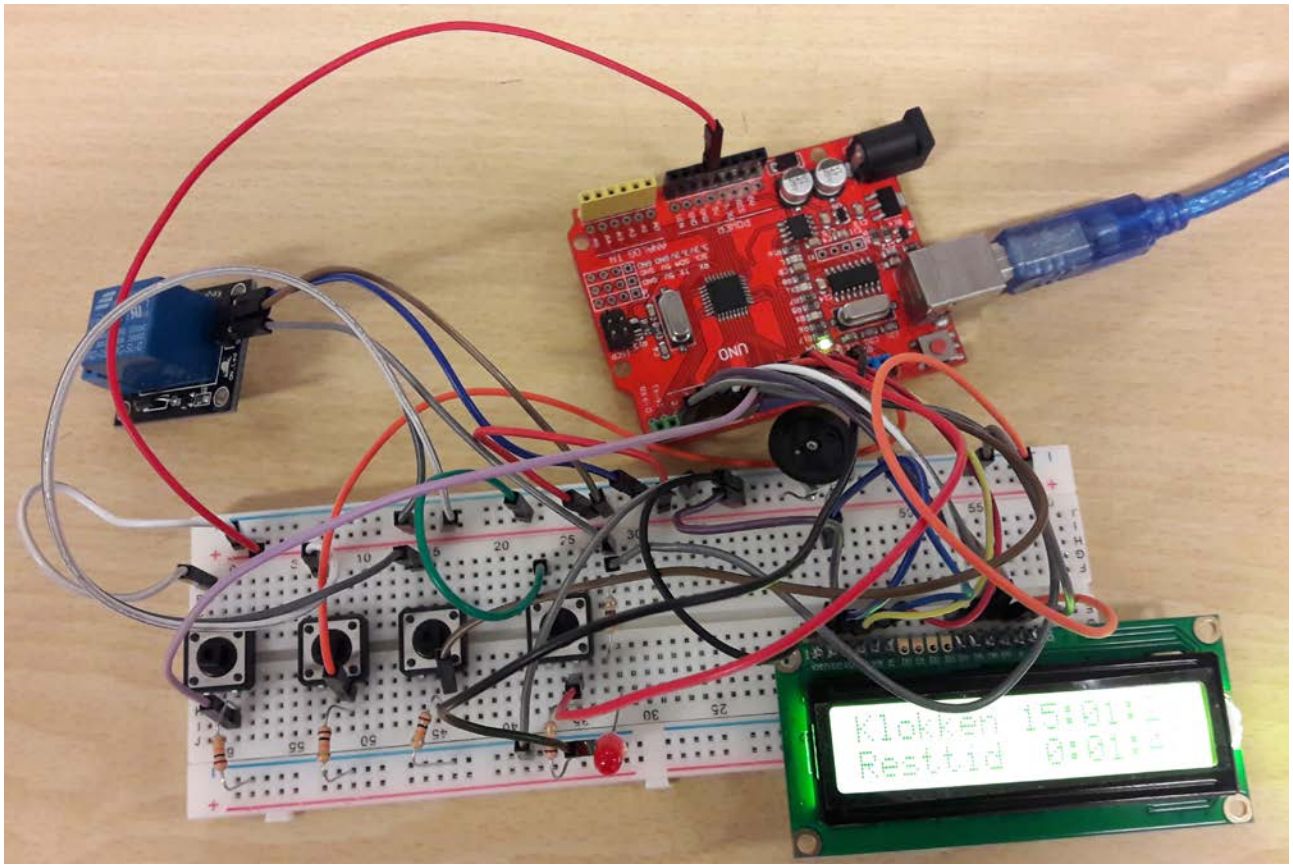
TEKNISK GYMNASIUM
UDDANNELSESCENTER HOLSTEBRO

Dokumentation af UR

Med børnetimer til TV-visning



Bent Arnoldsen

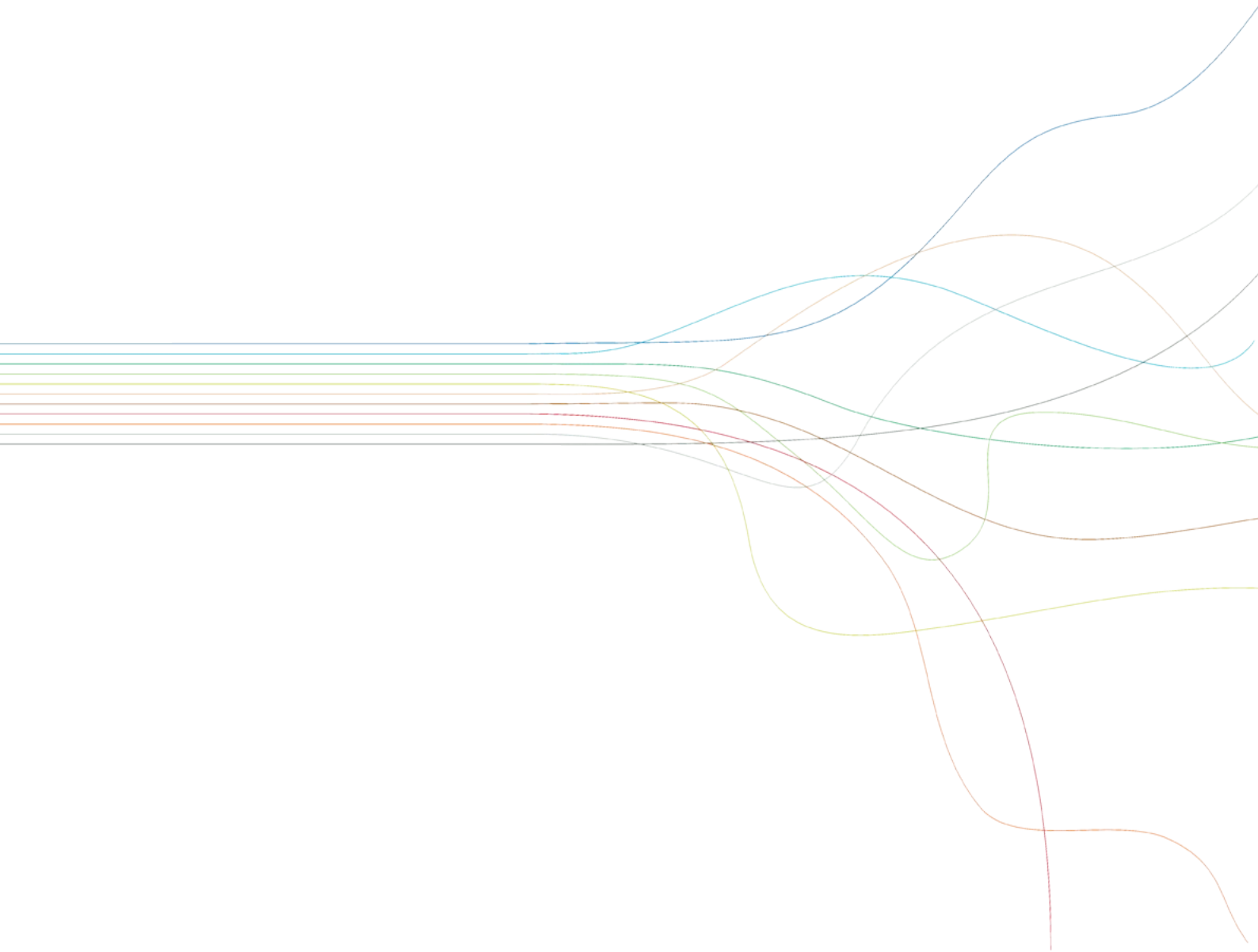
Holstebro HTX

Eksempelrapport

Dækker ind over fagene Teknologi B, Computer- og El-Teknik A og Programmering C

September-Oktober 2018

Indholdsfortegnelse



Indledning

Denne dokumentation er blevet til som resultatet fra et kursus i Arduino-programmering på Holstebro HTX, og er ikke tæt knyttet til et bestemt fag, men låner dokumentationsstandarder fra fagene teknologi, teknikfag-el og programmering. Dokumentationen skal således betragtes som et forslag til hvordan man kunne dokumentere inden for de forskellige fag, og ikke som en skabelon for en rapportform.

Hen gennem dokumentationen er der også indflettet kommentarer om hvordan dokumentationen er opbygget, og hvordan den ellers kunne udformes, altså en form for metakommunikation til dokumentationen.

Problemstillingen er valgt ud fra at der skal kunne illustreres forskellige programmeringsteknikker der viser forskellige måde at løse tingene på i en Arduino. Den konkrete problemstilling kunne være formuleret i følgende: Jeg ønsker et apparat der kan begrænse mine børns TV-tid til 2 timer i døgnet.

Problemformulering

Der ønskes fremstillet en konstruktion (på prototype-niveau) som kan udmåle tiden i et døgn, og som hvert døgn tildeler en fast mængde tid hvor et TV kan være tændt. Konstruktionen skal kunne tænde og slukke for TV'et, så brugeren selv styrer den tildelte tid. Der skal vises både hvad klokken er og hvor lang tid der er tilbage at se TV i.

Udviklingsprocessen

Selve udviklingsprocessen er præget af at det for udvikleren er en ret kendt problemstilling, så der er ikke de store tekniske forhindringer i at udvikle softwaren.

Processen er alligevel taget ud fra et iterativt synspunkt ved hjælp af stepwise improvement (Arnoldsen, 2013d), hvor der trinvist tilføjes nye dele til programmet, hvor der testes og tilrettes i koden til den nye del fungerer og spiller sammen med de resterende dele der er implementeret. Denne metode minder lidt om en agil metode, men minder nok mest om en spiral metode, da nogle dele bliver tilrettet også efter hvordan de næste dele udvikles.

Endelig har processen også lånt noget fra vandfaldsmodellen, da hele den grundlæggende kravspecifikation er sat op inden udviklingsprocessen går i gang.

