Funktionsbeskrivning

Vi valde att göra en dimbar lampa där man kunde justera färgtemperaturen.

Ljuset går att justeras på två olika sätt, med en applikation och med en fjärrkontroll. I applikationen kan man justera både färgtemperaturen och ljusstyrkan medans man bara kan justera ljusstyrkan med fjärrkontrollen. Lampans designad att byta ut en 60\*60 cm takplatta som finns i många kontor och skolor för för lätt upsättning.

Systembeskrivning

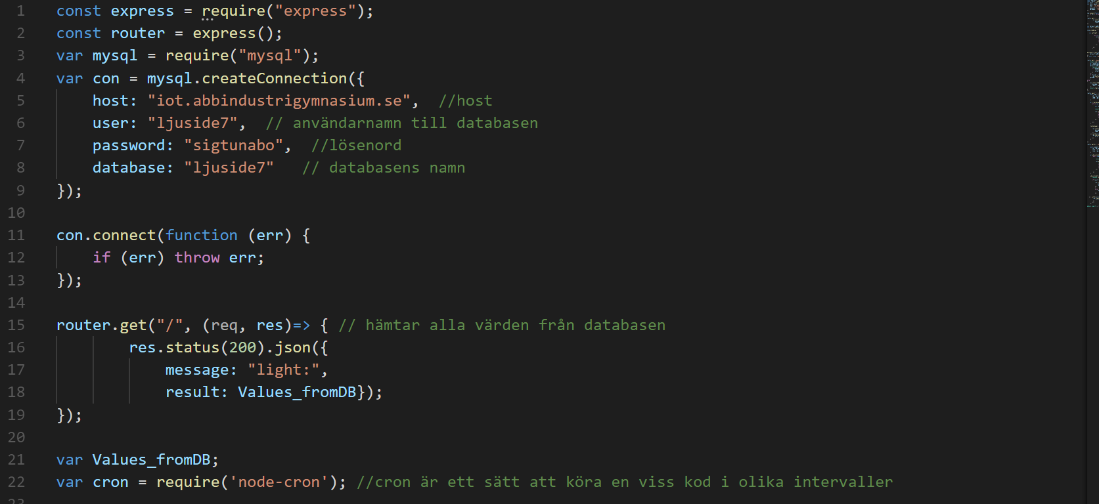
A close up of a white wall

Description generated with high confidence

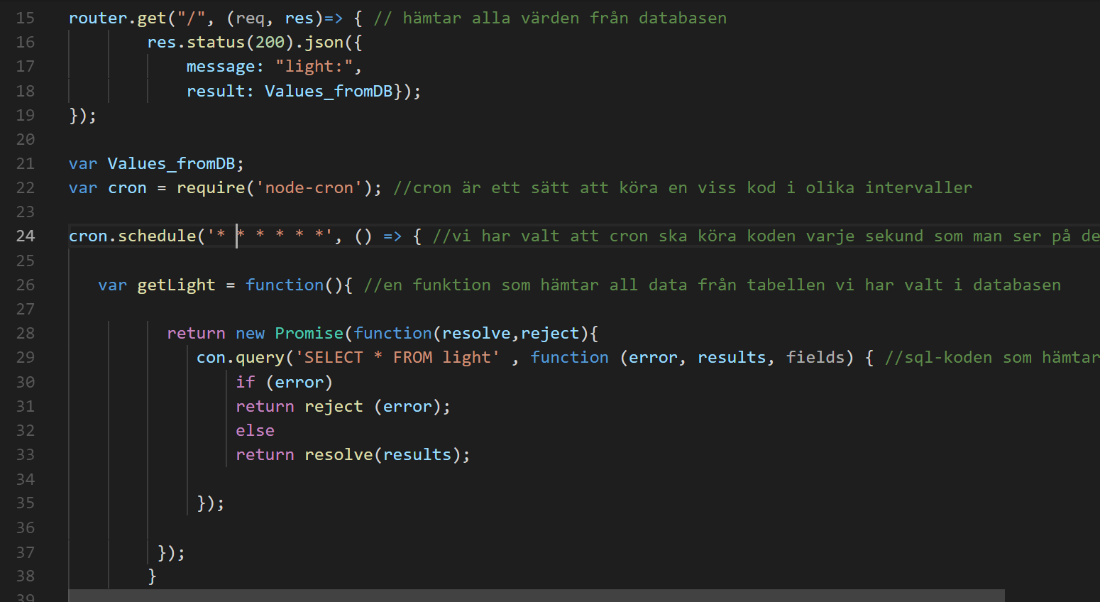
Våran lampa styrs av ett Arduino-kort som kontrollerar ljusstyrkan och färgtemperaturen. Ledstripsen är ihop lödda med ledstrips av samma färg/temperatur det vill säga att kalla dioderna är ihop lödda med de andra kalla dioderna och de varma dioderna är ihop lödda med de andra varma dioderna. Detta är för att man ska kunna kontrollera alla de olika dioderna för samma färgtemperaturer samtidigt. De är sedan ihopkopplade till olika portar på Arduino-kortet. Fjärrkontrollens kod är skriven i språket C++ är ihopkopplad med databasen och styr ljusstyrkan i 20% steg genom att hämta och uppdatera värdena i databasen. Vi har också konstruerat fjärrkontrollen så att den går att använda som både bärbar och stationär. Backenden är en mellanhand mellan databasen och frontenden/arduino-korten och uppdaterar värdena en gång i sekunden.

