

Alarmduino

EKSAMENSPROJEKT I COMPUTER- & EL-TEKNIK

GLENN KJÆR HERPING, TOBIAS KAJ NIKOLAJ SØRENSEN &
CHRISTIAN HENRIKSEN

Abstract

In the final project of the subject *computer & electronics* our group has worked on the construction of an alarm system called *Alarmduino*. The following report is a comprehensive description of both the thoughts behind the system and its functions as well as the electronic construction and calculations.

Firstly, the report covers the thoughts behind the project and what is to be accomplished throughout the project. It also includes an acknowledgement of the final project e.g. how far we have come with the project and which sections had to be modified.

Under functions are detailed descriptions of how both the website and the Arduino Mega board controls the alarm system.

The electronic constructions cover both constructions of a keypad, a few pushbutton and most importantly a passive infrared sensor which has been built up by different components instead of a finished module.

Lastly there are judgements of what could be important to the system, what could be added to the system, also including a piece about what the law says about uploading portraits of others to the internet and a conclusion of the project.

To understand the report some knowledge of physics, mathematics, programming and electronics is compulsory.

ABSTRACT	1
INDLEDNING.....	5
FORVENTNINGER TIL ALARMDUINO.....	5
UDVIKLINGEN AF "ALARMDUINO"	6
<i>Kravspecifikationer</i>	<i>6</i>
<i>Udvikling af systemets funktioner (blok- og funktionsdiagrammer)</i>	<i>7</i>
<i>Foregående tidsplan</i>	<i>9</i>
NUVÆRENDE STATUS	9
ENDELIGE BLOKDIAGRAMMER.....	10
<i>Sikring af døre og vinduer.....</i>	<i>10</i>
<i>Sensor</i>	<i>11</i>
<i>Ringklokke</i>	<i>11</i>
ENDELIG TIDSPLAN.....	11
FYSISK UDFORMNING AF ALARMDUINO.....	13
SAMLET EL DIAGRAM.....	14
<i>Oversigt over ud- og indgang på Arduino Mega.....</i>	<i>15</i>
PROGRAMMERING I ARDUINO GENERELT	15
SIKRING AF DØRE OG VINDUER	16
EL DIAGRAM	17
FYSISK UDFORMNING AF PANEL	18
STATEMASKINE	18
STATE DIAGRAM.....	19
TESTS	19
MATRIX TASTATUR OG LCD	20
EL DIAGRAM	20
FYSISK UDFORMNING AF PANEL	21
BESKRIVELSE AF FUNKTIONEN FOR TASTATURET.....	22
BESKRIVELSE AF SWITCH STATMENT	22
FLOWCHAT FOR LCD.....	23
TESTS	24
<i>Logik analyse af I2C</i>	<i>24</i>
ETHERNET SHIELD.....	25
KLIENT.....	25
BESKRIVELSER AF FUNKTIONER	26
<i>Beskrivelse af PrepareAlarm funktionen.....</i>	<i>27</i>
Flowchart for PrepareAlarm funktionen	27
<i>Beskrivelse af ReadEthernetSite funktionen</i>	<i>27</i>
Flowchart for ReadEthernetSite funktionen	28
<i>Beskrivelse af AlarmStatusCheck funktionen.....</i>	<i>28</i>
Flowchart for AlarmStatusCheck funktionen	28
<i>Beskrivelse af UpdateAlarmStatus funktionen</i>	<i>29</i>
Flowchart for UpdateAlarmStatus funktionen	29
<i>Beskrivelse af UpdateLog funktionen</i>	<i>30</i>
Flowchart for UpdateLog funktionen	30
<i>Beskrivelse af UpdateFilename funktionen</i>	<i>30</i>
Flowchart for UpdateFilename funktionen	31
KOMPLIKATIONER.....	31
<i>Polling</i>	<i>31</i>
<i>Timeout.....</i>	<i>32</i>
TEST.....	33

GSM SHIELD	33
AT KOMMANDOER	33
FLOWCHART	34
KOMPLIKATIONER	35
TEST	36
RINGKLOKKE	36
KOMPLIKATIONER	38
PIR-BEVÆGELSESENSOR	39
SENSORKREDSLØB	39
<i>El diagram i Eagle</i>	40
BESKRIVELSE AF KREDSLØBETS FUNKTIONELLE DELE	40
<i>Sensor low-pass filter</i>	40
<i>Første bånd-pass filter med tilbagekobling</i>	43
<i>Anden bånd-pass filter</i>	45
<i>Comparator "sammenligner"</i>	46
<i>Målinger for ind- og udgangsspænding</i>	48
FYSISK UDFORMNING AF BEVÆGELSESENSOR	48
REGULERING AF FØLSOMHED FOR SENSOR	49
BESKRIVELSE AF FUNKTIONEN SENSOR_REG()	50
Flowchart for funktionen Sensor_reg()	51
CONNECTOR	52
EL DIAGRAM	53
KONTROLPANEL (HJEMMESIDE)	54
KRAV TIL LOGINSYSTEMET	54
STRUKTUR PÅ SIDERNE	54
DATABASE	55
ALARM_STATUS	55
LOG	56
INDEX	56
OPRET BRUGER	57
KONTROLPANEL	59
KRYPTERING	60
<i>Forklaring på blowfish algoritmen</i>	61
Eksempel på XOR	62
REDIGER BRUGER	62
DATABASE CONNECT	63
LOG UD	64
SAMMENSPIL MED ARDUINO	64
<i>Update log</i>	65
<i>Read Alarm</i>	65
<i>Alarm log</i>	67
<i>Camera log</i>	68
PERSONDATALOVEN	69
VIDEREUDVIKLING	70
VIDEREUDVIKLING AF HJEMMESIDE	71
KONKLUSION	72
LITTERATURLISTE	73
FIGURLISTE	76

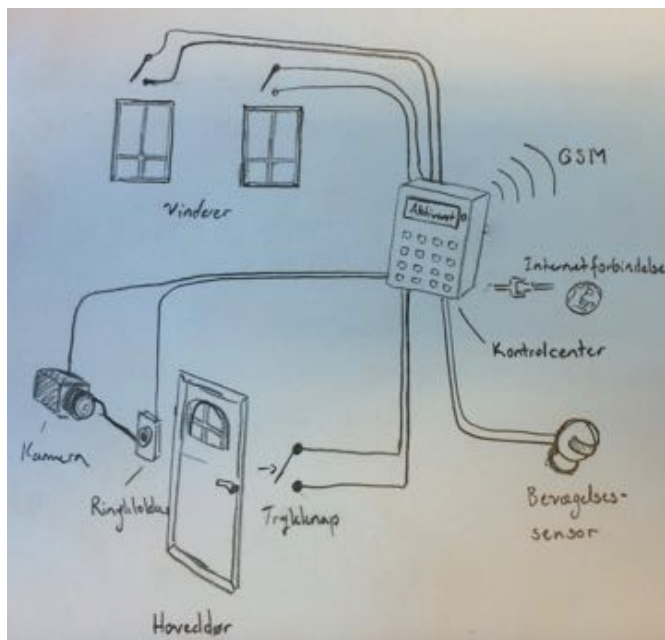
TABELLISTE	78
BILAG:	79
BILAG 1: KODE CONTROLPANEL.INO	79
BILAG 2: KODE DISPLAY.H	81
BILAG 3: KODE DOOR_AND_WINDOW.H	82
BILAG 4: KODE DOORBELL.H	84
BILAG 5: KODE ETHERNETSHIELD.H	87
BILAG 6: KODE GSM.H	90
BILAG 7: KODE MATRIX.H	91
BILAG 8: KODE SENSOR.H	92
BILAG 9: INDEX.PHP	93
BILAG 10: DATABASECONNECT.PHP	94
BILAG 11: STYLE.CSS	94
BILAG 12: LOGIN.PHP	96
BILAG 13: REDIGER BRUGER.PHP	98
BILAG 14: OPRET BRUGER.PHP	100
BILAG 15: CAMERA_LOG.PHP	102
BILAG 16: UPDATE_LOG.PHP	103
BILAG 17: ALARM_LOG.PHP	104
BILAG 18: READ_ALARM.PHP	107
BILAG 19: CHANGENAMEFTP.PHP	108
BILAG 20: LOGUD.PHP	109
BILAG 21: GODKENDTE PROJEKTBEKRIVELSE	110