Kravspecifikation

Ud fra ovenstående problemformulering opstilles følgende kravspecifikation, hvor der defineres et niveau for færdiggørelsen (den skal ikke sættes i produktion):

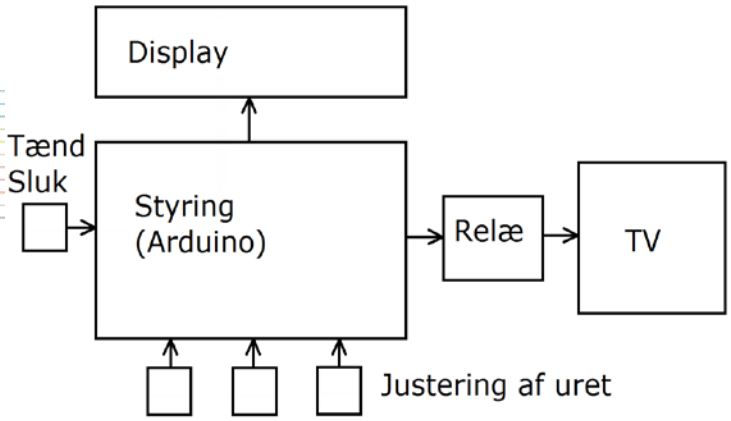
- Konstruktionen skal opbygges som en prototype på fumlebræt
- Uret skal holde styr på tiden i et døgn.
- Der skal være visning af timer, minutter og sekunder i døgnet
- Uret skal være let aflæseligt
- Der skal være mulighed for på brugervenlig vis at justere tiden
- Konstruktionen skal have en udgang der skal kunne tænde og slukke for et TV (230V)
- Der skal tildeles en fast tid (fx 2 timer) hvert døgn, hvor TV'et kan være tændt
- Der skal være en let aflæselig visning af den resterende tid
- Tiden skal tælle ned når TV'et er tændt
- TV'et skal ikke kunne tændes i den resterende del af døgnet, når den tildelte tid er brugt

Ud fra kravspecifikationen lægger det op til at konstruktionen kommer til at bestå af en række dele til at opfylde de forskellige krav, således at man ved hjælp af en central programmerbar enhed (Arduinoen) kan komme til at opfylde de stillede funktionskrav. Konstruktionen vil komme til at indeholde følgende dele:

- En Arduino som centralt element i styringen, den programmeres via en PC, som anvendes under udviklingen, men som kan erstattes med en simpel strømforsyning, når konstruktionen er færdigudviklet.
- Et display der kan vise klokkeslæt og resterende tid i det pågældende døgn.
- En kontakt til at tænde og slukke for TV'et
- Et relæ der faktisk kan tænde og slukke for 230V, så det kan styre om TV'et må være tændt.
- En betjening til at justere uret med.

Blokdiagram

Ud fra kravspecifikationen kan vi opstille et blokdiagram, der illustrerer sammenhængen mellem de dele der indgår i konstruktionen.

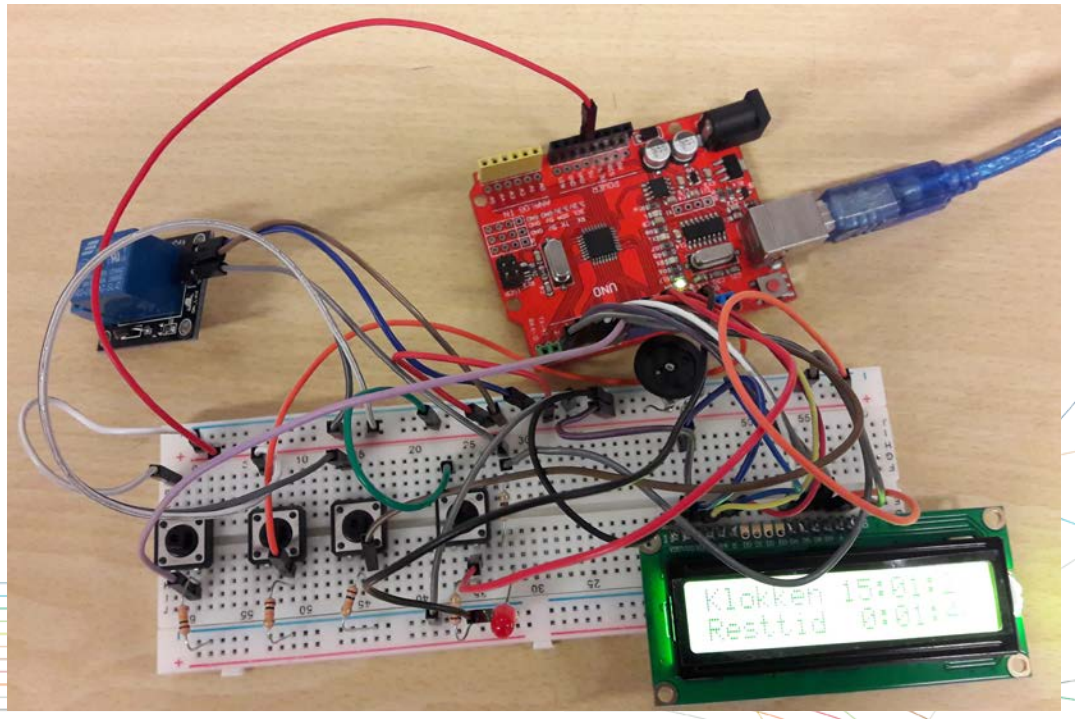


Figur 1. Blokdiagram over konstruktionen af uret med styring af TV-tid

Som det kan ses i blokdiagrammet på figur 1, så er Arduinoen den centrale enhed, og der er ikke indikeret hvor den egentlige tidsregistrering sker, men det tænkes gjort inde i Arduinoens program.

Hardware opbygning

Ud fra kravspecifikationen er der valgt at lave konstruktionen på et fumlebræt, så den ender med at se ud som vist i figur 2.

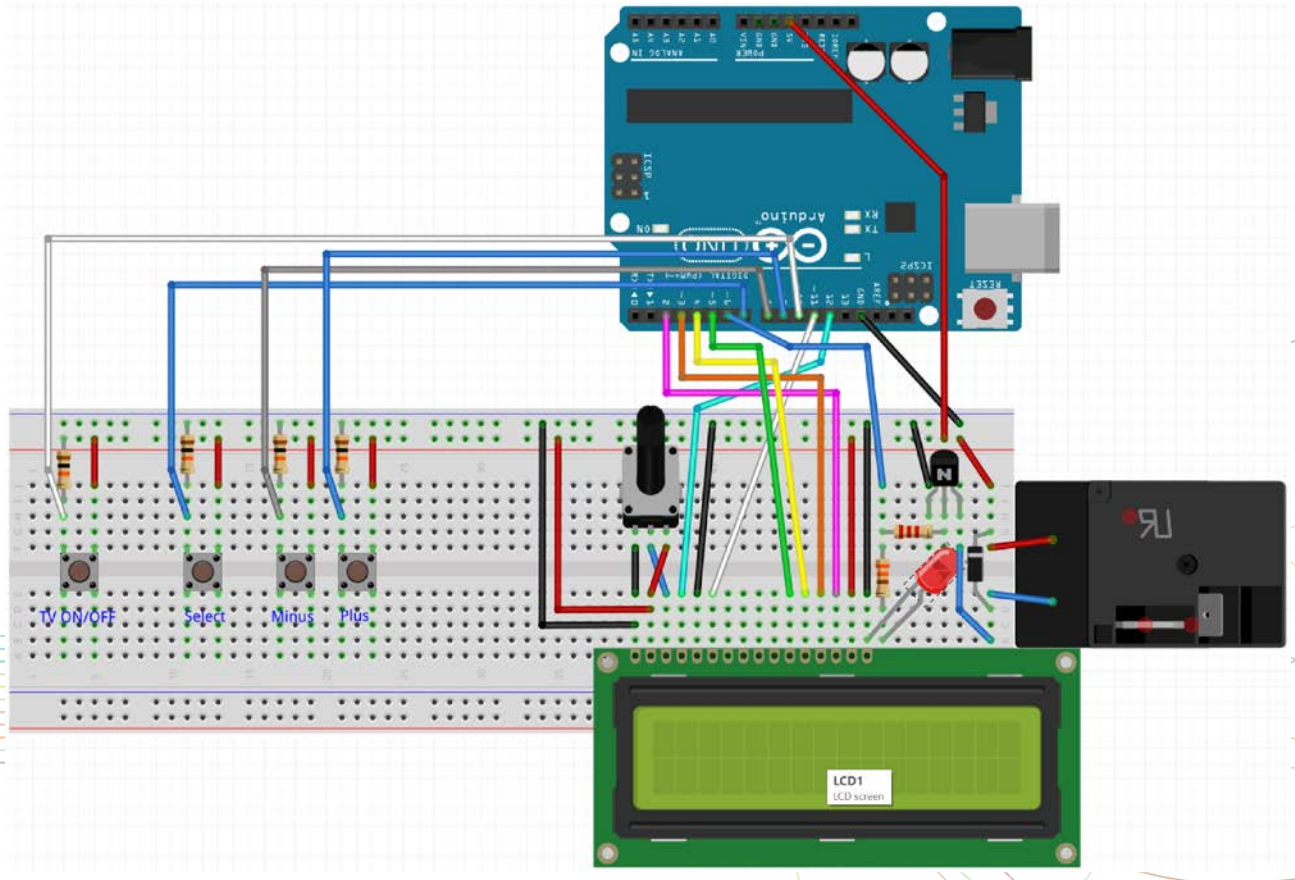


Figur 2. Den færdige prototype opbygget med en Arduino og komponenter på fumlebræt

Hardware dokumentation

For at dokumentere opbygningen af hardwaren vil det aldrig være godt nok at vise et billede af opstillingen som i figur 2. Det vil være et krav i teknikfaget at man anvender diagrammer, men man i teknologi kan vise en pæn skitse der dokumenterer sammenhængen i hardwaren.

Til at dokumentere hvordan hardwaren er bygget op kan man bruge forskellige programmer. I teknologi vil det være helt fint at vise opstillingen bygget i Fritzing (Source, 2018). Denne opbygning kan ses i figur 3.



Figur 3. Tegning af opstillingen lavet ved hjælp af Fritzing

Der kan også trækkes et diagram ud af Fritzing, men den viste breadboard-opbygning vil normalt være dækkende for teknologis vedkommende.

Det vil også hjælpe på dokumentationen hvis der er en kort beskrivelse af fumlebrættet:

- Øverst er der Arduinoen, der indeholder programmet der er udviklet på en PC
- Til venstre på fumlebrættet sidder de 4 kontakter til betjening
- Nederst ses displayet, der også har tilsluttet en kontrastjustering
- Til højre for fumlebrættet sidder relæet, der skal kunne styre 230V til TV'et, og inde på fumlebrættet er der en lysdiode som indikator for at TV'et er tændt samt de komponenter der skal til at drive relæet. I praksis anvendes et modul med transistoren på, men det fandtes ikke i Fritzing.

I teknikfaget vil denne dokumentation være lidt overflødig, men skal i stedet erstattes af diagrammer der fx kan tegnes i Eagle (Autodesk, 2018).

Display

Som display-enhed vælges et LCD-display med 2 linjer på hver 16 karakterer (Arnoldsen, 2017a). Displayet er i stand til at vise både tal og bogstaver, hvilket giver mulighed for at kunne tilføje forklarende tekst.

Denne løsning er valgt ud fra at det er et meget brugt display til Arduino, og at det er relativt let at tilslutte.