Detaljbeskrivning

**Backend**



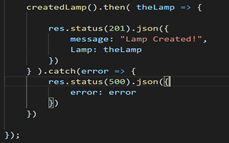
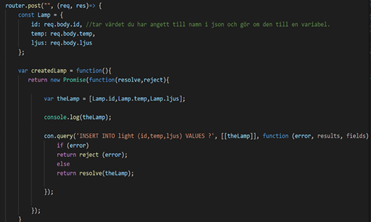
Om man börjar att kolla på början i backenden så ser man att vi först lägger in vilken användare, lösenord, databas och host som vi ska använda oss av.



Sen om man går vidare så ser man att det kommer en get funtion där man hämtar alla värden från databasen. Sen gör man en variabel med Values\_fromDB. Vi la också in node-cron som är ett sätt att köra sin kod i en bestämd intervall. Sen gör vi getLight till en function som ska hämta alla värden från databasen och den säger också att antingen ge ett error eller ge resultaten. Sen på slutet så kör man functionen getLight som sen lägger in värdena i en lokal databas (Values\_fromDB). Man säger också vilken tidzon man ska gå efter.



Sen har vi en get där vi säger att man ska hämta alla värden från ”Name”. ”Name” i det här fallet är lampans id och detta är väldigt bra eftersom man kan söka efter värdena hos en specifik lampa och om ingen lampa med det id hittas så får man ett error där det står ”no lamp found”,



Sen kommer våran post där vi har en const Lampa som då vill att man ska bestämma id, temp och ljus. Efter det skapar vi variabeln createdLamp som frågar om vad id, temp och ljus ska vara och kommer den sätta in värdena i light där tabbelen ligger.

Sen kör vi variabeln createLamp som också är en fuction. Då kommer den antingen ge oss statusen 201 och meddelande ”lamp created” eller så får vi error 500.



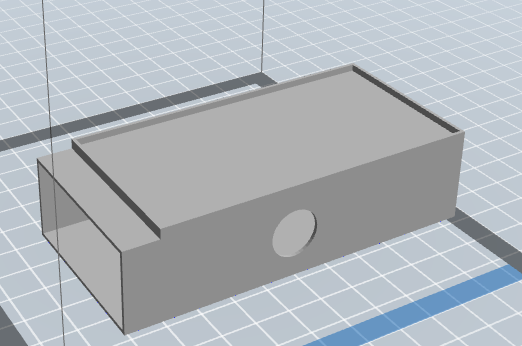
Nu kommer våran patch som specielt frågar efter vilket id lampan har. Vi gör också en variabel som heteradjustLight som också är en function. I den functionen så frågar vi efter vilka värden “temp” och “ljus” ska ha och sedan vilket id lampan vi ska ändra har.



