hämtar och lagrar data för senare användning.



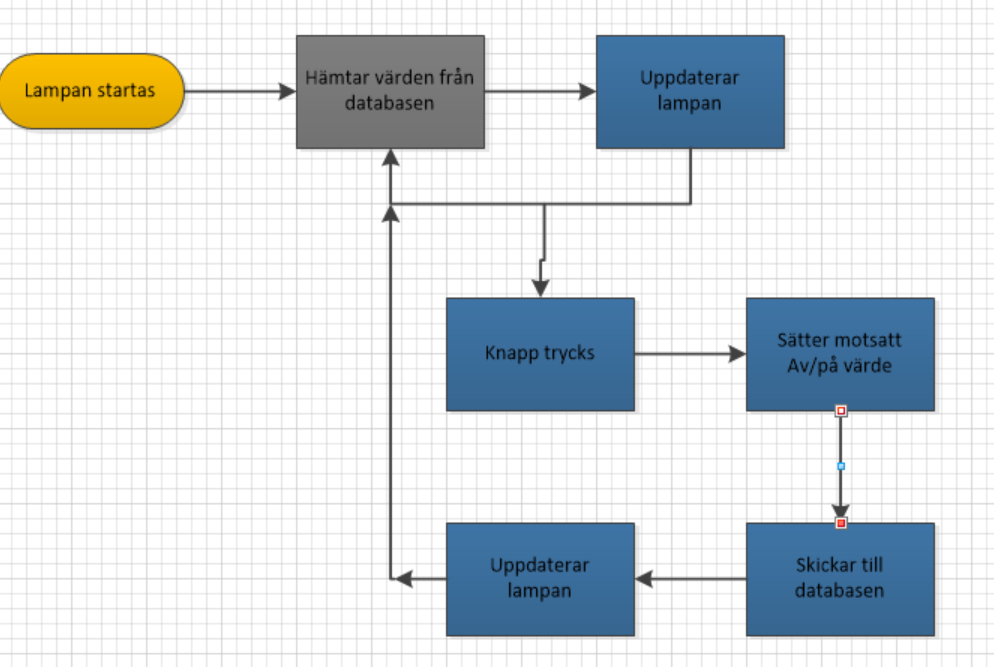
Databasen är en av de viktigaste delarna av systemet. Här lägger de andra systemen upp sin data för lagring. Detta betyder alltså att Frontend kan skaffa data, och lägga upp det på databasen så att Arduino, senare kan hämta Frontends data. Man kan säga att databasen är sättet systemen kommunicerar på. Exempel på data som finns i vår databas är Ljusstyrka, Av / På, LjusUtomhus.

ssssss

Arduino är det system som hanterar lampan och dess inställningar. Arduinon läser av datan som finns i databasen och använder den för att justera ljuset. När t.ex. Frontenden har lagrat sin Av / På data på databasen, kommer Arduinon och läser av det, processerar det, och ändrar om det ska vara av eller på.

Wireframe

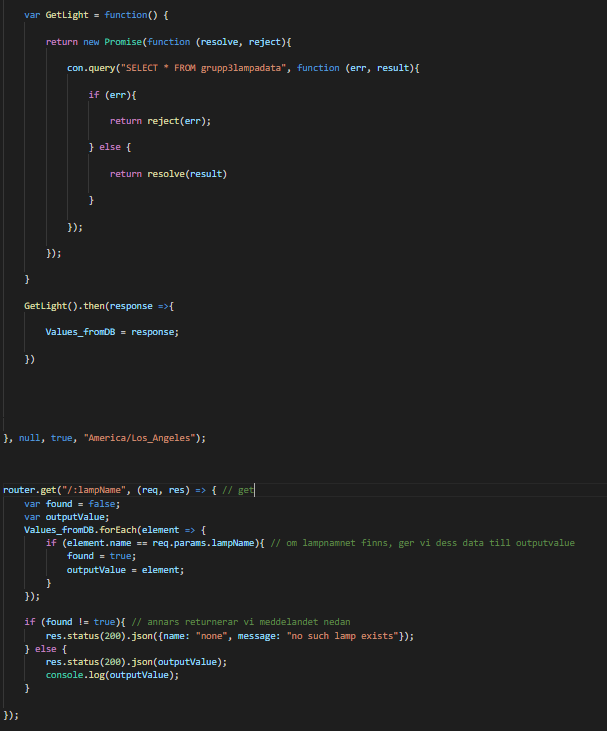
Backenden är själva huvuddelen eftersom alla system går igenom Backend för att kommunicera med varandra. När t.ex. Frontend vill hämta data från Databasen, skickar den en förfrågan till Backend, som översätter det och skickar vidare den till Databasen. Backenden får då ett svar av databasen, översätter svaret, för att sedan skicka det tillbaka till Frontend.



