### Design, implementering og test af unitHandler klassen (SN)

Denne klasse er designet til at håndtere enheder og rum, mens der samtidigt bliver holdt en struktur på SD-kortet som gør det nemt at hente ønskede oplysninger om en enhed.  
For at opnå dette var det nødvendigt at fastsætte nogle retningslinjer for allokeringen af blokke på SD-kortet, inden funktionerne blev lavet.

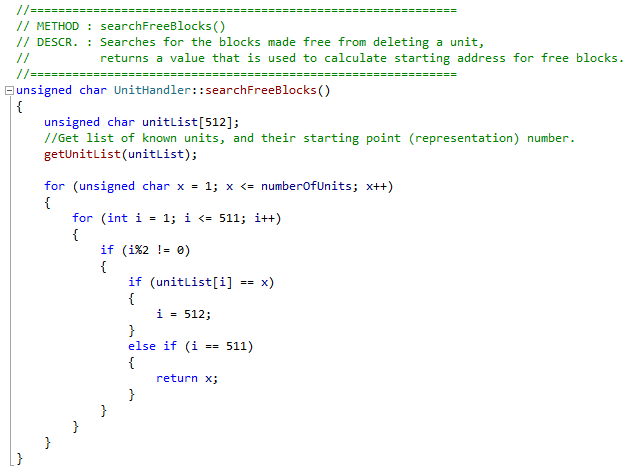
Der blev fastlagt følgende retningslinjer for allokeringen af datablokke på SD-kortet:

1. De første 2 blokke på SD-kortet (blok 0 og 1) bliver brugt til henholdsvis enhedsliste, og rumliste.
   1. På enhedslisten gemmes på første blok et enhedsnavn, med en efterfølgende counter-værdi. (bruges til beregning af startblok)
   2. På rumlisten gemmes, på første blok, et rum navn, med en efterfølgende counter for det rum.
2. For hver enhed der er tilføjet systemet allokeres 7 blokke, som hver skal indeholde tidsplanen for enheden på en given dag i ugen
3. Der bliver fastsat en adresse (start blok) for hvor fejlloggen starter. (Kommer i senere udvidelser)

Vi benytter således en counter, til at holde styr på hvor mange enheder der på et givent tidspunkt er oprettet i systemet, og beregner så på hvilken placering, den første blok skal ligge for den enhed vi er i gang med at tilføje.  
Oplysninger om enhedens ID, antal enheder oprettet, rum og dag gemmes på de første 4 bytes af hver blok, der oprettes for alle enheder. Det vælges derfor at lave counteren som en unsigned char, i stedet for at lave den som en integer da SD-kortets struktur gør det nemmere at arbejde med en enkelt byte.

Der var flere ting som viste sig særligt udfordrende i udviklingen af unitHandler klassen. Én ting, netop pga. fremgangsmåden med at beregne en startblok ud fra en counter-værdi, var at håndterer sletningen af enheder, og følgevirkninger heraf.  
Hvis en enhed slettes, og en ny oprettes, vil to enheder altså få den samme counter værdi.  
For at komme uden om dette problem, blev det besluttet at når en enhed slettes skulle blokkene som repræsenterede enheden overskrives med 0x00 på alle bytes. Der skulle så videre laves en funktion som kunne tjekke om standard tildelingen var ledig, og en anden som kunne søge efter de tomme blokke på SD-kortet (se Figur 2).

Koden der tjekker standard tildelingen kunne let implementeres pga. den valgte struktur på enhedslisten. Mens koden til at søge efter ledige pladser på SD-kortet kom til at give lidt flere problemer, selv om det endte med at være en simpel implementering (se Figur 2).



Figur - Kode der søger efter tommeblokke på SD-kortet.

Det blev således muligt at håndtere følgevirkningerne af slettede enheder, ved at tjekke standard tildelingen indledningsvis, og hvis denne returnerer false, søge efter de ledige pladser og bruge denne værdi på den tilegnede byte, som det er vist på

En anden overvejelse som opstod som følgevirkning af slettede enheder var håndteringen af enhedslisten. Når en enheds blev slettet midt i listen, var det enten nødvendigt at kunne søge efter den ledige plads, eller rykke resten af listen for at udfylde pladsen. Det blev her besluttet at komprimere listen, da det ellers ville være nødvendigt at gennemsøge enhedslisten ved hvert kald af addUnit-funktionen for at vide om der var tomme pladser i listen, og da vi kun kan skrive en hel blok til SD-kortet af gangen, vil der ikke være noget at vinde ved kun at ændre to bytes i blokken.

Det endelige klasse diagram for unitHandler-klassen ses på Figur 5.



Figur - klassediagram for unitHandler.

Alle funktioner i klassen testes med et til formålet skrevet test-program. Test programmet gør desuden brug af UART-driveren, til at outputte testdata på PC via. Tera-term (<https://ttssh2.osdn.jp/index.html.en>).

Test programmet er lavet ud fra bottom-up princippet, og gør brug af en test opstilling med ATmega2560 tilkoblet SD-modul, og forbundet til PC via. den indbyggede usb forbindelse.

Se bilag ”Testprogram for unitHandler” for yderligere information.