Indledning

I forbindelse med dette eksamensprojekt valgte vi temaet "velfærds-elektronik i hjemmet", som omfatter udvikling og fremstilling af et stykke velfærds-elektronik. Produktet skulle derfor løse en opgave med henblik på en bestemt målgruppe. Vi valgte derfor at arbejde med fremstillingen af et intelligent alarmsystem, der ved hjælp af elektriske kredsløb, et program og et internet- og mobilintegreret system kan sikre det normale hjem. Alarmsystemet skulle være brugervenligt og anvendeligt fra apparater med internetadgang, således at det kunne styres fra et webbaseret "kontrolcenter". Med andre ord skulle der laves en hjemmeside med et bruger-interface, hvorfra styring af systemet var muligt, vha. et loginsystem. På den måde skulle det være muligt at holde sig opdateret på huset, når man for eksempel er på arbejde eller ferie. Det var derfor også meningen, at alarmsystemet skulle kunne give besked og opdatere om eventuelle "besøgende".

Forventninger til Alarmduino

Systemet, som vi senere navngav "Alarmduino", skulle opstilles således, at det kunne beskytte 1 hoveddør, 2 vinduer og et område med en bevægelsessensor. Dørene og vinduerne skulle aktiveres vha. trykknapper, hvor der herudover skulle være en ekstra trykknop som dørklokke. Meningen med dørklokken var, at der ved uventet besøg, når alarmsystemet var tændt, ville være mulighed for at se hvem der ringer på. Dette skulle ske vha. et overvågningskamera ved døren. Der ville altså blive sendt et tidsstempelt billede til serveren og en besked til brugeren, når alarmen var aktiveret, så man derefter kunne logge ind på hjemmesiden og se, hvem der ringede på døren. Sensoren, som skulle beskytte et område, skulle registrere bevægelser, når alarmen var aktiveret.



Figur 1: Oversigt over bagtanke for Alarmduino-systemet

Hele bagtanken beskrevet ovenfor, blev skitseret som det kan ses på figur 1.

I punktform var forventningerne til "Alarmduino":

- Sikring af døre og vinduer

- Bevægelsessensor (PIR-sensor)
- GSM (mobilnetværk)
- Internetforbindelse
- Ringklokke med kamera

Udviklingen af "Alarmduino"

Det er væsentligt at se på selve udviklingen af system, altså projektet udformning fra start til slut. Ud fra de ovenstående bagtanker startede man med at gøre rede for kravspecifikationerne, hvorefter man kortlagde systemets funktioner i blokdiagrammer.

Kravspecifikationer¹

Ud fra bagtankerne kunne man begynde at opstillet krav til, hvad Alarmduinoen skulle kunne, således man på samme tid fik man mere præcise retningslinjer at arbejde efter:

- Kontrolcenter, være tilgængeligt hvor end man er, så længe der er internet. Herunder selve kontrolpanelet, hvor Arduinoen sidder og styrer resten.
 - Login på hjemmesiden påkrævet for at styre alarmer og se en log over tidligere aktiveret alarmer.
 - Ved oprettelse af en ny bruger skal en two-steps-verification bruges. Eventuelt hvor man opretter brugeren og derefter skal man indtaste en unik kode (tilfældig kode der generes hver gang en bruger oprettes) på selve alarmer. For at oprette en ny bruger skal man først være logget ind.
 - Man skal have mulighed for at tænde og slukke for alarmer på kontrolpanelet vha. en kode man har fra kontrolcenteret.
 - Der skal være et display, hvor man kan se diverse statusser.
 - Der skal eventuelt være en batteribackup på kontrolpanelet.
- Ringklokke
 - Når der trykkes på ringklokken skal der tages et billede af personen, der står udenfor.
 - Billedet uploades til kontrolcenteret og der tilføjes et tidsstempel.
 - Skal virke uanset om alarmer er aktiveret eller ej.
- Sikring af døre og vinduer

¹ Direkte uddrag fra projektbeskrivelsen (afleveret d. 11/3-16) – Læser henvises til bilag 21.

- En kontakt der bliver aktiveret hvis en dør/vindue åbnes mens alarmen er tændt. Såfremt alarmen ikke er tændt skal der selvfølgelig ikke ske noget.
- Uploade en log med tidspunktet og hvilken dør/vindue der blev aktiveret til kontrolcenteret, hvis det sker når alarmen er aktiveret.
- Sende en SMS til administratoren om det mulige indbrud.
- Sensor
 - En sensor skal, når alarmen er aktiveret, registrere bevægelser.
 - Uploade en log med tidspunktet til kontrolcenteret.
 - Sende en SMS til administratoren om det mulige indbrud.

Udvikling af systemets funktioner (blok- og funktionsdiagrammer)²

Herunder er skitseret de forskellige funktions- og blokdiagrammer for de tre detekteringstyper, som er hhv. døre og vinduer, ringklokke og bevægelsessensor.

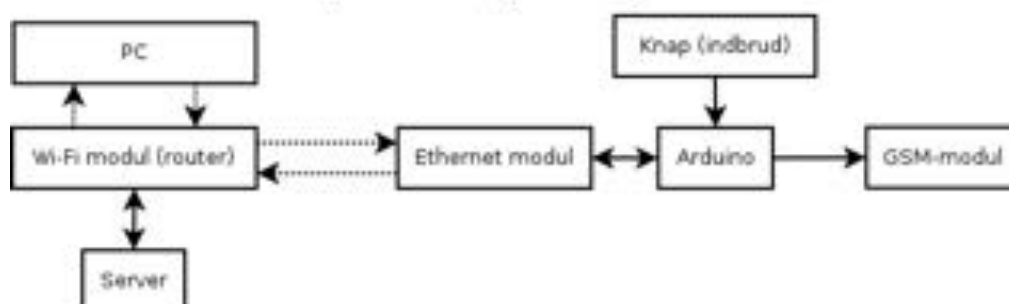
Funktionsdiagram - Sikring af døre og vinduer



Figur 2: Foregående funktionsdiagram for sikring af dør og vinduer

Sikringen af døre og vinduer sker i Arduinoens software. Er alarmen aktiveret, holdes der øje med om "låsen", repræsenteret af en trykknop, er brudt. Hvis den brydes vil alarmeringen ske og serveren vil få besked samtidigt med at der udsendes en SMS til både bruger og alarmcentral.