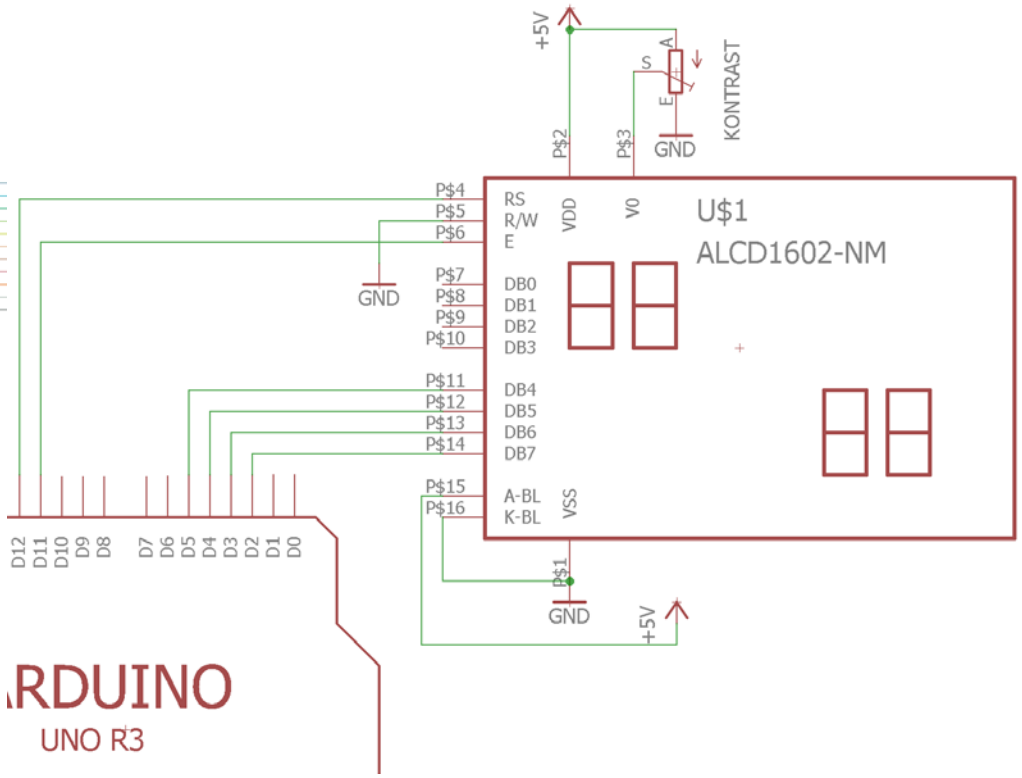
Alternative valg kunne være 7-segment display. Denne løsning er fravalgt, så det ville kræve 6 displays til hvert tid, altså 12 i alt. Dette ville give meget hardware-opkobling, krav til driverkredse og mange forbindelser fra Arduinoen.

Der kunne vælges et 2 linjers display med 8 karakterer på hver – dette ville lige kunne vise det ønskede, men giver meget lidt frihed i designet.

Der kunne vælges et TFT grafisk display (Arnoldsen, 2017b), så man kunne lave noget der minder om et analogt ur, og samtidigt have tekst – det ville endda give muligheden for at lægge betjeningen ind i displayet, da det også har touch-funktion. Dette er fravalgt, da det ville lægge meget af programmerings-fokus på netop denne komponent.

Tilslutning af displayet

Som illustreret i figur 4, så skal displayet kobles op med 6 signal-ledninger, et potentiometer til kontrastjustering og forsyning til både displayet og baggrundslys (A og K).



Figur 4. Deldiagram med tilslutning af displayet.

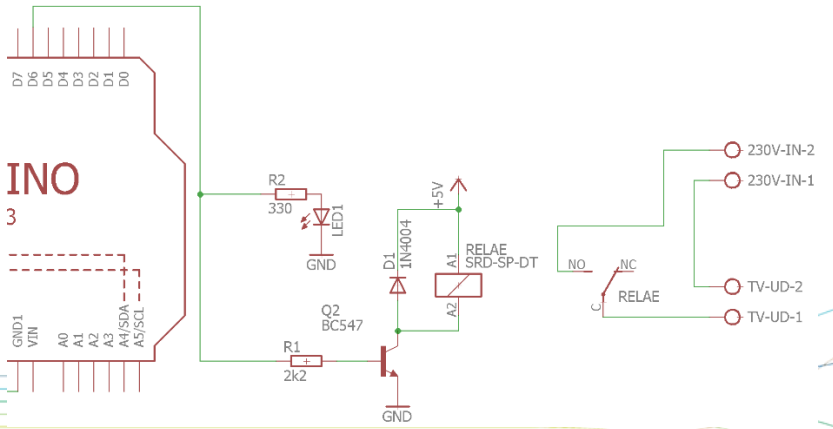
Kommunikationen til displayet er opdelt, så den foregår med 4 bit ad gangen på DB4-DB7, og den kontrolleres med de to kontrolledninger RS og E. Der er også mulighed for at læse i displayets hukommelse, men da det ikke udnyttes sætte R/W til stel, så man kun kan skrive til displayet. Selve skrivningen til displayet håndteres af et bibliotek der kommer med i Arduinoens IDE.

Styring af TV-et med en relæudgang

I sin endelige form tænkes konstruktionen indbygget i et TV, og vil derfor kunne integreres i den styring der sidder i TV'et. I prototypen skal det kunne godtgøres at konstruktionen kan tænde og slukke for TV'et. Dette gøres ved at anvende et relæ der kan arbejde med 230V på kontakterne – i prototypen vil der af hensyn til sikkerheden ved arbejde med 230V ikke blive testet med et TV.

Tilslutning af relæet

I diagrammet er der valgt at indsætte et relæ og en transistor til at drive relæet, da Arduinoen ikke kan levere strøm nok til relæspolen. Dette er illustreret i figur 5.



Figur 5. Deldiagram med relæ-tilslutningen og en Lysdiode som indikator.

I teknikfaget vil man foretage en beregning af modstanden R2 foran lysdioden (Arnoldsen, 2013b), hvor den viste modstand på 330 ohm er beregnet ud fra 10 mA i lysdioden.

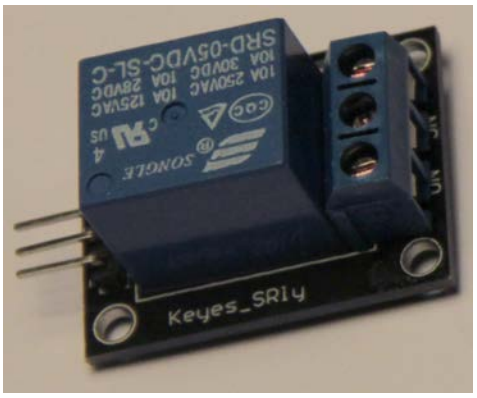
Tilsvarende beregnes formodstanden R1 til transistoren ud fra den strøm relæet kræver og transistorens strømforstærkning. Strømmen til relæet er på 89,3 mA (Relay, u.d.) og strømforstærkningen i transistoren sættes til 40 gange.

Dioden over relæet skal forhindre er der opstår en induktiv spænding fra relæspolen, som kan ødelægge transistoren.

$$I_{R1} = \frac{I_{Relæ}}{H_{FE}} = \frac{89{,}3mA}{40} = 2{,}23mA$$

$$R1 = \frac{V_{Arduino} - V_{BE}}{I_{R1}} = \frac{5V - 0{,}7V}{2{,}23mA} = 1926\ ohm$$

Der vælges en 2,2 kΩ til R1.



Figur 6. Relæmodulet der anvendes

I praksis er der valgt en simplere løsning, hvor der er valgt et kinamodul (Arnoldsen, 2017c) som det viste her til højre i figur 6.

Hvis opstillingen skal laves på print skal man tage højde for banetykkelser og afstande mellem banerne i den del af kredsløbet der skal håndtere de 230V.

Betjening af uret

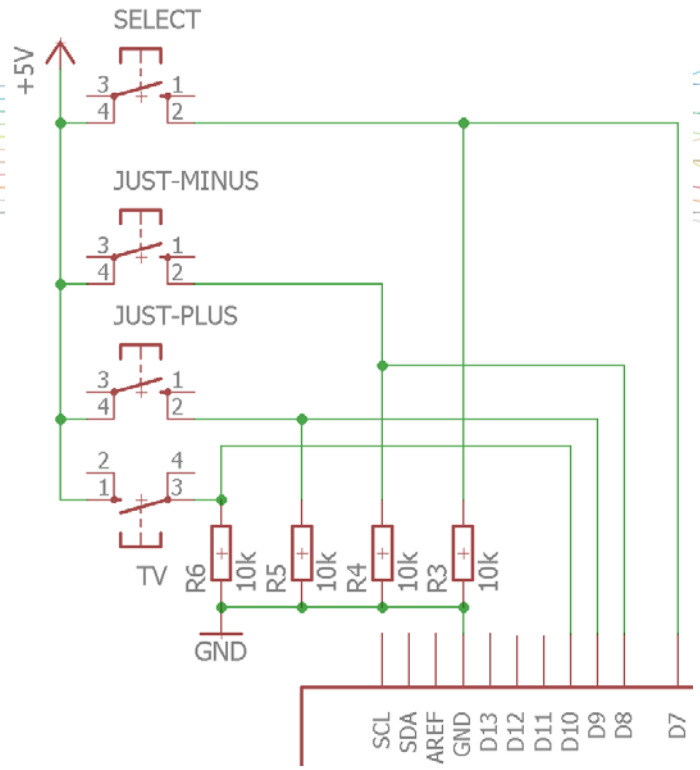
Der er to ting man skal kunne betjene på uret. Man skal kunne tænde og slukke for TV-udgangen og man skal kunne justere timer og minutter i uret, så tiden passer.

Tænd og sluk af uret kunne vælges som en vippeomskifter, så stillingen indikerer om der er tændt eller slukket. Disse kontakter er produktionsmæssigt dyrere, så derfor vælges i stedet en simpel trykknap som er tændt når man trykker på den. Programmet sørger så for at tolke den funktion de enkelte trykknapper skal have.

Til indstillingen af uret vælges en anden trykknap til at angive om der skal justeres og om det er timer eller minutter der skal justeres. For at gøre det brugervenligt skal brugeren justere både op og ned på det man justerer. Det kunne gøres med en Rotary Encoder (Arnoldsen, 2013c), hvor man kan registrere hvilken retning man drejer i, som så kunne skrue op og ned for tallet. I stedet vælges to simple knapper til op og ned.

Tilslutning af betjening

Overvejelserne omkring betjeningen af uret fører til deldiagrammet i figur 7.