# Detaljbeskrivning

Här en detaljbeskrivning på vår multifunktionella lamparmatur. Den viktigaste och mest komplicerade del i vårat projekt är vår “paraplymekanism”, vår armatur består av 3 olika mekanismer.

Nr 1, Slider med splines

Nr 2, Låsmekanismen

Nr 3, Vingfunktionen

Vår lampa är både fungerande som taklampa och bordslampa, för att skifta läge använder vi splines i våran armatur för en mjuk och enkel rörelse. Vi har ett avstånd på 0,5 mm mellan våran stång och vår topplatta, i topplattan finns en printad del i plast för att förstärka och skapa en slitstark koppling mellan stången och topplattan. Vi har även vår strömtillförsel till lysdioderna i stången med hjälp av ett hål som skär vertikalt genom armaturen.

Våran låsmekanism är både slitstark och enkel att hantera, låsmekanismen fungerar genom att föra in två stoppinnar i gjorda hål i armaturen. För bordsläget finns inga hål, istället finns det en plan yta på toppen av plattan där stoppinnarna låser fast topplattan.

Vingfunktionen på lampan gör att ljusbilden blir större och lampan får en större vidd. Denna funktion fungerar geonm att bottenplattan är större än topplattan och vingarna är delad på två med en 45 graders vinkel. När bottenpalttan skjuts upp trycks vingarna ut och

## Backend

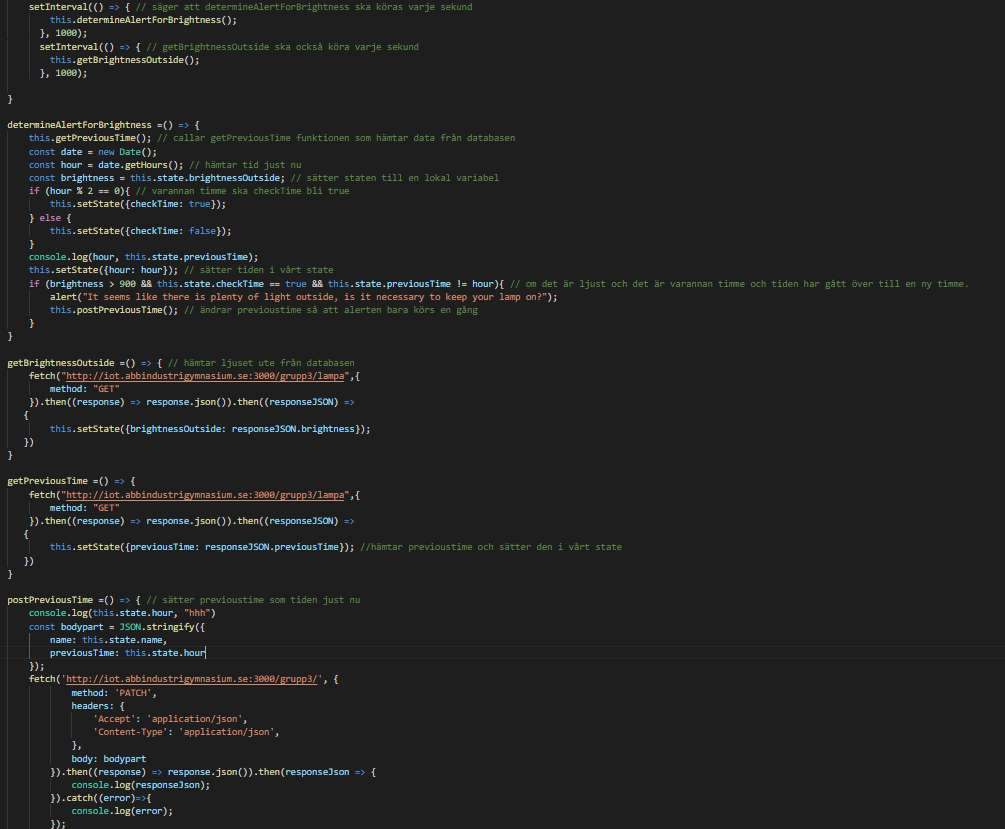
Backenden är en av de viktigaste delarna av hela systemet. Den letar hela tiden efter så kallade ”REQUESTS” som andra enheter skickar till backendet. De ”REQUESTS” som vi använder oss av är ”GET” och ”POST”. När den t.ex. identifierar en ”GET REQUEST” kommer koden nedanför ”router.get” köras. Först körs lite SQL kod för att hämta all data i databasen. Sedan sparas detta i ValuesfromDB och skickas tillbaka till enheten som frågade om datan.