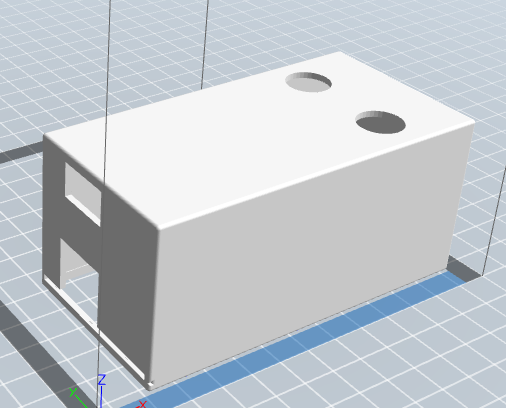
Sen kör vi variabeln adjustLight om det då finns mer än 0 påverkade rader I våran databas tabell så kommer den att säga “lamp adjusted” eller om inga rader påverkas “nothing changed”.

Fjärrkontroll Solidworks

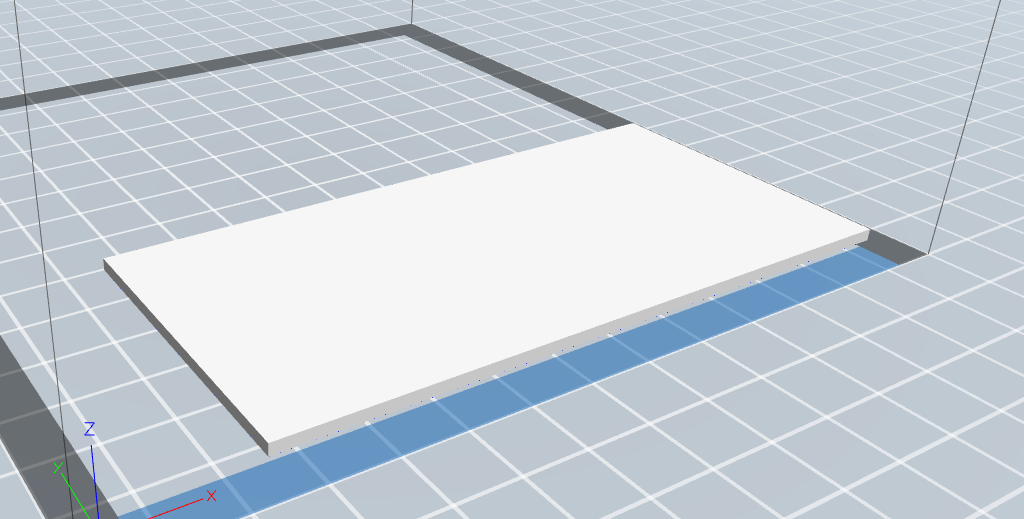
När vi först kom på att göra en fjärrkontroll så var det viktigt att den skulle vara trådlös eftersom annars kunde man lika gärna haft en kanpp på vägen. Så det första vi tänkte var en designe där man lätt mikrocontrollen sitta fast i en rätblock där man också skulle ha ett batteri fast sittande och sen ha ett skjut lock som sattes på från ovansidan. Tyvärr så insång jag att det var några stora problem med den designen. Blandannat så skulle det bli väldigt svårt att komma åt knapparna som man ändra ljusstyrkan med eftersom några sladdar var högre än vad kanpparn satt. Det andra problemet var att ha ett batteri som står och går hela tiden. Lösnigen på dom här två problem var att först och främst göra en designe där knapparna var mer lättåtkompliga och det gjorde vi genom att ta låss en del av sladdarna så att man kunde böja dom bättre och låta knapparna komma upp högre på det sättet. Det andra problemet med batteriet löste vi genom att använda oss av en powerbank och därför fick jag cadda till en hållare för powerbanken under minckrokontrollen.