Figur 7. Betjeningsknapper tilsluttet Arduinoen

Alle knapper er tilsluttet +5V, så de giver et højt signal når der trykkes på dem. Når der ikke er trykket på knapperne vil indgangene på Arduinoen bare svæve, så der sættes en modstand på hver kontakt der kan trække signalet til stel. Størrelsen af denne modstand er svær at beregne, men en erfaringsværdi på 10-100 kΩ kan anvendes, her er valgt 10 kΩ.

Totaldiagram

Et samlet diagram over hele opstillingen kan ses i Bilag 2.

Strømforsyning til opstillingen

I hele testfasen er Arduinoen tilsluttet en PC, så den får 5V fra USB-porten, hvilket er i stand til at forsyne hele opstillingen.

I en driftsfase vil det naturligvis ikke være smart at skulle have en PC stående sammen med produktet, så her vil man kunne anvende en 9V adapter (Arnoldsen, 2013a) til at forsyne Arduinoen med som vist i figur 8.

Figur 8. Eksempel på adapter der kan forsyne en Arduino

Grænsefladespecifikation

Den grænseflade der dokumenteres her er mellem den omliggende elektronik og Arduinoen, så softwaren ved hvilke ben der anvendes til hvad.

Ben	Funktion	Retning	Kommentar
D0	Rx	Input	Anvendes til kommunikation med PC'en
D1	Tx	Output	Anvendes til kommunikation med PC'en
D2	DB7	Output	Databen til Displayet
D3	DB6	Output	Databen til Displayet
D4	DB5	Output	Databen til Displayet
D5	DB4	Output	Databen til Displayet
D6	TV_OUT	Output	Styreben til relæet – TV
D7	Select Juster	Input	Vælger justerings-mode: Vis Ur / Juster Timer / Juster Minutter
D8	Juster Minus	Input	Justering af Timer/Minutter nedad
D9	Juster Plus	Input	Justering af Timer/Minutter opad
D10	Tænd/Sluk TV	Input	Tænder og slukker for TV-udgangen (hvis der er tid tilbage)
D11	En	Output	Enable – styresignal til displayet
D12	RS	Output	Register Select - styresignal til displayet
D13			Disponibel / LED på Arduino-boardet
+5V	Forsyning	Power	Forsyning til Display, relæ og kontakter
GND	Stel	Power	Referencepunkt for signaler og forsyning
A0-5		Analog In	Disponible ben – kan også anvendes til digital I/O

Tabel 1. Ben-definitioner på Arduinoen

Denne grænsefladespecifikation der vises i tabel 1 er den beskrivelse der skal bruges til softwaren.

Test

En test kan afgøre flere forskellige ting, og det er vigtigt at reflektere over hvad det egentlig er man tester.

I denne dokumentation er der fokuseret på at teste softwaren, men dette vil selvfølgelig også indbefatte at man tester via hardwaren, så den bliver faktisk også testet Hardwaren vil også have indflydelse på testen.

Når man lige har programmeret en ny funktion/del ind i koden, så er det en god ide at teste om det man har lavet opfører sig som forventet. Dette er den helt basale form for test.

Den næste form for test er at man prøver at udsætte produktet for forskellige fejlbetjeninger/underlige situationer. Det kan virke lidt underligt, men det er med til at teste om produktet fungerer robust, og ikke laver uventede ting. Det kan være svært at forestille sig hvilke fejlbetjeninger et produkt udsættes for, men det er ikke desto mindre vigtigt at teste det, for at kunne lave et ordenligt produkt.

For at kunne dokumentere denne form for test, så er man nødt til at beskrive hvordan man tester de enkelte fejlsituationer, og hvilket resultat testen giver, samt om det er acceptabelt i forhold til kravspecifikationen.

Den sidste form for test man skal udføre er en bruger-test, hvor man tager en vilkårlig bruger fra den tiltænkte målgruppe for produktet, og hvis testen skal give mening, så må brugeren ikke kende produktet på forhånd. Denne test er virkelig vigtig på et færdigudviklet produkt, men det er ikke en god ide at vente med denne test til produktet er lige før det går i produktion, da det på dette tidspunkt kan være meget dyrt at lave rettelser på produktet.

Det er heller ikke altid lige let at lave bruger-test på en prototype, da der tit mangler navne på knapper og produktet ikke ser ud som man forventer at et produkt skal se ud. Dette skal man forsøge at kompensere for i testen – her skal man dog passe på at man ikke kompenserer for meget, da det vil gøre at man måske hjælper brugeren uden om nogen faldgruber i produktet, som ellers kunne være fejl man skulle have rettet.

Det gode ved en bruger-test er at man kommer ud i situationer som man ikke lige havde tænkt over, samt at man får fokus på brugerens behov. Denne måde at anvende test på er det der har ført til agile (Wikipedia, 2018a) udviklingsprincipper.

Det der er rigtigt vigtigt ved en brugertest er at man forholder sig til hvad brugeren siger. Det kan tit være en god ide at optage testen, så man kan se hvad det er der sker, høre hvad brugeren tænker og evt. kan genskabe fejlsituationer. Den største fejl man normalt begår ved bruger-testen er at man afskriver fejlsituationer ved at tænke at så dumme kan brugere ikke være – man skal tage det alvorligt, hvis en bruger ønsker en anderledes betjening, så skal man prøve at lave det, så længe det ikke går ud over andre krav til produktet, selvom det også kan virke besværligt.

Man skal også notere hvad brugeren finder smart og godt – det er vigtigt at disse funktioner bliver bevaret, når man går i gang med at rette andre fejl/uhensigtsmæssigheder.

Software opbygning

Softwaren er opbygget og testet hen ad vejen som et udviklingsprojekt ved hjælp af Stepwise Improvement, altså ud fra princippet at man starter helt fra bunden og så kobler flere og flere funktioner på programmet.

Der er både fordele og ulemper ved denne metode. Fordelen er klart at man kan overskue hvad man laver, og når man ikke har så meget programmeringserfaring, også skal lære teknikkerne undervejs. Det letter også fejlfindingen at man ved hvilken del man har rettet i, og dermed hvor man skal begynde med at lede efter fejl.

Selvom det var dette princip, så startede udviklingen med at sætte et overblik over hvad det færdige projekt skulle ende med at kunne. Dette låner principper fra vandfalds-modellen (Wikipedia, 2018b), at man opstiller alle krav fra starten, gennemfører kodningen og foretager en sluttest af produktet.

Hoved-loopet

I hovedloopet er det fokus på at holde fast i et overblik over koden, så her er koden begrænset til at kalde forskellige funktioner der så løser de enkelte ting.

Dette kan udtrykkes i det flowchart der er i figur 9.

Det fører til en kode i `loop()` der ser ud som følger:

```
// Hovedloopet der gennemløbes igen og igen
void loop() {
  taelTid();
  udskrivSerial();
  justerTid();
  taendSluk();
  visDisplay();
}
```

Figur 9. Flowchart over hovedloopet

Denne kode afspejler flowchartet på nær en enkelt ting, nemlig ventetiden på 1 sekund. Dette er nærmere gennemgået i afsnittet Håndtering af tid.

I starten laves der bare 5 tomme funktioner der bare skal gøre at koden kan oversættes. Sådan en tom funktion kan se ud som følger:

```
void taelTid(){
  ;
}
```

Start-koden – setup()

Ud fra grænsefladespecifikationen er der defineret en række bennumre, som blot angiver hvilket bennummer der er knyttet til en bestemt funktion. Dette er defineret som følger:

```
// Konstanter der definerer input og output ben
const byte TV_OUT = 6;
const byte select = 7;
const byte justerMinus = 8;
const byte justerPlus = 9;
const byte skiftTV = 10;
```

Disse definitioner er lavet inden `setup()` der skal sætte alle dele op til at fungere inden loop() begyndes at blive kaldt. Definitionerne er lavet for at input og output kan sættes op med pinMode(navn, mode). I sin endelige version indeholder `setup()` forskellige ting, der skal startes inden hoved-loopet kan fungere. Disse ting er den serielle port, så vi kan skrive til PC'en hvad der foregår i programmet og der er displayet som skal startes, og der skrives en velkomstbesked i displayet, for at identificere programmet. Dette er vist i følgende kode:

```
void setup() {
  // Seriel port til visningen i starten og til test senere
  Serial.begin(9600);
  // Start displayet - antal karaterer og linjer
  lcd.begin(16, 2);
  // Velkomst-besked.
  lcd.print("TV Timer UR");
  lcd.setCursor(0,1);
  lcd.print("Version 1.0");
  delay(2000);
  lcd.clear();
  // Sæt ind og udgange op til kontakter og relæet til TV'et
  pinMode(select, INPUT);
  pinMode(justerMinus, INPUT);
  pinMode(justerPlus, INPUT);
  pinMode(skiftTV, INPUT);
  pinMode(TV_OUT, OUTPUT);
}
```

Ud over selve `setup()` sker der også en del definitioner af variabler og kontakten til display-koden, dette bliver dokumenteret sammen med beskrivelsen af de forskellige dele af koden.

Optælling af uret i sekunder, minutter og timer

I starten af udviklingen arbejdedes med selve det at få tiden til at tælle fornuftigt, så der var tidsregistreringen baseret på at lave en delay(1000) der blot venter et sekund, så loop() blev gennemført ca. en gang i sekundet

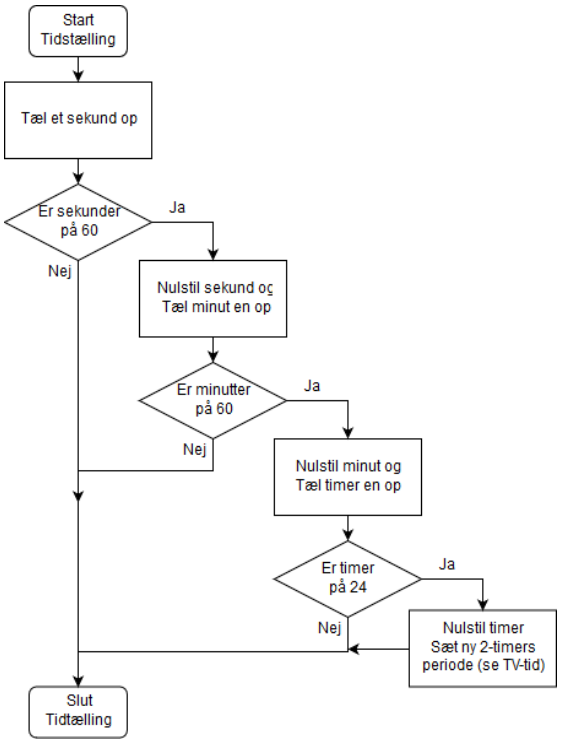
Til at huske tiden, så er der lavet 3 variabler sekunder, minutter og timer, der skal kunne indeholde tal fra 0 til 59 eller fra 0 til 23, derfor er typen valgt til en byte der kan indeholde 0 til 255.