Blokdiagram - Sikring af døre og vinduer

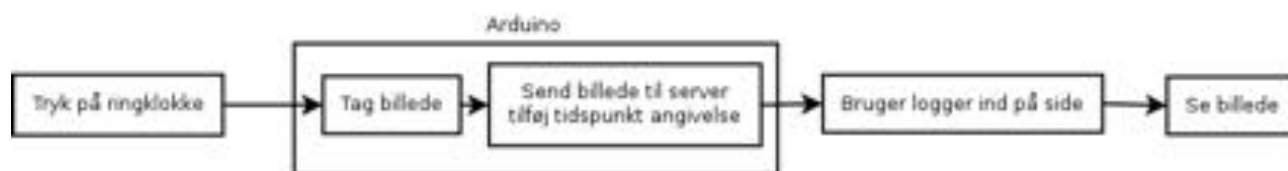


Figur 3: Foregående blokdiagram for sikring af dør og vinduer

² Direkte uddrag fra projektbeskrivelsen (afleveret d. 11/3-16) – Læser henvises til bilag 21.

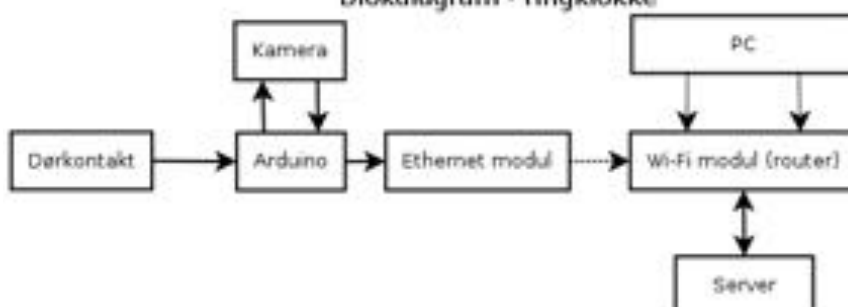
Ringklokken vil være forbundet med Arduionen, så der ved brug af denne, vil sætte gang i softwaren, som tager et billede og sender til serveren med tidsangivelse. Brugeren vil herefter få mulighed for at se billedet på hjemmesiden.

Funktionsdiagram - ringklokke



Figur 4: Foregående funktionsdiagram for ringklokke

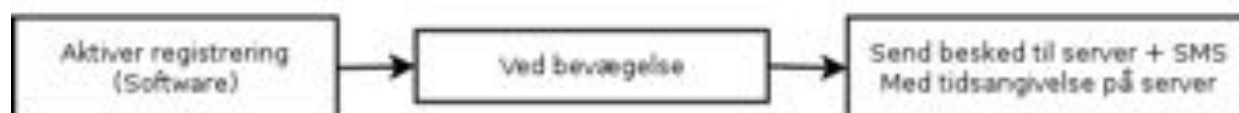
Blokdiagram - ringklokke



Figur 5: Foregående blokdiagram for ringklokke

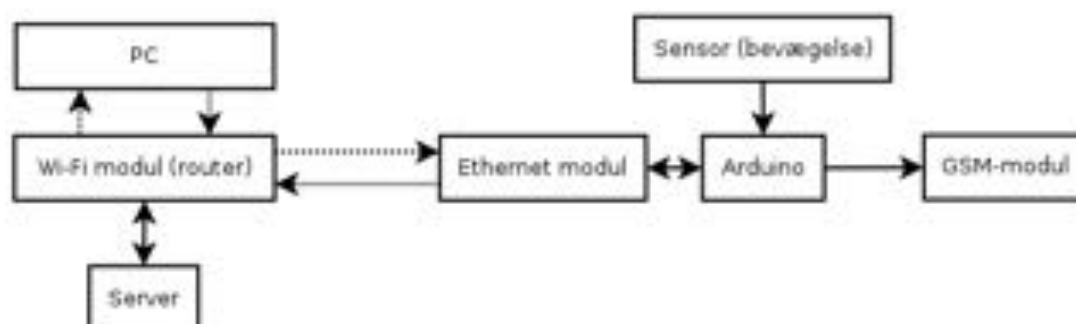
Sensoren virker efter samme princip som trykknapperne ved døren og vinduerne. Her er det blot en bevægelsesfølsom sensor, der aktivere alarmen, som så sender SMS og data til serveren.