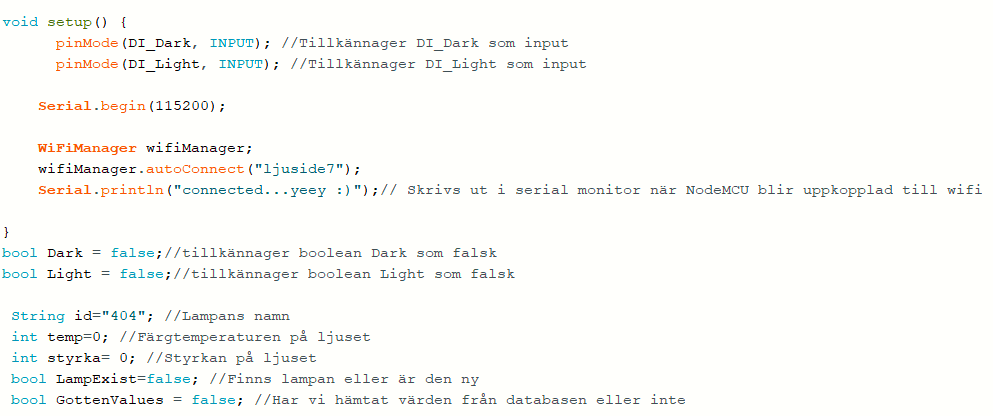
Sen kommer vi till den andra delen av kontrollen och det är chasit eftersom man kan ju inte bara ha en mirckrokontroller och massa sladdar som satt helt öppet. Ett problem som uppstog när jag skulle göra chasit var att 3d skrivaren klarade inte av att skriva ett tillräkligt långt chasi för att usb sladden skulle få plats innuti chasist. Det fick jag helt enkelt lösa genom att cadda ut 2 hål där sladden kunde gå från mickrokontrollen till powerbanken. Sen kaddade jag ut ett hål längst ner för att kunna göra plats för locket som skulle gå att skjuta in så att inte hållaren ramlade ur.



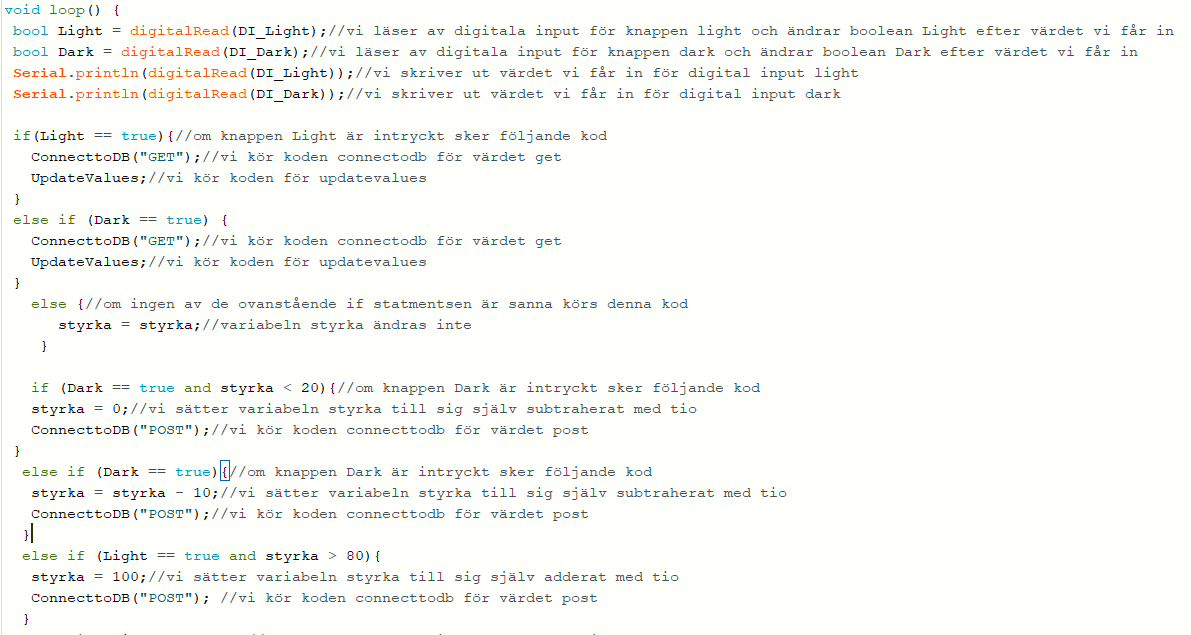
Sen när även det var utskrivet så var det bara att cadda locket som skulle hålla upp allt. Det var väldigt lätt eftersom man behövde bara använda samma mått som man använde när man cadda hållet för den.