Variablerne sættes til et tidspunkt der er valgt tilfældigt – Arduinoen har ikke indbygget en mulighed for at lade tiden fortsætte når den er slukket. Skulle man have lavet den funktion ville det kræve en Real Time Clock (Arnoldsen, 2018a), der er et separat modul med batteribackup, som sikrer at tiden tælles videre med Arduinoen er slukket.

```
// Variabler indeholdende tiden
byte sekunder = 55;
byte minutter = 59;
byte timer = 14;

unsigned long tidsRegistrering = 0;
boolean taelSekund = false;
```

Håndteringen af tiden kan illustreres i flowchartet i figur 10, der er implementeret i koden som vises herunder.



Figur 10. Flowchart over optælling af tid

En del af funktionen håndterer større præcision i tidsmålingen, men er udeladt her af hensyn til overskueligheden. Dette dokumenteres i næste afsnit.

```
void taelTid() {  // Funktion der håndterer tiden
  // .. .. Her er udeladt kode der forklares i næste afsnit
  // Håndter tællingen af tid
  sekunder = sekunder + 1;
  if (sekunder == 60) {
    sekunder = 0;
    minutter = minutter + 1;
    if (minutter == 60) {
      minutter = 0;
      timer = timer + 1;
      if (timer == 24) {
        timer = 0;
        restTid = doegnTid;
      }
    }
  }
  // .. .. Her er udeladt kode der forklares i næste afsnit
}
```

Præcis håndtering af tiden

Som nævnt håndteredes tiden i starten af udviklingen ved blot at lave en delay(1000) i hvert gennemløb af koden. Problemet med denne metode er at afviklingen af den anden kode i loopet også tager tid. Det kunne man kompensere ved at man tilpasser delay indtil tiden går rigtigt. Det vil kræve meget tilpasning og lang tids test indtil man finder den rette tid, og hvis man så retter noget i koden, så skal man processen igennem igen.

Den teknik der anvendes nu i den ovenstående kode er at anvende en indbygget funktion `millis()` (Arduino, 2017) til at registrere hvor lang tid der er gået. millis() returnerer det antal millisekunder der er gået efter reset. For udnytte den anden kode der er skrevet så registreres der hvornår vi sidst har talt et sekund op i variablen tidsRegistrering, der er af samme type som millis() returnerer – en unsigned long, der kan indeholde positive tal op til ca. 4.200.000.000.

Når koden tæller et sekund frem, så lægges der 1000 til tidsregistrering, så den ved hvornår der skal tælles frem næste gang, ellers returnerer koden bare, så der ikke sker andet. Dette sikrer at hvis ellers koden kan nå at kalde taelTid(), så bliver der talt et sekund frem hvert sekund, også selvom det rammer lidt skævt.

```
// Indiker at der ikke er gået et sekund
taelSekund = false;
// Fortsæt kun, hvis der er gået et sekund.
if (millis() < tidsRegistrering) {
  return;
}
// Stop tidstælling, hvis der justeres på tiden.
if (selectMode > 0) {
  tidsRegistrering = millis();
  return;
}
tidsRegistrering += 1000;  // sæt tidsregistreringen et sekund frem
// .. .. Her står koden med håndtering af tiden, der tæller sekunder osv.

// Signal til de andre rutiner om at der er gået et sekund
taelSekund = true;
```

For at sikre at tiden ikke opfører sig underligt mens man indstiller uret, så afbrydes tidstællingen men der justeres på uret. Det gøres ved at se om selectMode angiver at indstillingen er i gang.

Den sidste ting der sket i taelTid() er at hver gang der er talt et sekund frem, så er taelSekund sand, ellers er den falsk. Det gør at andre rutiner der også skal gøre noget hvert sekund også kan se at der er gået et sekund.

Kommentarer til overløb af millis()

Som det er dokumenteret på www.htx-arduino.dk så er der et problem med den måde tidsmålingen bliver lavet på her (Arduino, 2018b).

Problemet er at variablen `tidsRegistrering` vil løbe over før `millis()` gør det, og indtil `millis()` løber over, så vil den være størst, og den vil så tælle et sekund frem hver gang funktionen kaldes, hvilket vil gøre at tiden ”springer” et stykke frem, indtil `millis()` løber over inden for et sekund.

Variablen `tidsRegistrering` vil så være talt et pænt stykke frem, hvilket vil gøre at uret ”står stille” indtil `millis()`er kommet frem til det rigtige tidspunkt. Tiden vil rette sig selv, men opførslen er forkert.

Måden det skal håndteres på er at registrere hvornår et sekund er startet, og så se om `millis()` er kommet et sekund længere. Det gør at overflowet bliver håndteret i denne beregning, og hele tiden giver det rigtige resultat.

Variablerne bør navngives lidt anderledes fx som følger:

```
unsigned long startSekund = 0;
unsigned long sekundInterval = 1000;
```

Med disse variabler vil koden kunne laves som følger:

```
  // Indiker at der ikke er gået et sekund
  taelSekund = false;
  // Fortsæt kun, hvis der er gået et sekund.
  if ((millis() - startSekund) < sekundInterval) {
    return;
  }
  // Stop tidstælling, hvis der justeres på tiden.
  if (selectMode > 0) {
    tidsRegistrering = millis();
    return;
  }
  startSekund += sekundInterval;  // sæt tidsregistreringen et sekund frem
  // .. .. Her står koden med håndtering af tiden, der tæller sekunder osv.

  // Signal til de andre rutiner om at der er gået et sekund
  taelSekund = true;
```

Det vil ikke være let at teste denne del af koden, da det først sker efter ca. 49 dage og 17 timer, så med mindre man vil til at manipulere systemvariablerne, så er det ret besværligt at teste denne problematik.

Udskrift af tiden i Serial Monitor

I starten af udviklingen er det let at få noget ud i Serial Monitor (Arduino, 2018a), så der udskives tiden ved hjælp af udskrivSerial() funktionen. Koden i funktionen ser ud som følger:

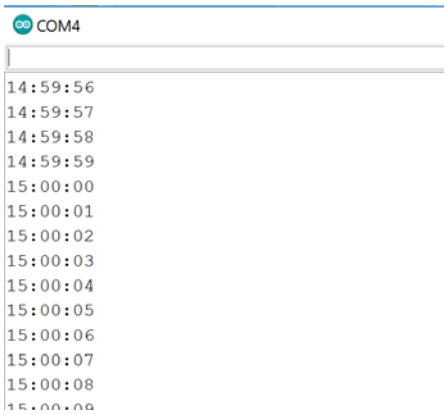
```
void udskrivSerial() {  // Funktion der udskriver tiden
  if (! taelSekund) {
    return;
  }
  // Udskriv tiden til Seriel-porten i formatet hh:mm:ss - minutter og sekunder
anvender foranstillede 0'er
  Serial.print(timer);
  Serial.print(":");
  if (minutter < 10) {
    Serial.print(0);
  }
  Serial.print(minutter);
  Serial.print(":");
  if (sekunder < 10) {
    Serial.print(0);
  }
  Serial.println(sekunder);
}
```

Funktionen udskrivSerial() anvender ikke egne variable, men bruger blot de variable som taelTid() har defineret.

Der fortsættes kun i funktionen hvis der er talt et sekund op, ellers ville der komme voldsomt meget i den serielle monitor.

Der udskrives i formatet ”hh:mm:ss”, hvor der indsættes 0 foran minutter og sekunder, hvis de er under 10, så tiden har et fast format.

Den serielle monitor viser tiden som vist i figur 11:



Figur 11. Visning af tiden i seriel monitor.

Test af tiden i uret

Her testes hvordan tiden forløbet i uret, og det svarer til den forventede opførsel.

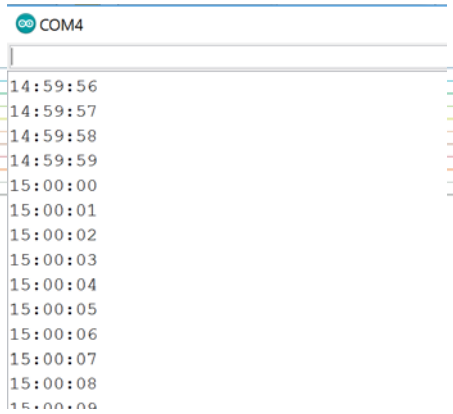
Testen er i første omgang udført på den serielle monitor, og senere på displayet som det er illustreret på figur 12. De test der er udført er beskrevet i tabel 2.

Figur 12. Visning af tiden i displayet.

Testbeskrivelse	Forventet resultat	Res. midlertidig test	Resultat Sluttest
Sekunder tæller op	At de gør	Det gør de	Det gør de
Sekundskifte fra 59 til 00 skal tælle minutter op	At det sker	Det sker	Det sker
Minutskifte fra 59 til 00 skal tælle timer en op	At det sker	Det sker	Det sker
Timeskifte fra 23 til 00 skal ske, og TV-tiden skal stilles til 2 timer for næste døgn.	At det sker	Det sker med test-tid 1 minut og 40 sekunder	Det sker med test-tid 1 minut og 40 sekunder
Præcision af uret	At uret ikke afviger mere end et sekund i døgnet	Svingende test-resultater under udviklingen – Ikke testet ordentligt	Ikke testet ordentligt, men afviger ikke over 1 sekund pr. time[1]

Tabel 2. Test af tidens forløb i uret

I figur 13 ses en af testene for tiden, der dokumenterer at både sekund og minut skiftene fungerer som de skal. Desuden dokumenteres at justeringen af formatet med foranstillede nuller fungerer.



Figur 13. Udskrift på serial monitor, der viser at overløb af sekunder og minutter fungerer korrekt

Illustrationen i figur 13 efterviser desuden af den serielle udskift fungerer som ønsket – skulle man ønske mere kunne man også have en udskrift af resttiden samt af status på TV'et.

Justering af tiden

I kravspecifikationen er der defineres at der skal kunne vælges at justere minutter og timer for sig, og at der skal kunne justeres både op og ned. Dette er valgt ved at en knap vælger justeringsmode og to knapper justerer hhv. op og ned.

