UnitHandler klasse

**Ansvar:** Skal håndterer (oprette, redigerer og slette) alle enheder i systemet.

Denne klasse implementerer en række funktioner som gør det muligt at systemet gemmer oplysninger om enheder, rum og tidsplaner på det SD-kort, som er en del af vores styreboks.   
Funktionerne i denne klasse sørger også for at allokeringen af datablokke på SD-kortet sker på en struktureret måde, som gør det muligt nemt at hente ønskede data fra SD-kortet.

### Allokering af data-blokke på SD-kort.

Allokeringen af blokke på SD-kortet sker ud fra følgende regler:

1: de første 2 blokke på SD-kortet (blok 0 og 1) bliver brugt til henholdsvis enheds-liste, og rum-liste.

2: Hver enhed tilføjet til systemet får allokeret 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)



# Funktioner:

|  |  |
| --- | --- |
| Funktion: | UnitHandler(sdCard \*SDObj) |
| Parametre: | Pointer til et sdCard objekt. |
| Returværdi: | Ingen. |
| Beskrivelse: | Constructor for klassen, skal bruge en pointer til et sd-Objekt ved oprettelsen. |

|  |  |
| --- | --- |
| Funktion: | bool AddUnit(unsigned char unitID, unsigned char roomID 0x00) |
| Parametre: | unitID som er den unikke identifier for den enhed der forsøges oprettet. roomID, som er muligheden for at gøre enheden til en del af en gruppe (rum), standart-værdi som er intet rum = 0x00 |
| Returværdi: | Boolsk. |
| Beskrivelse: | AddUnit-funktionen kalder først funktionen checkStdAssignment(); for at se om vi kan anvende vores standart tildeling af blokke på sd-kortet.  Hvis checkStdAssignment(); returnerer false, vil funktionen kalde endnu en funktion: searchFreeBlocks(); som søger efter frie blokke på sd-kortet, som kan anvendes til oprettelsen. Når start-blokken er lokaliseret, allokeres 7 blokke på sd-kortet til den oprettede enhed, og der gemmes på hver blok tidsplanen for den givne dag i ugen. |

|  |  |
| --- | --- |
| Funktion: | bool RemoveUnit(unsigned char unitID) |
| Parametre: | unitID som er den unikke identifier for den enhed der skal slettes. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen henter først unit-listen, og gennemsøger denne efter et matchende unitID, og henter her fra den værdi som benyttes til at beregne start blokken for den enhed. Funktionen tjekker så om enheden er tildelt et rum, eller standart værdien 0x00, hvis enheden er del af en rum, vil funktionen først trække et fra antallet af enheder i det rum, og vil så herefter over skrive de 7 blokke, som før repræsenterede enheden med 0x00, og kalder til sidst funktionen removeUnitFromList(), for at alle data på SD-kortet som repræsenterede den enhed er slettet. |

|  |  |
| --- | --- |
| Funktion: | unsigned char getUnitCount() |
| Parametre: | Ingen. |
| Returværdi: | unsigned char …. Retunerer numberOfUnits. |
| Beskrivelse: | Retunerer varriablen numberOfUnits, som i klassen bruges til at holde styr på antallet af gemte enheder. |

|  |  |
| --- | --- |
| Funktion: | void getUnitList(unsigned char Array[]) |
| Parametre: | Unsigned char Array[] |
| Returværdi: | Ingen. |
| Beskrivelse: | Fylder det array man giver funktionen som parameter med enhedslisten fra blok 0 på SD-kortet. |

|  |  |
| --- | --- |
| Funktion: | bool addUnitToList(unsigned char unitID, unsigned char address) |
| Parametre: | unitID som er den unikke identifier for den enhed der skal tilføjes listen.  Address som er den værdi vi bruger til at beregne start-blokken for enheden på sd-kortet. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen henter først enhedslisten, og gennemsøger denne efter første ledige plads. Hvis ikke der er flere ledige pladser på listen, vil funktionen returnerer false, ellers vil den skrive unitID på første ledige plads, og address på pladsen lige efter, og returnerer true hvis det lykkedes at skrive til sd-kortet, eller false hvis det fejler. |