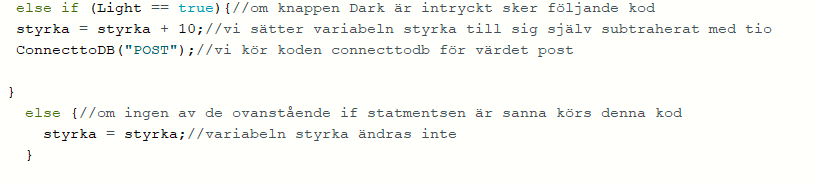


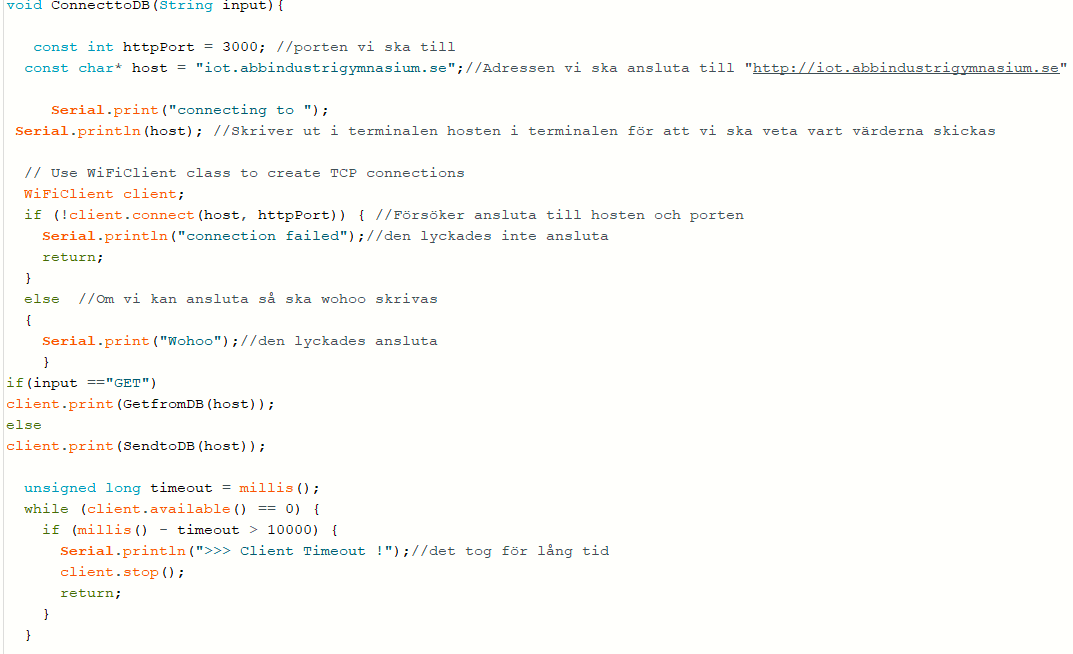
**Fjärrkontroll C++**

Koden till fjärrkontrollen kan hämta uppdatera och lägga till nya värden i databasen. Den börjar med att inkludera ett antal bibliotek, definiera globala variabler och definiera vilka in och utgångar/ingångar som används på NodeMCU kortet. Den kopplar sedan upp sig till wifi för att senare kunna få kontakt med databasen.

Efter den kopplat upp till internet läster den av sensorerna tills den får in ett värde över noll. När den får in ett värde över noll kopplar den upp databasen och hämtar värdena som finns under id 404. Den sparar sedan de hämtade värdena i de globala variabler som skapades i början. Koden går efter det igenom en if sats där den globala variabeln styrka uppdateras beroende på vad det hämtade värdet var och beroende på vilken av sensorerna som gav utslag. Den kopplar sedan upp sig till databasen igen och uppdaterar värdena under id 404.