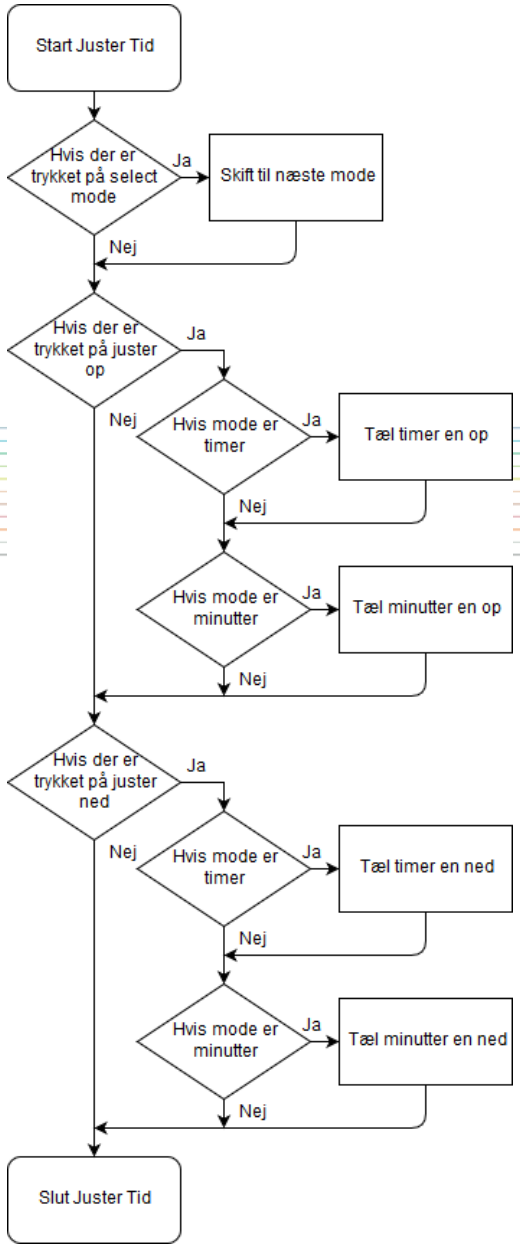
Da knapperne skal kunne reagere hurtigt på brugerens tryk registreres forkanten (Arnoldsen, 2018b) at et tryk på knappen skal tælle en fremad, så skal der til de enkelte knapper defineres variabler som kan husk hvilket stadie knappen var i sidst justerTid() blev kaldt.

Der skal også være en variabel selectMode, der er 0 når uret kører normalt, 1 når der justeres timer og 2 når der justeres minutter.

[1] Denne test afhænger af hvor præcist Arduinoens krystal er, så det kan svinge op til +/- 1 %. Ønsker man større præcision kan man anvende en real-time-clock som DS3231 der har en unøjagtighed på 2 ppm.

```
// Variabler til justering af tiden og display-visning
boolean lastSelect = false;
byte selectMode = 0;
boolean lastPlus = false;
boolean lastMinus = false;
```

Et groft flowchart over funktionen er vist i figur 14. Flowchartet er lavet ud fra at trykket betyder at der er modtaget en forkant på knappen



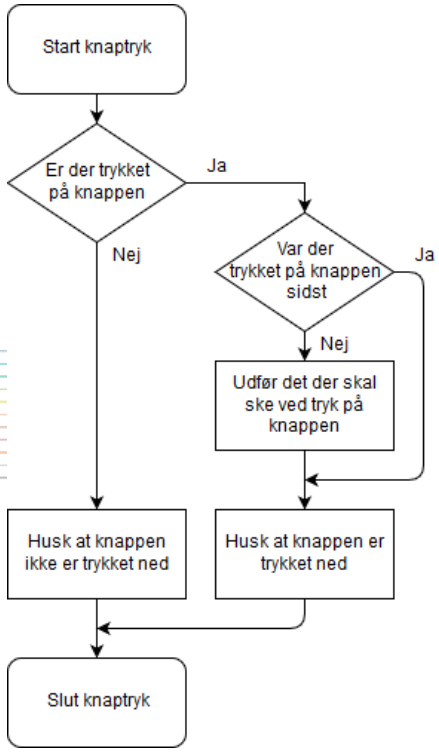
Figur 14. Groft flowchart over justering af tid

I `visDisplay()` håndteres tiden for opdateringen af displayet, og dette spiller sammen med justerTid() ved hjælp af de følgende variabler, hvor `visJuster` angiver om det tal man er ved at justeres skal vises, så det står at blinker. Variablen `timeOut` sørger for at den går ud af justermode, hvis man ikke betjener den

og `justerCount` holder styr på hvor mange gange der er talt op mens man holder knappen ned, så den kan justere hurtigere efter den har talt 5 langsomt frem. Disse variabler bliver nulstillet i justerTid().

```
boolean visJuster = false;
int timeOut = 0;
int justerCount = 0;
```

Det at fange forkanten af knaptrykket er lavet ud fra flowchartet i figur 15. Ideen med kun at reagere på forkanten er at der kun skal reageres en gang på hvert knaptryk. Flowchartet er lavet generelt for et knaptryk, og princippet anvendes på de 3 knapper i justerTid().



Figur 15. Generelt flowchart over tryk på en knap

Herunder ses koden der vælger hvilket mode man befinder sig i, hvor der anvendes princippet i at finde en forkant på knaptrykket. Ved registreringen af forkanten tælles selectMode en frem og begrænses til 0, 1 eller 2. Variablerne omkring display af justeringen nulstilles når der skiftes mode. Til sidst er der indført et lille delay på 30 ms, dette er et forsøg på at eliminere at der er noget prel (Arnoldsen, 2018c) på trykknapperne.

```
// Vælg om der skal justeres - bestemmes af selectMode
 if (digitalRead(select)) {
   if (! lastSelect) {
     // Tæl justeringsmodet en frem og begræns til 0, 1 og 2
     selectMode++;
     selectMode %= 3;
     timeOut = 0;
     justerCount = 0;
     visJuster = false;
     delay(30);  // eleminer prel
   }
```

```
    lastSelect = true;
  } else {
    lastSelect = false;
  }
```

Når selectMode er på 1 eller 2, så kan uret justeres. Der er to knapper til at justere hhv. positivt og negativt. De to koder minder ret meget om hinanden. Der bruges samme teknik som før, som finder forkanten af et tryk på knappen, og alt efter selectMode justeres så timer eller minutter en op (eller en ned i den negative del af koden).

```
  // Håndter positiv justering af uret - afhængig af selectMode
  if (digitalRead(justerPlus)) {
    if (! lastPlus) {
      delay(30);  // eleminer prel
      if (selectMode == 1) {
        timer = timer + 1;
        if (timer == 24) {
          timer = 0;
        }
      }
      if (selectMode == 2) {
        minutter = minutter + 1;
        if (minutter == 60) {
          minutter = 0;
        }
      }
    }
    lastPlus = true;
  } else {
    lastPlus = false;
  }
```

Test af justeringen af uret

Justeringen af tiden er lavet ud fra at man sætter den i justeringsmode, og at man så kan skifte mellem time- og minut-justering. Begge dele kan justeres både op og ned, og hvis man holder knappen inde skal den først tælle langsomt, derefter hurtigt. Endelig skal justerings-mode afsluttes, hvis man efterlader uret inde i justering, da uret er stoppet i dette mode.

De test der er udført er beskrevet i tabel 3.

Testbeskrivelse	Forventet resultat	Res. midlertidig test	Resultat Sluttest
Skift mellem visning af uret og justering af hhv. timer og minutter	Skifter Ur – timer – minutter – Ur	Det virker nogenlunde, men problemer med prel	Prel er ikke løst
Justering op i timer / minutter	At der lægges en til ved hvert tryk	Det virker nogenlunde, men problemer med prel	Prel er ikke løst
Justering ned i timer / minutter	At der trækkes en fra ved hvert tryk	Det virker nogenlunde, men problemer med prel	Prel er ikke løst
Langsom justering ved hold	Den tæller ”roligt” op / ned	Det virker	Det virker
Hurtig justering ved hold	Efter 5-6 langsomme trin tæller den ”hurtigt” op / ned	Det virker	Det virker
Exit fra justering ved passivitet	At den går tilbage til Ur-visning efter passende tid	Det virker	Går ud efter ca. 20 sekunder og viser Ur

Tabel 3. Test af justering af uret

Problemerne med prel formodes at komme fra at kontakterne sidder løst i fumlebrættet, så det vurderes ikke at det vil være et problem, hvis man laver opstillingen på print. Skulle der stadig være prel-problemer i kontakterne, så kan man forsøge at eliminere det yderligere i softwaren, men erfaringen siger at det normalt er nok med de 30 ms der er indført i softwaren.

Visning i Display

Selve den elektrisk kommunikation med displayet er ret kompliceret, da alle bytes der skrives til uret bliver delt op i 2 gange 4 bit og sent til displayet ved hjælp af de to ledninger RS og EN.

Hele denne kommunikation er håndteret af et bibliotek LiquidCrystal (Arduino, 2018b) der kommer sammen med Arduinoens IDE, og benforbindelserne der er anvendt, er taget fra et af eksemplerne til displayet Hello World (Arduino, 2018a).

Det er fra denne kode at der er taget den grundlæggende setup af displayet i følgende kode:

```
// Biblioteket til displayet
#include <LiquidCrystal.h>

// Displayets objekt lcd - med bendefinitioner
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

Koden henviser til biblioteket med include.

Derefter defineres hvilke ben der anvendes til displayet ved at navngive dem som konstanter

Til sidst oprettes et objekt ved navn `lcd` som kan kommunikere med displayet via forskellige metoder.

Til at håndtere hvor tit der skal skrives til displayet oprettes en variabel der indeholder tidspunktet i millisekund for den næste visning af displayet

```
unsigned long displayVisning = 0;
```

Denne variable anvendes også til timingen af justeringen af uret, når man ønsker justere hurtigt, så er det denne displaytid der tilpasses om justeringen af uret skal gå langsomt frem eller løbe hurtigere.

Det er funktionen visDisplay() der skriver i displayet, og som lige forklaret er det variablen displayVisning der bestemmer hvornår displayet skal opdateres. Dette sker i starten af funktionen som vist her:

```
void visDisplay() {  // Visning på et display
  // Opdater displayet efter det valgte interval (efter hvad visningen angiver)
  if (millis() < displayVisning) {
    return;
  }
```

Display-visningen er styret af selectMode, der bestemmer om man er i gang med at justere uret. Hvis man ikke er i gang med at justere, så vises tiden og der sættes en opdatering til 250 ms senere – det viste sig at den ikke bare skulle opdatere hvert sekund, da det kunne gøre at den ville springe et sekund over en gang imellem.

Selve tidsvisningen starter med at rense displayet, og skriver derefter tiden med lcd.print().

```
  // Visningen af tiden og resttid
  if (selectMode == 0) {
    // Opdater 4 gange i sekundet - ellers kan displayet springe sekunder over
    displayVisning += 250;
    lcd.clear();
    // Print visningen af tiden i displayet - visning hh:mm:ss
    lcd.print("Klokken ");
    if (timer < 10) {
      lcd.print(" ");
    }
    lcd.print(timer);
    lcd.print(":");
    if (minutter < 10) {
      lcd.print("0");
    }
    lcd.print(minutter);
    lcd.print(":");
    if (sekunder < 10) {
      lcd.print("0");
    }
    lcd.print(sekunder);
```

Resttiden af TV-tid skrives herefter i den næste linje, bestemt af setCursor(0,1).

Da resttiden er lagret som antal sekunder tilbage, så konverteres den tid først til timer, minutter og sekunder i lokale variable, inden der skrives ud til displayet på samme måde som urets tid.