Vi har också en så kallad ”Smart Backend”. Ett problem som uppstådde när vi jobbade med backend var att vi den hela tiden uppdaterade alla värden i databasen, även fast vi inte skickade med just de värdena. Så om vi t.ex. bara ville skicka ”previousTime” till databasen, återställdes alla andra värden i databasen. Vi löste detta genom att skapa en ”queryString”, som ändras beroende på vad enheten skickar in, och sedan använder ”queryString” för att köra SQL kod. Först tittar vi vilka värden som skickades med, och använder dessa för att ändra ”queryString”. Om enheten t.ex. skickar med ”previousTime”, kommer ”queryString” utökas med ”previousTime = ”. Sedan använder vi våran ”queryString” för att köra SQL och ”posta” värdena till databasen.

Nedan har vi dokumentation på våra variabler och requests.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | | Type | | Description | |
| Name | | String | | Namnet på lampan | |
| Temperature | | Int | | Kallt / Varmt | |
| Strength | | Int | | Ljusstyrka på lampan | |
| Onoroff | | Int | | Av / På | |
| Brightness | | Int | | Ljusstyrka ute | |
| PreviousTime | | Int | | Tid vid förra inlogg | |
| Base | URL | Type | Body Parameter | Description | Return |
| /grupp3 | / | GET |  | Hämtar alla värden | String all data, String message |
| /grupp3 | /:LampName | GET |  | Hämtar specifik lampa | String data, String message |
| /grupp3 | / | PATCH |  | Uppdaterar specifik lampa | String messagess |

## Power Saving Alert

Till höger ser vi vår kod till ”Power Saving Alert”. Det första som händer när man startar appen, är att en setInterval börjar köra både ”determineAlertForBrightness” och ”getBrightnessOutside” funktionerna varje sekund.

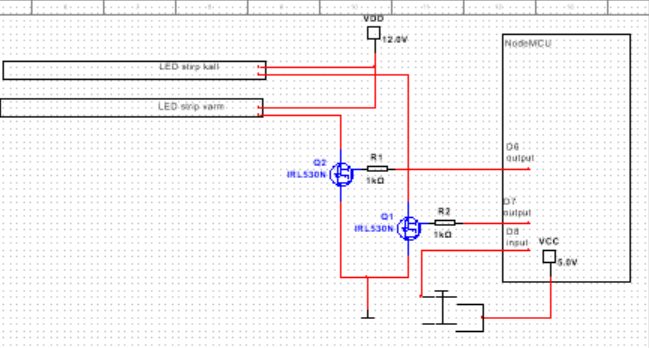
I ”getBrightnessOutside” körs en simpel ”GET” för att hämta ljusstyrkan ute, ett heltal mellan 0 och 1024. 0 är mörkast och 1024 är starkast. Detta heltal sparar vi statet ”brightnessOutside” för senare användning.