Funktionsdiagram - Registrer bevægelse



Figur 6: Foregående funktionsdiagram for bevægelsessensor

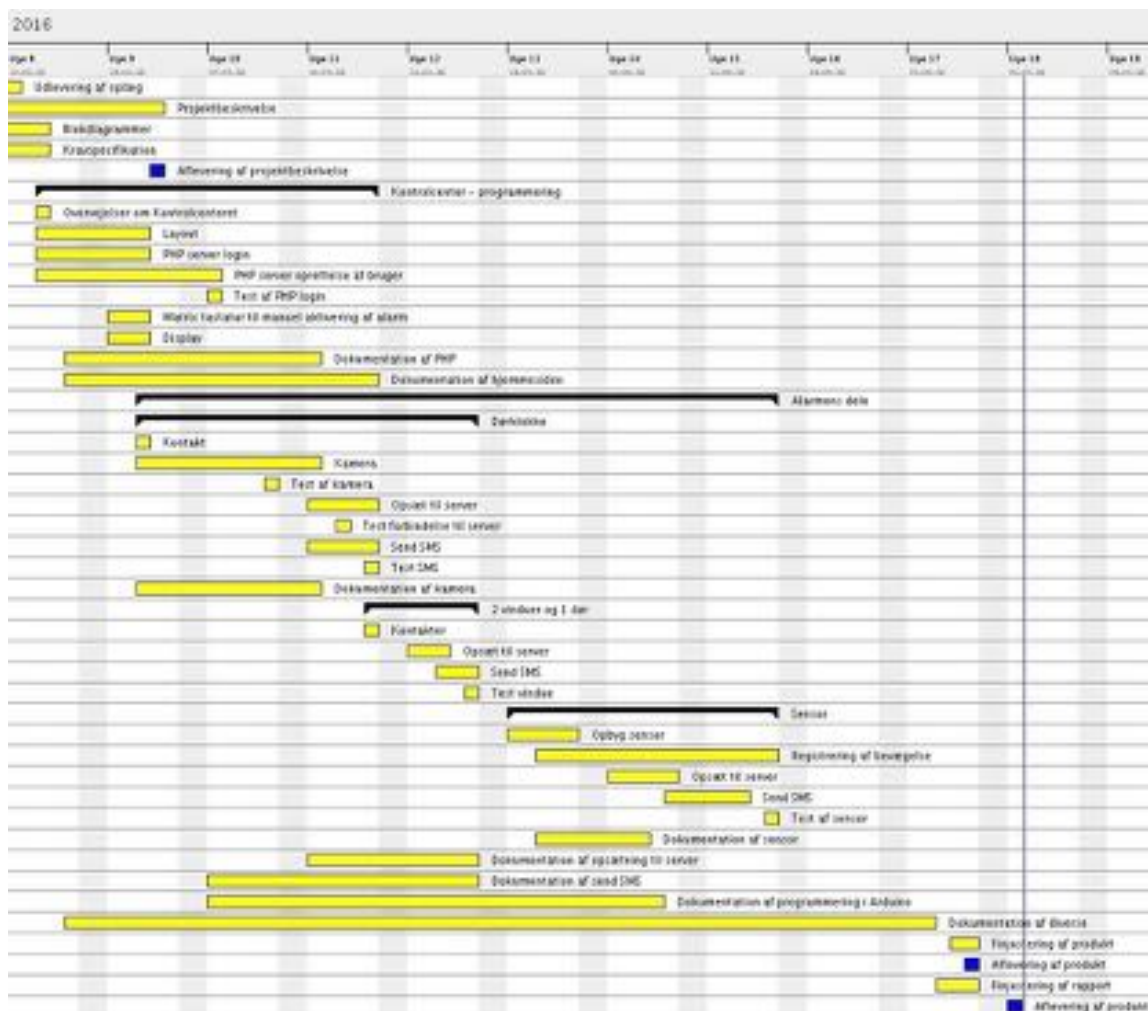
Blokdiagram - Registrer bevægelse



Figur 7: Foregående blokdiagram for bevægelsessensor

Foregående tidsplan

Før projektets start blev der i projektbeskrivelsen, dannet et overblik over, hvad der skulle laves og hvornår. Dette blev gjort gennem et Gantt-diagram, som kan ses på figur 8.



Figur 8: Oprindeligt Gantt-diagram

Nuværende status

I sidste ende har man formået at konstruere stort set alle de planlagte dele af systemet. Den eneste betydelige undtagelse er ideen om det skjulte kamera, som skulle tage billeder af folk der bruger ringklokken. Man undlod denne del af en enkel årsag, at det ville blive for tidskrævende at sætte sig ind i opsætningen af sådan et kamera og billedbehandling. Blot forsøget på at forstå, hvordan et kamera kan opsættes gennem Arduino, tog over 2 ugers kostbar arbejdstid fra projektforløbet. Men som det også bliver forklaret senere i rapporten, blev alternativet at der bliver uploadet et billede fra en database, i stedet for et faktisk kamera der tager billederne. En kort opsummering af, hvad man er kommet frem til gennem projektet:

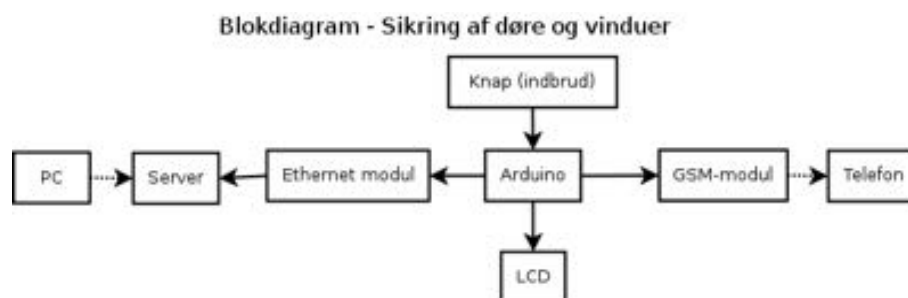
- Interfacet er brugervenligt
- Alarmduinoen er sikret vha. blowfish-algoritme og salt-værdier, så udefrakommende ikke umiddelbart kan tilgå alarmerne.
- Opsætning af fire sensorer: 3 kontakter til dør og vinduer samt en PIR-sensor
- Opkobling til LAN-netværk (Local Area Network)
- Kan sende SMS til admin og bruger, når en sensor er brudt.
- Hjemmeside med historik-log over brud på sensorer.
- Oversigt over huset med placering af sensorer.
- Uploading af billede fra database, når der ringes på døren.
- Redigering af bruger-kartoteket. Både opretning og sletning af brugere.
- Mulighed for at styre alarmerne via internet, selvom man ikke er hjemme.

Endelige blokdiagrammer

Alarmduinoen er hovedsageligt opbygget i tre dele: Sikring af døre og vinduer, ringklokke og sensor. For disse medfølger der blokdiagrammer, der beskriver de forskellige blokke, som har en kort beskrivelse i de efterfølgende afsnit. På blokdiagrammerne fungerer de stiplede pile som trådløse forbindelser.

Sikring af døre og vinduer

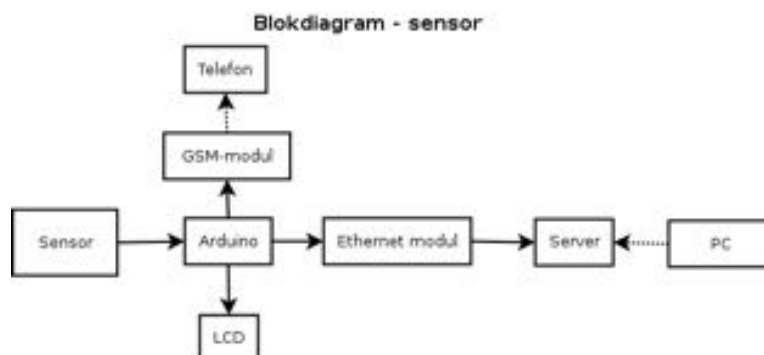
Dette modul har hovedformålet, at hvis der bliver aktiveret en af knapperne ved døren eller vinduerne sendes der en besked til Arduinoen, som da videresender information til GSM-modul, LCD og ethernet modul. Ethernet modulet skriver da gennem en router til serveren, som der kan tilgås fra en computer, hvor man kan se information omkring systemet.



Figur 9: Blokdiagram for sikring af dør og vinduer

Sensor

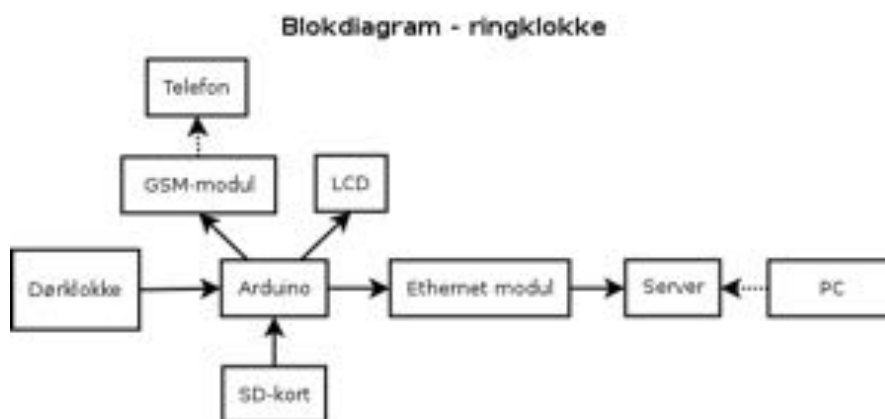
Sensorens blok diagram er bygget op stort set magen til sikring af dør og vinduer, og det skyldes da også, at der sker næsten samme. Forskellen er, at der her skal være en registreret bevægelse i stedet for en aktivering af knap.