**Front-end**

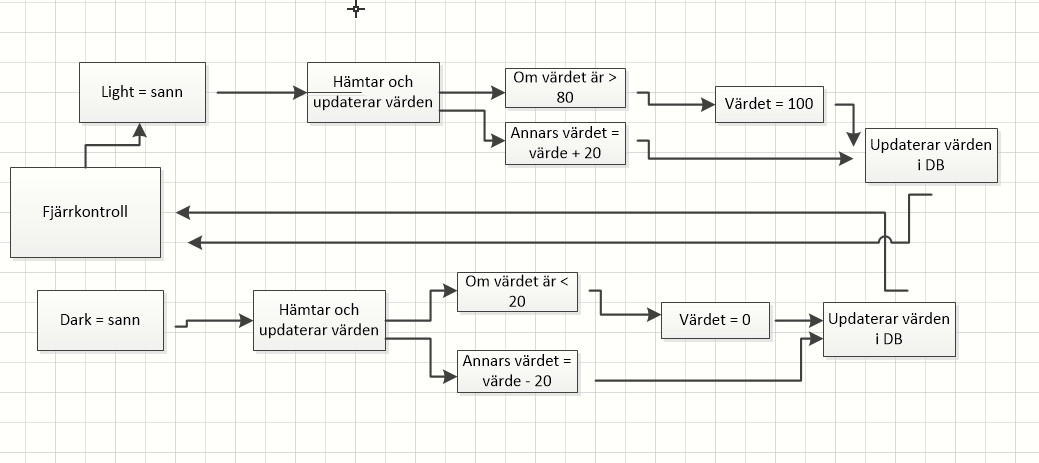
Vår front-end är en Android applikation som man kan använda för att dimma lampan och ändra färgtemperaturen på lampan. Applikationen är skriven i JavaScript med hjälp av verktyget react native. Den består av två sliders; en som bestämmer ljusstyrkan i procent och en som bestämmer färgtemperaturen i procent där 100 är varmt och 0 är kallt. Högts upp i applikationen ser vi ABB industrigymnasiums logga och en lampa. Den fungerar genom att vid start hämta de värden som finns på databasen för Ljusstyrka och Färgtemperatur för att sedan göra om dem till variabler i koden. När det är gjort så ändras sliderna till de värden som hämtats från databasen. I koden har vi sedan sett till att värdena bara ändras när man släpper slidern. Detta för att det inte ska skickas onödigt många värden till databasen. Ovanför slidesen ser vi procenten som slidesen står på. Applikationen förhandsgranskades och byggdes om till en android-applikation med hjälp av verktyget expo.

**Lampans mikrokontroller**

På Arduino-kortet som styr lampan körs en kod som hämtar värden från databasen och justerar lampan efter det, så ofta som möjligt. Koden är skriven i C++ och. Datan som hämtas från databasen bestämmer sedan hur lampan ska lysa. För att detta ska funka så ansluter först kortet till WiFi med hjälp av WiFiManager som är ett smartare sätt att ansluta kortet till WiFi. Vi har sedan även skapat en funktion som hämtar data från backenden och gör om det till lokala variabler. Dessa används sedan i en matematisk funktion för att bestämma hur starkt de varma respektive de kalla led-stripsen ska lysa. I bilden under ser vi hur funktionen för lamp-justeringen ser ut. Hur mycket de varma led-stipsen lyser bestäms alltså av ljusstyrkan och värdet ”varm”, som är en slags förändringsfaktor som säger hur mycket de varma ska lysa, detta multipliceras sedan med 10,24 för att outputen ska vara ett värde mellan 0 och 1024. Samma sak händer med de kalla led-stripsen men med det omvända värdet av ”varm”. Innan den här funktionen körs så hämtar vi nya värden från databasen för att lampan alltid ska vara det värdet som står i databasen. analogWrite används för att ändra led-stripsen till de nya värdena.

A screenshot of a cell phone

Description generated with very high confidence



Våran tillståndsgraf visar hur våran fjärrkontroll fungerar. Om vi börjar med att kolla på det övre ledet så ser vi att och arduino-kortet läser av om knappen Light lyser eller Light=sann så kommer den att först hämta data från databasen, sen kan två olika saker hända. Antingen så får den tillbaka ett värde från databasen som är större än 80 och då sätts värdet till 100 och arduino skickar tillbaka värdet till databasen. Om man istället hämtar värdet och värdet är mindre än 80 så triggas istället att värdet ökar med +20 för varje knapp tryck man gör och sen skickas det nya värdet tillbaka till databasen.

Om knappen Dark stället är intryckt eller dark=sann så hämtar den värden från databasen. Om värdet är mindre än 20 så blir det nya värdet 0, annars minskar värdet med 20 för varje knapptryck. Det nya värdet skickas sedan tillbaka till databasen. Denna process återupprepas så länge arduino kortet har ström och är uppkopplat mot nätverket.