```
    lcd.setCursor(0,1);
    lcd.print("Resttid ");
   // Konverter resttiden i antal sekunder til timer minutter og sekunder tilbage
    byte restSec = restTid % 60;
    byte restMin = restTid / 60;
    byte restTim = restMin / 60;
    restMin = restMin % 60;
    // Print den resterende tid i displayet - visning hh:mm:ss
    if (restTim < 10) {
      lcd.print(" ");
    }
    lcd.print(restTim);
    lcd.print(":");
    if (restMin < 10) {
      lcd.print("0");
    }
    lcd.print(restMin);
    lcd.print(":");
    if (restSec < 10) {
      lcd.print("0");
    }
    lcd.print(restSec);
  }
```

Hvis man er i gang med at justere uret, så er det valgt at det kun er timer eller minutter der skal vises, og at de vises i den position som de normalt står i, mens der vises hvad man justerer.

Den første del er den samme uanset om man justerer timer eller minutter, så der er skrevet sammen til funktionen justeringValg() som forklares senere, men ellers er det ved timerne følgende kode der giver visningen i displayet, hvor der startes med at slette displayets indhold med lcd.clear(), hvorefter der skrives Timer og så variablen timer, hvorefter der skrives de to : som normalt deler tiden, så man kan se hvad det er man justerer på.

```
  // Visning af kun timer hvis der justeres timer
  if (selectMode == 1) {
    justeringValg();
    lcd.clear();
    lcd.print("  Timer ");
    if (visJuster) {
      if (timer < 10) {
        lcd.print(" ");
      }
      lcd.print(timer);
    } else {
      lcd.print("  ");
    }
    lcd.print(":");
    lcd.print("  ");
    lcd.print(":");
  }
```

Dette giver følgende i figur 16 (taget i en test før der blev tilføjet teksten Timer)

Figur 16 Visning af timer under justering af uret

Visningen af minutter under justering er lavet ud fra samme principper som det ses her:

```
  // Visning af kun minutter hvis der justeres minutter
  if (selectMode == 2) {
    justeringValg();
    lcd.clear();
    lcd.print("   Min. ");
    lcd.print("  ");
    lcd.print(":");
    if (visJuster) {
      if (minutter < 10) {
        lcd.print("0");
      }
      lcd.print(minutter);
    } else {
      lcd.print("  ");
    }
    lcd.print(":");
  }
}
```

Den fælles rutine justeringValg() er laves som følger, hvor den udnytter de variabler `lastPlus` og `lastMinus` som husker om der var trykket på en knap i sidste gennemløb, dels til at se om der er trykket på knapperne, og dels ved at sætte dem false, så funktionen justerTid() ser det som et nyt tastetryk.

Ved hjælp af justerCount ses det om der er lavet 5 langsomme tællinger, ellers tælles hurtigere ved at manipulere displayVisning.

```
// Hjælperutine der håndterer tiden i visningen af tiden og som aktiverer
tælling, når man holder knapperne nede
void justeringValg() {
  // Hvis en af knapperne er holdt nede
  if (lastPlus || lastMinus) {
    // I starten tælles langsomt op, senere hurtigt
    if (justerCount < 5) {
      displayVisning += 800;
    } else {
      displayVisning += 200;
    }
    justerCount++;
```

```
    // Signaler det som et nyt tastetryk
    if (lastPlus) {
      lastPlus = false;
    }
    if (lastMinus) {
      lastMinus = false;
    }
    // Vis tallet og nulstil timeout for justering
    visJuster = true;
    timeOut = 0;
```

Hvis der ikke er trykket på en af knapperne til justering, så bruges variablen visJuster til at få tallet til at blinke, så der er en indikation af at man er ved at indstille uret.

Der håndteres også at der timeoutes, hvis uret står mere end 20 sekunder i juster-modet.

```
  } else {
    // Hvis der ikke er nedtrykket en knap, så blinker tallet og der timeoutes
    visJuster = ! visJuster;
    displayVisning += 400;
    justerCount = 0;
    timeOut++;
    if (timeOut > 50) {
      selectMode = 0;
    }
  }
}
```

Test af TV-tiden og relæ

Hele formålet med konstruktionen er at kunne tænde og slukke for et TV, og kunne begrænse tiden det er tændt i løbet af et døgn. De test der er udført er beskrevet i tabel 4.

Testbeskrivelse	Forventet resultat	Res. midlertidig test	Resultat Sluttest
Der kan tændes og slukkes for TV'et	At relæet kan tænde og slukke	Det virker	Det virker
Tiden tæller ned mens TV'et er tændt	At det sker	Det virker	Det virker
Tiden tæller ikke ned mens TV'et er slukket	At rest tiden står stille	Det virker	Det virker
At nedtællingen stopper ved 0:00:00 samt at TV'et slukkes	At tiden stopper og TV'et slukkes	Det virker	Det virker med 0:01:40 som test-tid
At TV'et ikke kan tændes når resttiden står på 0:00:00	At relæet ikke påvirkes, når man forsøger at tænde TV'et	Det blokkerer	Det blokkerer fra tiden talt ned fra 0:01:40
At der tilskrives ny resttid når døgnet skifter til ur-visning 0:00:00	At der kommer den bestemte resttid	Det sker med 0:01:40	Det sker med 0:01:40
At resttidens timer, minutter og sekunder vises korrekt	At det vises som normal tid	Kun testet på de sidste 100 sekunder	Kun testet på de sidste 100 sekunder
Test af relæ-funktionen	LED'en tænder og relæet skifter	LED'en tændes / slukkes korrekt. Man kan høre relæet.	LED'en tændes / slukkes korrekt. Selve kontakten i relæet ikke testet.

Tabel 4. Test af TV-tiden og relæet

Hele testen er kun foretaget med en resttid på 100 sekunder (0:01:40), men da de vigtige ting baseres på en int-variabel, så vil det ikke gøre nogen forskel om den initialiseres med 100 eller 7200 (2 timer i sekunder). Det eneste der bør testes nøjere er visningen af resttiden, som dog formodes at blive vist korrekt.

Sluttest af prototype

Der er i tabellerne for testen under udvikling også noteret resultaterne af den sluttest der er foretaget, hvor hele testen er løbet igennem, for at verificere at udviklingen ikke har ødelagt funktioner der var testet tidligt i forløbet.

Som bemærket er der stadig enkelte ting der ikke testet ordenligt igennem, dette bør selvfølgelig afsluttes, men da det er relativt enkle ting der fejler eller ikke er testet til ende, så betragtes dette ikke som en alvorlig mangel ved projektet.

Til gengæld burde brugertesten gennemføres, da det kun er udvikleren der har testet. Dette kan give en del ændringsforslag, specielt omkring justeringen af uret, hvis det ikke opfører sig som forventet.

Konklusion

Umiddelbart virker produktet som det er specificeret i kravspecifikationen, men dog kun på prototype-niveau, som kan illustrere de forskellige funktioner der ønskes implementeret i produktet.

Der er stadig fejl i betjeningen, men de kan sikkert tilskrives hardwaren, fordi det er en prototype, og der er derfor ikke gjort yderligere for at rette fejlen.

Dokumentationen har nået et stort omfang, også fordi den dækker ind over forskellige fag som har hver deres dokumentations-tradition, men også fordi der på relevante steder i dokumentationen er argumenteret for hvorfor dokumentationen er lavet som den er. Dette vil normalt ikke relevant i en dokumentation.

Perspektivering

Projektet opfylder sit egentlige formål, at vise hvordan man kan dokumentere et projekt, hvor der indgår både hardware og software.

Som et egentlig produkt vil det ikke have nogen salgsværdi, da det er for begrænset i sin funktion. Projektet kan dog vise vejen frem mod en funktionalitet der kan indbygges i et TV.

Dette ville fordre en del flere funktionaliteter indbygget i produktet som fx

- Forældre login
- Administration af flere børns tid
- Tildeling af individuel tid til de enkelte børn
- Adgangskontrol for børn (login / iris-scanning)
- Mulighed for akkumulering af tid over flere døgn

Jo mere smarte TV bliver i dag, og har funktionaliteter på nettet kunne dette også være en del af bruger-interfacet (fx som administrationsdelen for forældrene).

Kildeliste

Arduino, 2017. *Arduino Refernce - millis.* [Online]
Available at: https://www.arduino.cc/reference/en/language/functions/time/millis/
[Senest hentet eller vist den 14 9 2018].

Arduino, 2018a. *Hello World.* [Online]
Available at: https://www.arduino.cc/en/Tutorial/HelloWorld
[Senest hentet eller vist den 25 9 2018].

Arduino, 2018b. *Liquid Crystal Display.* [Online]
Available at: https://www.arduino.cc/en/Tutorial/LiquidCrystalDisplay
[Senest hentet eller vist den 25 9 2018].

Arduino, H., 2018a. *HTX Arduino - Serial Monitor.* [Online]
Available at: http://www.htx-arduino.dk/index.php?title=Serial_Monitor
[Senest hentet eller vist den 14 09 2018].

Arduino, H., 2018b. *Tid og Samtidighed i software - millis().* [Online]
Available at: http://www.htx-arduino.dk/index.php?title=Tid_og_Samtidighed_i_Software#millis.28.29
[Senest hentet eller vist den 1 10 2018].

Arnoldsen, B., 2013a. *Adapter - Holstebro HTX Wiki.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Adapter
[Senest hentet eller vist den 19 9 2018].

Arnoldsen, B., 2013b. *LED - formodstand.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=LED#Beregning_af_formodstand
[Senest hentet eller vist den 12 09 2019].

Arnoldsen, B., 2013c. *Rotary Encoder - Holstebro HTX Wiki.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Rotary_Encoder
[Senest hentet eller vist den 12 09 2018].

Arnoldsen, B., 2013d. *Stepwise Improvement.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Stepwise_Improvement
[Senest hentet eller vist den 20 9 2018].

Arnoldsen, B., 2017a. *Arduino Display - Holstebro HTX Wiki.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Arduino_Display
[Senest hentet eller vist den 11 09 2018].

Arnoldsen, B., 2017b. *Arduino TFT Touch display.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Arduino_TFT_Touch_Display
[Senest hentet eller vist den 11 9 2018].

Arnoldsen, B., 2017c. *Relæ Modul.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Rel%C3%A6_modul
[Senest hentet eller vist den 12 09 2018].

Arnoldsen, B., 2018a. *Real Time Clock.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Real_Time_Clock
[Senest hentet eller vist den 14 09 2018].

Arnoldsen, B., 2018b. *Trykknap, forkant - Holstebro HTX Wiki.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Trykknap#Forkant
[Senest hentet eller vist den 14 9 2018].

Arnoldsen, B., 2018c. *Trykknap, Prel - Holstebro HTX Wiki.* [Online]
Available at: http://htx-elev.ucholstebro.dk/wiki/index.php?title=Trykknap#Prel
[Senest hentet eller vist den 4 9 2018].

Autodesk, 2018. *Eagle software - Autodesk.* [Online]
Available at: https://www.autodesk.com/products/eagle/overview
[Senest hentet eller vist den 17 9 2018].

Relay, S., u.d. *PDF med datablad til relæet.* [Online]
Available at: http://htx-elev.ucholstebro.dk/el/komponent/SRD-relay.pdf
[Senest hentet eller vist den 12 09 2018].

Source, O., 2018. *Fritzing - Electronics made Easy.* [Online]
Available at: http://fritzing.org/home/
[Senest hentet eller vist den 17 09 2018].

Wikipedia, 2018a. *Agil Systemudvikling.* [Online]
Available at: https://da.wikipedia.org/wiki/Agil_systemudvikling
[Senest hentet eller vist den 10 10 2018].

Wikipedia, 2018b. *Vandfaldsmodellen.* [Online]
Available at: https://da.wikipedia.org/wiki/Vandfaldsmodellen
[Senest hentet eller vist den 10 10 2018].

Bilag 1 – Programkoden