Figur 10: Blokdiagram for sensor

Ringklokke

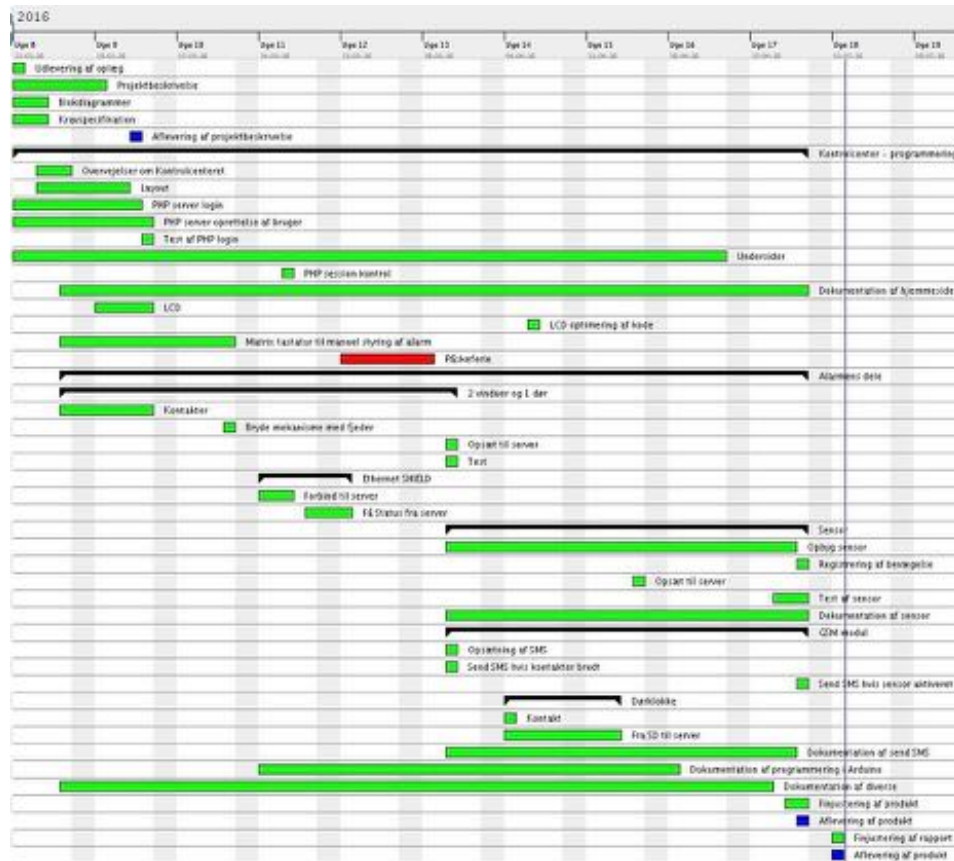
For ringklokken, der tidligere blev beskrevet, til at skulle tage et billede når nogen ringede på døren er der foretaget nogle ændringer. Det at få et kamera koblet op og tage et billede viste sig, at være mere kompliceret end antaget, og det blev derfor besluttet, at bruge et eksisterende billede, lagre det på et SD kort og så uploade derfra. Ud over dette er ringklokken også meget lig sikring af døre og vinduer samt sensoren. Der sker en hændelse og der bliver sendt information med SMS og til serveren.



Figur 11: Blokdiagram for ringklokke

Endelig tidsplan

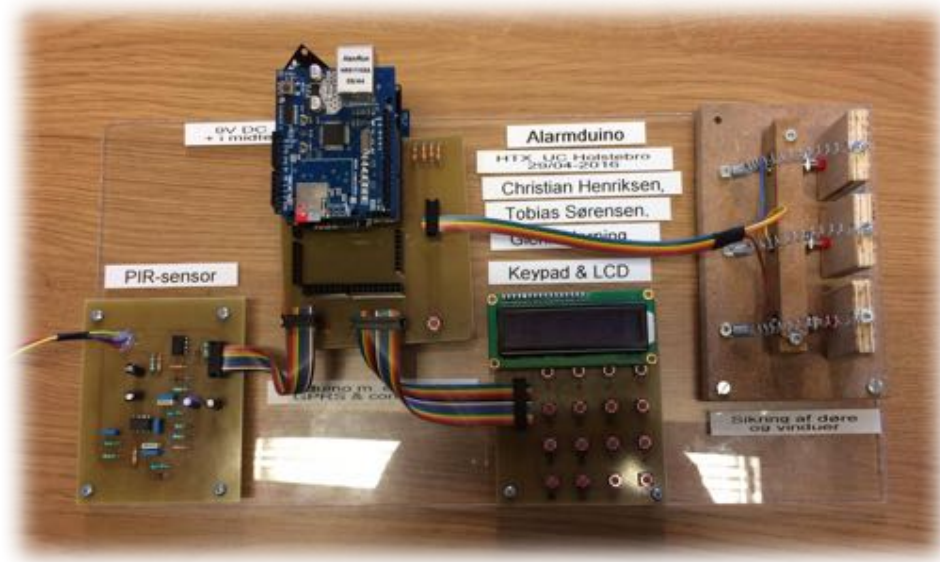
Igennem projektets forløb er denne tidsplan dog blevet revideret nogle gange, for at få et mere realistisk blik på, hvor i forløbet vi var kommet til og hvad der stadig manglede. Det har resulteret i nogle afvigelser i forhold til den oprindelige tidsplan. Den reelle tidsplan kan ses på figur 12.



Figur 12: Det reviderede Gantt-diagram

Fysisk udformning af Alarmduino

Den fysiske udformning af Arduinoen er på et stykke plexiglas, hvormed man også kan se printenes bagside. Derudover er de forskellige moduler hovedsagligt bygget på print, som alle går til connector shieldet med fladkabel. Dette gøres for at styre ledningerne. Oven på connector shieldet sidder der desuden et GSM shield og et ethernet shield.