|  |  |
| --- | --- |
| Funktion: | bool removeUnitFromList(unsigned char unitID) |
| Parametre: | unitID som er den unikke identifier for den enhed der skal fjernes fra listen. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen kalder først getUnitList(), og gennemsøger så dette array for placeringen af den specificerede enhed. Når placeringen er fundet, oprettes et nyt array, hvor alle pladser indtil placeringen af enheden kopieres lige over fra det første array, og alle pladser efter den specificerede enhed shiftes 2 pladser. Til sidst skrives det nye array til plads 0 på SD-kortet. |

|  |  |
| --- | --- |
| Funktion: | void getRoomList(unsigned char Array[]) |
| Parametre: | Unsigned char Array[] |
| Returværdi: | Ingen. |
| Beskrivelse: | Fylder det array man giver funktionen med rum-listen fra blok 1 på SD-kortet. |

|  |  |
| --- | --- |
| Funktion: | bool createNewRoom(unsigned char RoomID) |
| Parametre: | Unsigned char RoomID som er identifier for det rum der skal oprettes. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen kalder først getRoomList(), og gennemsøger så det fyldte array for tomme pladser. Hvis der ikke er flere tomme pladser returneres false, ellers skrives roomID på første ledige plads, og det opdaterede array bliver skrevet til blok 1 på SD-kortet. |

|  |  |
| --- | --- |
| Funktion: | bool deleteRoom(unsigned char RoomID) |
| Parametre: | Unsigned char RoomID, som er unik identifier for det rum som ønskes slettet. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen starter med at kalde getRoomList(), og gennemsøger så det udfyldte array for det rum der ønskes slettet, og hiver informationer ud fra arrayet om hvor mange enheder der på det givne tidspunkt er tilføjet rummet, og gemmer den info lokalt (i samme scope). Når placeringen er fundet, oprettes et nyt array, hvor alle pladser indtil placeringen af det specificerede rum kopieres lige over fra det første array, og alle pladser efter det specificerede rum shiftes 2 pladser, hvorefter det nye array skrives til plads 1 på SD-kortet. Til sidst gennemsøges startblokke i SD-kortet (efter lister, hver syvende blok) efter enheder som er tildelt til det rum som ønskes slettet.  Når en enhed som er en del af det rum bliver fundet, kaldes funktionen removeUnitFromRoom(), og disse enheder fjernes fra rummet. Alt dette sker i et loop, indtil der er fjernet lige så mange enheder som ved funktionskaldet var en del af det rum vi ønskede at slette. |

|  |  |
| --- | --- |
| Funktion: | bool addUnitToRoom(unsigned char roomID) |
| Parametre: | Unsigned char roomID, som er unik identifier for det rum hvor der ønsked tilføjet en enhed. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen kalder først getRoomList(), og gennemsøger så det fyldte array for pladsen for det specificerede rum. Når placeringen er fundet lægges 1 til antallet for dette rum, og det opdaterede array skrives til sidst til SD-kortet. |

|  |  |
| --- | --- |
| Funktion: | bool removeUnitFromRoom(unsigned char unitID, unsigned char roomID) |
| Parametre: | unitID som er den unikke identifier for den enhed der skal fjernes fra rummet specificeret. roomID som er den unikke identifier for det rum der skal opdateres. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Først kaldes getRoomList(), og det fyldte array gennemsøges efter pladsen for det specificerede rum. Hvis rummet findes trækkes én fra antallet tilhørende dette rum, og den opdaterede liste skrives til SD-kortet. Hvis ikke rummet findes i listen, går funktionen blot videre til næste punkt. Herefter kaldes getUnitList(), og det fyldte array gennemsøges efter den specificerede enhed. Når enheden er fundet i enhedslisten beregnes start-blokken for enheden, og alle 7 blokke der repræsenterer enheden hentes ned og rettes til rum 0x00, og skrives igen til SD-kortet. |