```
/*
 * Program-eksempel til et elektronisk ur
 * Udviklet i forbindelse med Arduinokursus 2018
 * Bent Arnoldsen
 */

// Variabler indeholdende tiden
byte sekunder = 55;
byte minutter = 59;
byte timer = 14;
unsigned long tidsRegistrering = 0;
boolean taelSekund = false;

// Variabler og konstanter der holder styr på tændt-tiden
boolean TV_ON = false;
const int doegnTid = 100;
int restTid = doegnTid;
boolean lastSkift = false;

// Konstanter der definerer input og output ben
const byte TV_OUT = 6;
const byte select = 7;
const byte justerMinus = 8;
const byte justerPlus = 9;
const byte skiftTV = 10;

// Variabler til justering af tiden og display-visning
unsigned long displayVisning = 0;
boolean visJuster = false;
boolean lastSelect = false;
byte selectMode = 0;
boolean lastPlus = false;
boolean lastMinus = false;
int timeOut = 0;
int justerCount = 0;

// Biblioteket til displayet
#include <LiquidCrystal.h>

// Displayets objekt lcd - med bendefinitioner
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // Seriel port til visningen i starten og til test senere
  Serial.begin(9600);
  // Start displayet - antal karaterer og linjer
  lcd.begin(16, 2);
  // Velkomst-besked.
  lcd.print("TV Timer UR");
  lcd.setCursor(0,1);
  lcd.print("Version 1.0");
  delay(2000);
  lcd.clear();
  // Sæt ind og udgange op til kontakter og relæet til TV'et
  pinMode(select, INPUT);
```

```
  pinMode(justerMinus, INPUT);
  pinMode(justerPlus, INPUT);
  pinMode(skiftTV, INPUT);
  pinMode(TV_OUT, OUTPUT);
}

// Hovedloopet der gennemløbes igen og igen efter setup()
void loop() {
  // De enkelte servicerutiner ar lavet til at returnere hurtigt, så der ikke er
noget der blokerer
  taelTid();
  udskrivSerial();
  justerTid();
  taendSluk();
  visDisplay();
}

void taelTid() {  // Funktion der håndterer tiden
  // Indiker at der ikke er gået et sekund
  taelSekund = false;
  // Fortsæt kun, hvis der er gået et sekund.
  if (millis() < tidsRegistrering) {
    return;
  }
  // Stop tidstælling, hvis der justeres på tiden.
  if (selectMode > 0) {
    tidsRegistrering = millis();
    return;
  }
  tidsRegistrering += 1000;  // sæt tidsregistreringen et sekund frem
  // Håndter tællingen af tid
  sekunder = sekunder + 1;
  if (sekunder == 60) {
    sekunder = 0;
    minutter = minutter + 1;
    if (minutter == 60) {
      minutter = 0;
      timer = timer + 1;
      if (timer == 24) {
        timer = 0;
        restTid = doegnTid;
      }
    }
  }
  // Signal til de andre rutiner om at der er gået et sekund
  taelSekund = true;
}

void udskrivSerial() {  // Funktion der udskriver tiden
  if (! taelSekund) {
    return;
  }
  // Udskriv tiden til Seriel-porten i formatet hh:mm:ss - minutter og sekunder
anvender foranstillede 0'er
  Serial.print(timer);
  Serial.print(":");
  if (minutter < 10) {
    Serial.print(0);
```

```
  }
  Serial.print(minutter);
  Serial.print(":");
  if (sekunder < 10) {
    Serial.print(0);
  }
  Serial.println(sekunder);
}

void justerTid() {  // Funktion der håndterer justering af minutter og timer
  // Vælg om der skal justeres - bestemmes af selectMode
  if (digitalRead(select)) {
    if (! lastSelect) {
      // Tæl justeringsmodet en frem og begræns til 0, 1 og 2
      selectMode++;
      selectMode %= 3;
      timeOut = 0;
      justerCount = 0;
      visJuster = false;
      delay(30);  // eleminer prel
    }
    lastSelect = true;
  } else {
    lastSelect = false;
  }
  // Håndter positiv justering af uret - afhængig af selectMode
  if (digitalRead(justerPlus)) {
    if (! lastPlus) {
      delay(30);  // eleminer prel
      if (selectMode == 1) {
        timer = timer + 1;
        if (timer == 24) {
          timer = 0;
        }
      }
      if (selectMode == 2) {
        minutter = minutter + 1;
        if (minutter == 60) {
          minutter = 0;
        }
      }
    }
    lastPlus = true;
  } else {
    lastPlus = false;
  }
  // Håndter negativ justering af uret - afhængig af selectMode
  if (digitalRead(justerMinus)) {
    if (! lastMinus) {
      delay(30);  // eleminer prel
      if (selectMode == 1) {
        if (timer > 0) {
          timer = timer - 1;
        } else {
          timer = 23;
        }
      }
      if (selectMode == 2) {
```

```
        if (minutter > 0) {
          minutter = minutter - 1;
        } else {
          minutter = 59;
        }
      }
    }
    lastMinus = true;
  } else {
    lastMinus = false;
  }
}

void taendSluk() {  // Funktion der håndterer set tid og tænd-sluk
  // Håndter tastetrykket (ønsket om at tænde/slukke)
  if (digitalRead(skiftTV)) {
    if (! lastSkift) {
      TV_ON = ! TV_ON;
    }
    lastSkift = true;
  } else {
    lastSkift = false;
  }
  // Sluk hvis tiden er gået, eller tæl ned hvis TV'et er tændt
  if (restTid == 0) {
    TV_ON = false;
  } else {
    if (TV_ON && taelSekund) {
      restTid--;
    }
  }
  // Sluk TV'et mens uret justeres (eleminerer snyd)
  if (selectMode > 0) {
    TV_ON = false;
  }
  // Sæt TV-udgangen til det der er fundet frem til
  digitalWrite(TV_OUT, TV_ON);
}

// Hjælperutine der håndterer tiden i visningen af tiden og som aktiverer
tælling, når man holder knapperne nede
void justeringValg() {
  // Hvis en af knapperne er holdt nede
  if (lastPlus || lastMinus) {
    // I starten tælles langsomt op, senere hurtigt
    if (justerCount < 5) {
      displayVisning += 800;
    } else {
      displayVisning += 200;
    }
    justerCount++;
    // Signaler det som et nyt tastetryk
    if (lastPlus) {
      lastPlus = false;
    }
    if (lastMinus) {
      lastMinus = false;
    }
```

```
    // Vis tallet og nulstil timeout for justering
    visJuster = true;
    timeOut = 0;
  } else {
    // Hvis der ikke er nedtrykket en knap, så blinker tallet og der timeoutes
    visJuster = ! visJuster;
    displayVisning += 400;
    justerCount = 0;
    timeOut++;
    if (timeOut > 50) {
      selectMode = 0;
    }
  }
}

void visDisplay() {  // Visning på et display
  // Opdater displayet efter det valgte interval (efter hvad visningen angiver)
  if (millis() < displayVisning) {
    return;
  }
  // Visningen af tiden og resttid
  if (selectMode == 0) {
    // Opdater 4 gange i sekundet - ellers kan displayet springe sekunder over
    displayVisning += 250;
    lcd.clear();
    // Print visningen af tiden i displayet - visning hh:mm:ss
    lcd.print("Klokken ");
    if (timer < 10) {
      lcd.print(" ");
    }
    lcd.print(timer);
    lcd.print(":");
    if (minutter < 10) {
      lcd.print("0");
    }
    lcd.print(minutter);
    lcd.print(":");
    if (sekunder < 10) {
      lcd.print("0");
    }
    lcd.print(sekunder);
    lcd.setCursor(0,1);
    lcd.print("Resttid ");
    // Konverter resttiden i antal sekunder til timer minutter og sekunder
tilbage
    byte restSec = restTid % 60;
    byte restMin = restTid / 60;
    byte restTim = restMin / 60;
    restMin = restMin % 60;
    // Print den resterende tid i displayet - visning hh:mm:ss
    if (restTim < 10) {
      lcd.print(" ");
    }
    lcd.print(restTim);
    lcd.print(":");
    if (restMin < 10) {
      lcd.print("0");
    }
```

```
    lcd.print(restMin);
    lcd.print(":");
    if (restSec < 10) {
      lcd.print("0");
    }
    lcd.print(restSec);
  }
  // Visning af kun timer hvis der justeres timer
  if (selectMode == 1) {
    justeringValg();
    lcd.clear();
    lcd.print("  Timer ");
    if (visJuster) {
      if (timer < 10) {
        lcd.print(" ");
      }
      lcd.print(timer);
    } else {
      lcd.print("  ");
    }
    lcd.print(":");
    lcd.print("  ");
    lcd.print(":");
  }
  // Visning af kun minutter hvis der justeres minutter
  if (selectMode == 2) {
    justeringValg();
    lcd.clear();
    lcd.print("   Min. ");
    lcd.print("  ");
    lcd.print(":");
    if (visJuster) {
      if (minutter < 10) {
        lcd.print("0");
      }
      lcd.print(minutter);
    } else {
      lcd.print("  ");
    }
    lcd.print(":");
  }
}
```

Bilag 2 – Totaldiagram