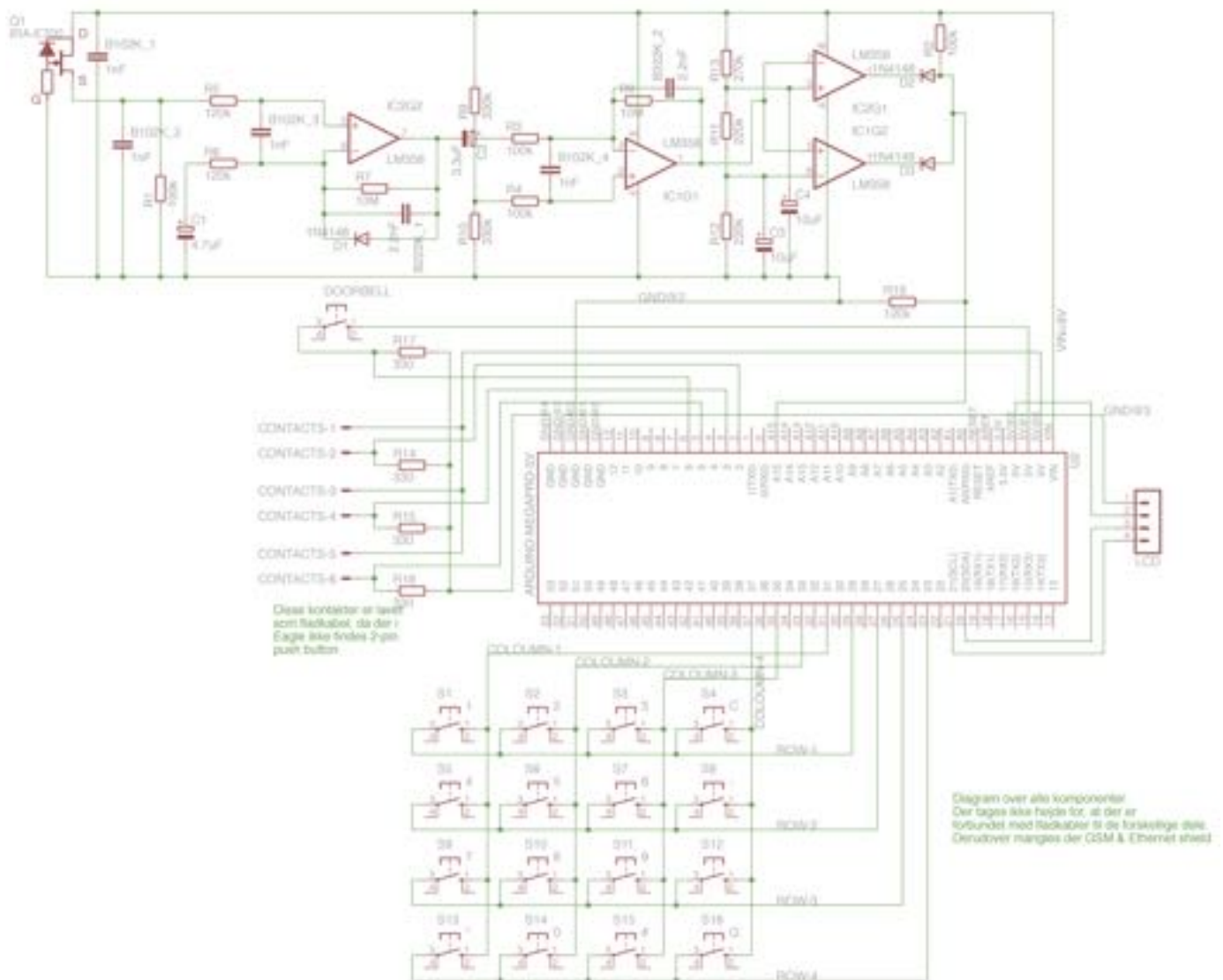


Figur 13: Fysisk udformning af Alarmduino

Samlet el diagram

På figur 14 ses det endelige samlede el diagram for hele systemet. El diagrammet er opdelt i fem dele: sensoren, placeret øverst, kontakterne til sikring af døre og vinduer samt dørklokken, placeret i midten til venstre, tastaturet nederst, Arduino Mega i midten og LCD'et til højre. På el diagrammet mangler der de to shields ethernet og GSM, da de ikke findes som komponenter i Eagle programmet.

El diagrammerne i denne rapport er desuden alle opbygget således, at alle inputs er i venstre side og outputs i højre side.



Figur 14: Samlet el diagram for Alarmduino

Oversigt over ud- og indgang på Arduino Mega

For nemmere at kunne hurtigt at kunne holde styr på hvilke ud- og indgange de forskellige moduler har på Arduinoen har et Excel ark været brugt som reference. Det er hovedsageligt gjort af den grund, at man på den måde hurtigt kan se om en ny tildeling af en pin kan tillades. For eksempel var pin 4 oprindeligt tildelt vindue 2 (TriggerWindow2), men da SD-kortet blev koblet sammen med, var det pludseligt ikke længere muligt, da systemet ikke længere fungerede, og det var da hurtigt at finde hvor fejlen lå i, nemlig at SD skal bruge pin 4. Tabel 1 viser dette Excel ark. Arket kan desuden bruges, hvis der skal tilføjes moduler, hvormed man kan se, hvor dette kan lade sig gøre.

Liste over indgange optaget på Arduino								
Digitale:	Hvad	Pin på shield	Digitale:	Hvad	Pin på shield	Analoge:	Hvad	Pin på shield
0 / TX0			27	ROW 2	KEYBOARD 7	0 / TX0		
1 / RX0			28			1 / RX0		
2	TriggerDoor	CONTACTS 2 + R14	29	ROW 1	KEYBOARD 9	2		
3	TriggerWindow1	CONTACTS 4 + R15	30			3		
4	SD		31	COLOUMN 1	KEYBOARD 11	4		
5	TriggerWindow2	CONTACTS 6 + R16	32			5		
6	DoorbellContact	DOORBELL 3 + R17	33	COLOUMN 2	KEYBOARD 13	6		
7	GSM TX		34			7		
8	GSM RX		35	COLOUMN 3	KEYBOARD 15	8		
9	GSM power on		36			9		
10	ETHERNET + SD		37	COLOUMN 4	KEYBOARD 16	10		
11	ETHERNET		38			11		
12	ETHERNET		39			12		
13	ETHERNET		40			13		
14 / TX3			41			14		
15 / RX3			42			15	Sensor udgangsværdi	SENSOR 7
16 / TX2			43			Diverse:		
17 / RX2			44			5V @ 0	5V	CONTACTS 1+3+5 & DOORBELL 1
18 / TX1			45			5V @ 1		
19 / RX1			46			5V @ 2	5V	KEYBOARD 1
20 / SDA	LCD SDA - purple	KEYBOARD 4	47			3.3V		
21 / SCL	LCD SCL - green	KEYBOARD 2	48			GND @ 0		
22			49			GND @ 1		
23	ROW 4	KEYBOARD 3	50	ETHERNET		GND @ 2	GND	SENSOR 1
24			51	ETHERNET		GND @ 3	GND	KEYBOARD 14
25	ROW 3	KEYBOARD 5	52	ETHERNET		GND @ 4		
26			53	ETHERNET (SS) - skal forblive output		AREF		
						RESET		
						VIN	9V	SENSOR 9

Tabel 1: Liste over ind- og udgange på Arduino

Programmering i Arduino generelt

Det fysiske produkt, der består af kontakter, tastatur, LCD, ringklokke, sensor, GSM og ethernet er programmeret i Arduino, hvor der flere dokumenter – et for hvert modul, som alle bliver kørt igennem hovedfilen Controlpanel.ino. Således er filerne opdelt til de forskellige moduler som ses på tabel 2.