I ”determineAlertForBrightness” körs huvuddelen för att bestämma om vi ska meddela användaren, eller inte. Det första som händer är funktionen ”getPreviousTime”. Där hämtar vi det klockslag som det var när vi loggade in förra gången i vår databas. Anledningen till varför vi gör detta kommer att klarna senare i koden. Efter detta så hämtar vi den nuvarande tiden i timmar. Sedan tittar vi om den nuvarande tiden i timmar är delbart med 2. Detta för att vi bara vill att funktionen ska köras varannan timme. Om det är delbart med 2 kommer ”checkTime” bli true, annars förblir den false. Nu har vi all data vi behöver för att bestämma om programmet ska notifiera användaren. Då kör vi en ”if-sats” där vi frågar om ”checkTime” är true, om brightnessOutside är mer än 900 och om ”previousTime” inte är lika med ”currentTime”. Detta betyder att tiden i timmar är delbart med 2, ljusstyrkan ute är starkare än 900 och förra gången vi var inne på appen inte är samma som den nuvarande tiden. Om alla dessa är sanna, kommer vi att notifiera användaren med ett meddelande. Vi kommer också köra funktionen ”postPreviousTime”, där vi sätter ”previousTime” i databasen som den nuvarande tiden. Då kommer notifikationen bara skickas när det är en ny timme, varannan timme.

## Frontend

I frontend har vi olika funktioner. Vi har en ”switch” knapp för Av / På, en ”slider” för att ändra ljusstyrka, och tre knappar för att ändra varmt eller kallt ljust, eller båda. Vi kör en så kallad ”componentDidMount”, som körs direkt när appen startas, där vi hämtar alla värden i databasen med en ”GET”, och lägger dessa värden till våra knappar. Om det t.ex. står att lampan ska vara tänd i databasen, kommer det visas på ”switch” knappen, den blir alltså true.

## Arduino

För att kunna tända och styra lampan så behövs ett mikrokontroller och den styrs av kod. Koden i sig har fyra viktiga funktioner en ljus sensor, syrning av lamporna, på och av knapp samt skicka och hämta värden från databasen. ljussensorn använder sig av en LDR (light dependent resistor) som ökar motståndet när det börjar mörkna, ljust är mellan 3- 5 ohm medans mörkt han ligga mellan 10 – 300 kΩ. (kopplingen av detta kan du se i bilaga z och koden bilaga n). Styrningen av LED stripsen sköts av två stycken IRL530n transistorer (se koppling bilaga y)som styrs av aruino koden(se bilaga x), som i sin tur skickar och hämtar värden via ett bibliotek som heter Wifi-maneger som kopplar upp sig på internetet och hämtar samt skickar värden från databasen som den sedan använder de värderna för att syra styrkan och färgen på lamporna. Av och på knappen är designad för att kunna fungera separat med resten av lampan och det gör den med en så kallad interrupt funktion som pausar programmet då ett tilstånd uppfylles och kör väldigt snabbt ett par få rader kod då man inte vill pausa systemet under en längre tid, denna funktion behövs då mikrokontrollen inte kan göra två saker samtidigt eftersom den består av en kärna.(se bilaga x) Knapp trycket uppdaterar både lampan och databasen för att inte skapa ett blinkande då lampan uppdateras.

Bilaga 1 Kod (lämplig rubrik)

(Fullständig och kommenterad)

Bilaga 2 (Lämplig rubrik)