|  |  |
| --- | --- |
| Funktion: | bool UpdateTime(unsigned char unitID, unsigned char schedule[]) |
| Parametre: | Unsigned char unitID, som er den unikke identifier for enheden der skal opdateres. unsigned char schedule[], som er det array med hele tidsplanen for en dag der modtages fra PC. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen kalder først getUnitList(), og gennemsøger det fyldte array for placeringen af den specificerede enhed, og gemmer lokalt værdien der bruges til at beregne start-blokken for enheden. Funktionen tjekker så det array som er modtaget som parameter, for hvilken dag denne repræsenterer for enheden, og benytter så disse oplysninger til at beregne hvilken blok der skal opdateres.  funktionen opdaterer så det modtagne array (schedule) med start blok repræsentationen, og skriver til sidst den nye opdaterede tidsplan til SD-kortet. |

|  |  |
| --- | --- |
| Funktion: | bool getTimeTable(unsigned char day, unsigned char UnitID, unsigned char schedule[]) |
| Parametre: | Unsigned char day, fortæller hvilken dag vi ønsker at få tidsplanen for. Unsigned char UnitID, som er den unikke identifier for den enhed som ønskes en tidsplan for. Unsigned char schedule[], som er det array der skal fyldes med tidsplanen for den givne dag. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen kalder først getUnitList(), og gennemsøger denne efter den specificerede enhed. Når denne er fundet, bruges parameteren ”day” sammen med denne værdi til at beregne hvilken blok vi skal hente fra SD-kortet. Til sidst læses den beregnede blok ind i schedule-arrayet som funktionen modtog som parameter. |

|  |  |
| --- | --- |
| Funktion: | bool editUnit(unsigned char previusUnitID, unsigned char newUnitID, unsigned char roomID) |
| Parametre: | Unsigned char previousUnitID, som er det unitID som ved funktionskaldet er gemt i styreboksen. Unsigned char newUnitID, som er det nye unitID vi ønsker gemt. Unsigned char roomID, som er det nye roomID vi ønsker enheden tilføjet (standart er 0x00). |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen kalder først getUnitList(), og gennemsøger det fyldte array efter ”previousUnitID”, og dennes start-blok position, som gemmes i en lokal variable.  Denne enheds unikke identifier opdateres så til ”newUnitID” og den opdaterede liste skrives igen til SD-kortet. Nu læses den første blok for enheden fra SD-kortet, for at finde ud af hvilket rum denne enhed i forvejen er tilføjet, og dette gemmes i en lokal varriable. Så kaldes getRoomList(), og det fyldte array gennemsøges efter et roomID som matcher det roomID som enheden i forvejen havde, og her trækkes én fra antallet af enheder i det rum.  Funktionen gennemsøger nu rumlisten for at se om det rum som enheden skal tilføjes i forvejen eksisterer. Hvis rummet i forvejen eksisterer lægges en til antallet af enheder i rummet, og den opdaterede liste skrives til SD-kortet. Hvis ikke rummet i forvejen eksisterede, vil funktionen oprette rummet med createNewRoom() og herefter tilføje en enhed til det rum med addUnitToRoom(). Til sidst er et loop, hvor de 7 blokke som repræsenterer enheden bliver hentet fra SD-kortet en ad gangen, får opdateret unitID og roomID og igen bliver skrevet til SD-kortet. |

|  |  |
| --- | --- |
| Funktion: | unsigned char searchFreeBlocks() |
| Parametre: | Ingen. |
| Returværdi: | Unsigned char. |
| Beskrivelse: | Funktionen kalder først getUnitList(), og gennemsøger så det fyldte array efter den ”numberOfUnits” som er blevet slettet. Når en værdi er fundet vil den blive returneret. |

|  |  |
| --- | --- |
| Funktion: | bool checkStdAssignment() |
| Parametre: | Ingen. |
| Returværdi: | Boolsk. |
| Beskrivelse: | Funktionen kalder først getUnitList(), og gennemsøger så det fyldte array efter en værdi som er lig numberOfUnits. Hvis denne værdi findes, vil funktionen returnerer false, da standart tildelingen så allerede er i brug. |