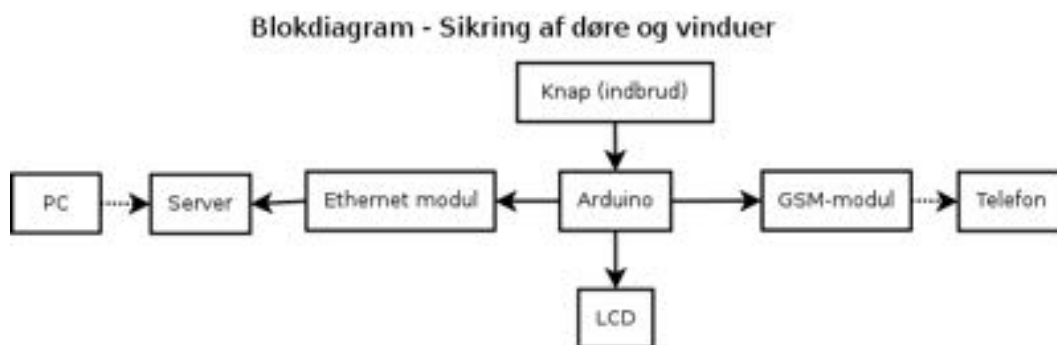
Filnavn	Modul
Controlpanel.ino	Samlet (med void setup & loop)
Display.h	LCD
Door_and_window.h	Sikring af døre og vinduer
Doorbell.h	Ringklokke
EthernetSHIELD.h	Ethernet shield
GSM.h	GSM shield
Matrix.h	Matrix tastatur
PasswordFile.h	Fil med kodeord til FTP
Sensor.h	Sensor

Tabel 2: Liste over filer og hvilket modul de tilhører

Sikring af døre og vinduer

Sikringen af døre og vinduer har det formål, at kunne registrere, hvorvidt der forekommer et indbrud, ved at enten et vindue eller døren bliver åbnet. Sker der en sådan hændelse, vil programmet udføre en dertil beregnet handling. Koden er bygget op i en statemaskine, der gør det meget overskueligt at vide hvilke handlinger der skal udføres når programmet er i den tilstand.

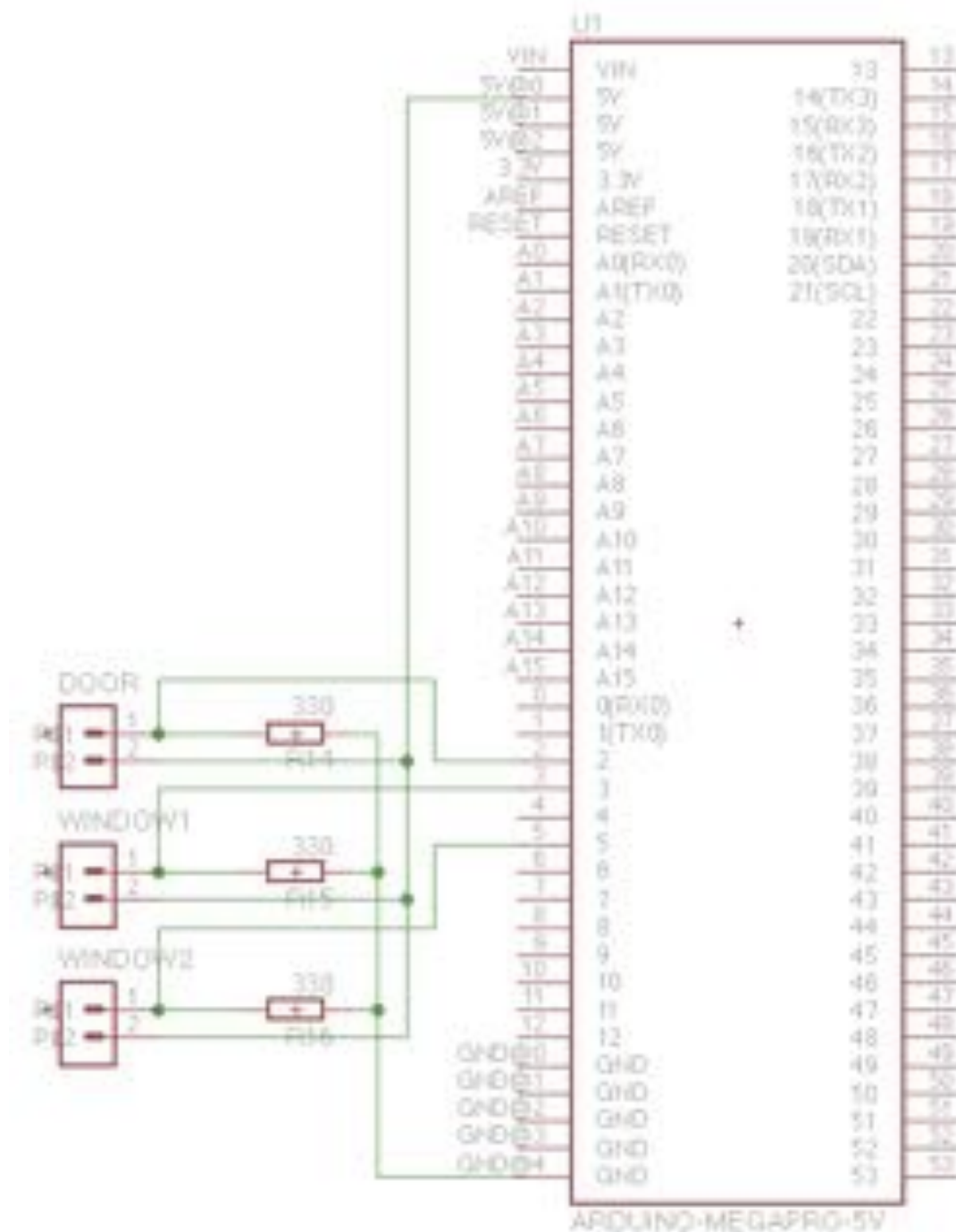
Overordnet set ser blokdiagrammet for sikring af døre og vinduer ud som på figur 15. På blokdiagrammet kan det ses, at der ved tryk på en knap, fx døren, bliver sendt en besked til Arduinoen, der sender beskeder flere steder hen. Både til et GSM-modul, der skriver en SMS, til en server, gennem et ethernet modul og til et display.



Figur 15: Blokdiagram for sikring af døre og vinduer

El diagram

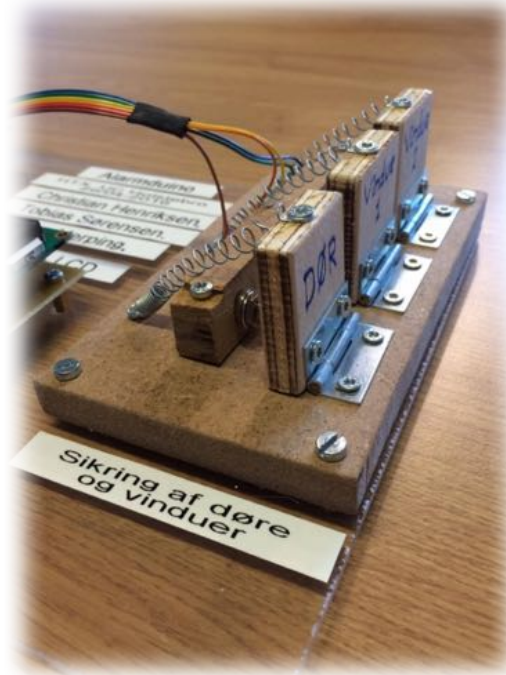
På figur 16 ses el diagrammet for sikring af døre og vinduer. På diagrammet ses, hvor de tre kontakter er koblet på Arduino Mega. Det kan desuden ses, hvordan de tre kontakter er koblet op med pull-down modstande. Dette forhindrer kontakterne i at svæve. Dette beskriver dog kun kontakterne, hvor der i den senere beskrevet kode også indgår en sensor. Denne sensor er beskrevet i et afsnit for sig selv.