**Armatur**

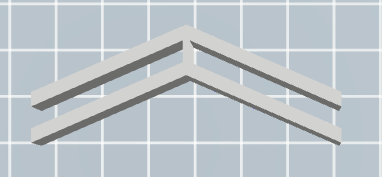
Armaturen består av plexiglas, cellpannå och PLA plast. Plexiglaset är utskuret i 4st 2\*55,4 cm bitar och en 55,4\*55,4 cm bitar. Plexiglaset är opalt och släpper igenom 78% av ljuset som går igenom. Vi valde att använda detta glas för att det släpper igenom mycket ljus men ljuset som släpps igenom blir ej bländande. Cellpannån är utskuren i 8st 29\*16,2 cm bitar där en triangel med måtten 15,2\*16,2\*16,2 cm utskuren ur sin ena kant se bild nr1. Två av de 8 cellpannåbitarna sätts sedan ihop så de bildar en vinklad sida i den kvadratiska armaturen se bild nr1. De 3D printade bitarna i PLA plast är 4 stora hörn, 10 mindre hörn och 12 raka urholkade bitar för att hålla ihop cellpannåbitarna i bild nr1. Det finns sedan en 30.38\*30.38cm kvadratisk cellpannå platta som ledstripsen sitter på. Lampans olika delar byggs ihop till en icke spetsig pyramid som ger möjlighet till mycket direkt ljus men även spridning av ljus över taket för att rummet inte ska se litet ut.





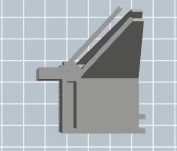
Bit för att hålla ihop cellpannå

Bitarna ovan håller ihop två av de 8 cellpannåbitarna tillsammans. Det sitter tre stycken mellan varje par av cellpannå.

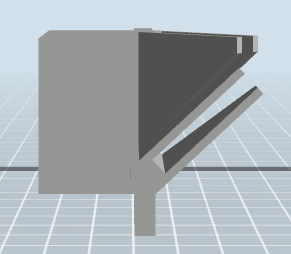


Mindre hörn

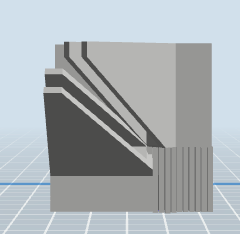
De mindre hörnen håller ihop de vinklade cellpannåbitarna med 30,38\*30,38 cm cellpannåplattan där ledstripsen sitter. Det sitter två stycken på två av de fyra kanterna och tre stycken på de andra två kanterna.



Hörn stora



Hörn stora

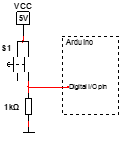


Hörn stora

De stora hörnen håller vardera ihop 2st 2\*55,4 cm plexiglas bita, ett av de fyra hörnen på 55,4\*55,4 cm plexiglasplattan och två av de fyra vinklade cellpannå bitarna. De hänger även alla komponenter på metallrelingen som går i taket med två rektangulära utstickande bitar.

**Koppling**

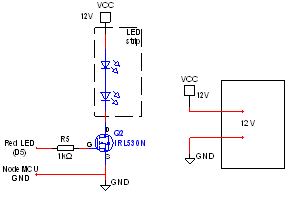
Fjärrkontrollen är kopplad med två knappar som kan få utfallet 0 och 1. Varje knapp är kopplad till jord, VCC och en digitalinput pinne på arduino-kortet. Kopplingsschema 1 är för att koppla in en av de två knapparna, knapp nummer två går att koppla in på samma sätt som knapp 1 det vill säga följ kopplingsschema 1 2ggr.



Kopplingsschema 1

Ledstripsen är 27,5cm långa och 8st av varje färgtemperatur. De är fastsatta med 1.6cm avstånd från varandra på armaturen.

Led stripsen är kopplade till VCC och jord. De är även kopplade till en transistor för att kunna driva flera ampere men inte överstiga 12 millieampere. Se kopplingsschema 2



Kopplingsschema 2

Ledstripsen är 27,5cm långa och 8st av varje färgtemperatur. De är fastsatta med 1.6cm avstånd från varandra på armaturen.