Figur 16: El diagram til sikring af døre og vinduer

Fysisk udformning af panel

På figur 17 ses den fysiske udformning af kontakterne. Det kan her ses, hvordan de altid vil være trykket ind, hvormed går er høj / 1. At udforme kontakterne på denne måde skyldes, at de skal simulere, at døren eller vinduerne er lukket og dermed presser kontakterne ind, ligesom de ville gøre det i et rigtigt hus. Når man trækker i en af pladerne så kontakterne lav / 0, og hvis statemaskinen da befinder sig i den rigtige tilstand, hvor alarmen er tændt (state 2) sker der de dertilhørende handlinger.



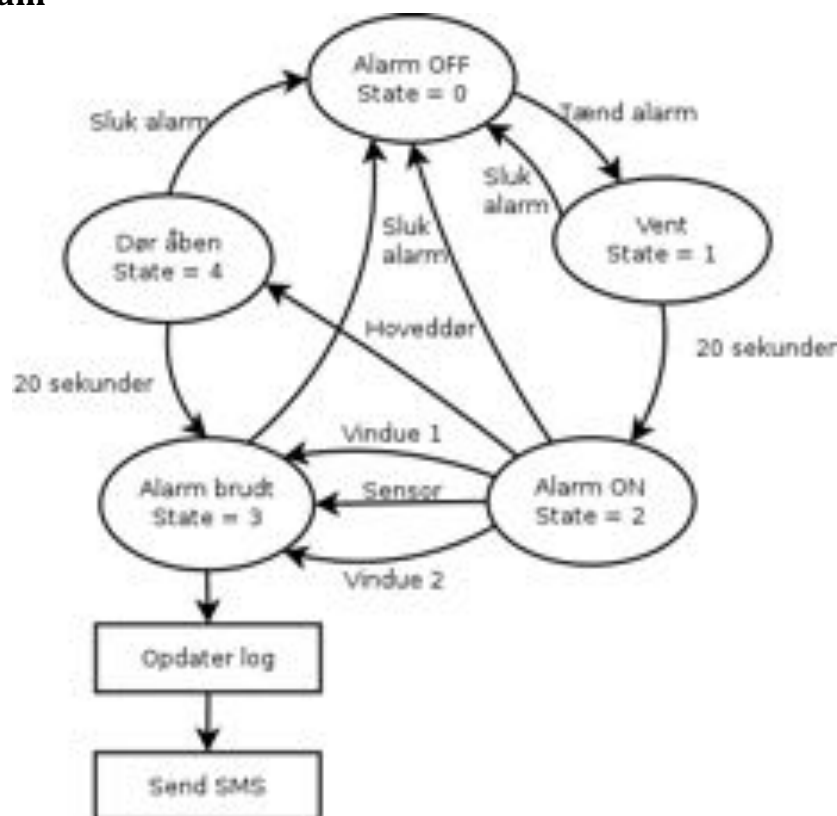
Figur 17: Fysisk udformning af sikring af døre og vinduer

Statemaskine

På figur 18 ses statediagrammet for sikring af døre og vinduer. Der er i alt 5 states, som statemaskinen kan være aktive i, dog kun en ad gangen. I state 0 (Alarm OFF) er alarmen slukket. Herfra kan alarmen kun blive aktiveret, hvormed den ændrer state til 1 (vent). I denne state venter alarmen 20 sekunder, så man har mulighed for at komme ud af døren når man forlader sit hjem. Der er dermed samtidigt taget højde for den mulighed, at man ved en fejltagelse har aktiveret alarmen, og da har man mulighed for at slukke alarmen igen. Dernæst skifter maskinen automatisk til state 2 (Alarm ON). Herfra kan der ske flere hændelser. Alarmen kan enten slukkes, et vindue bliver åbnet, sensoren registrer bevægelse eller døren åbnes. Slukkes alarmen skiftes der selvfølgelig tilbage til state 0. Såfremt enten et vindue eller sensoren bliver aktiveret, skiftes der til state 3 (Alarm brudt). Er det derimod døren der bliver åbnet / brudt skiftes der til state 3 (Dør åben). Dette sker selvfølgelig pga. den funktion, at man skal have mulighed for at komme ind og slukke alarmen når man kommer hjem. Derfor ventes der 20 sekunder før der skiftes til state 3. Slukker man for alarmen inden for disse 20 sekunder skiftes der selvfølgelig til state 0. I state 3 har alarmen registret, at der er blevet brud ind og alarmen sender en SMS til admin brugeren og opdaterer loggen på hjemmesiden, så man også senere kan vide hvor og hvornår det skete.

Programmeringen af statemaskinen kan ses i bilag 3 under funktionen Door_and_windows().

State diagram



Figur 18: Statediagram for dør, sensor og vinduer

Tests

Der er udformet nogle tests for statemaskinen. Disse tests kan ses i tabel 3, hvor der er beskrevet tilstanden, hvad der skal ske og hvorvidt det faktisk sker.

Tilstand	Hvad skal der ske?	Succes?
State = 0, alarm bliver tændt	Alarmen skifter til state 1, venter 20 sekunder, hvorefter alarmen skifter til state 2.	Ja
State = 1, alarmen bliver slukket	Alarmen skifter til state 0	Ja
State = 2, alarmen bliver slukket	Alarmen skifter til state 0	Ja
State = 2, vindue1- eller vindue2-kontakten går lav, eller sensoren bliver aktiveret	Alarmen skifter til state 3	Ja
State = 2, dørkontakten bliver deaktiveret	Alarmen skifter til state 4, venter 20 sekunder, og skifter til state 3.	Ja
State 4, alarmen slukkes	Alarmen skifter til state 0	Ja
State 3	Opdater log samt send SMS	Ja
State 3, alarmen slukkes	Alarmen skifter til state 0	Ja

Tabel 3: Tabel over tests til dør, sensor og vinduer

Matrix tastatur og LCD

De efterfølgende afsnit beskriver tastaturet og LCD'et, som er det sted i huset, hvor man kan koble alarmen til eller fra, se hvad Alarmduinoen på nuværende tidspunkt laver, fx uploade et billede hvis der er trykket på ringklokken. På figur 19 kan det ses hvilken karakter de forskellige knapper har fået tildelt sig. Det bemærkes måske, at der ikke er nogen pull-up modstande på kontakterne. Dette skyldes, at der i biblioteket "Keypad", som benyttes, er kodet til interne pull-up modstande, som Arduinoen kan levere.



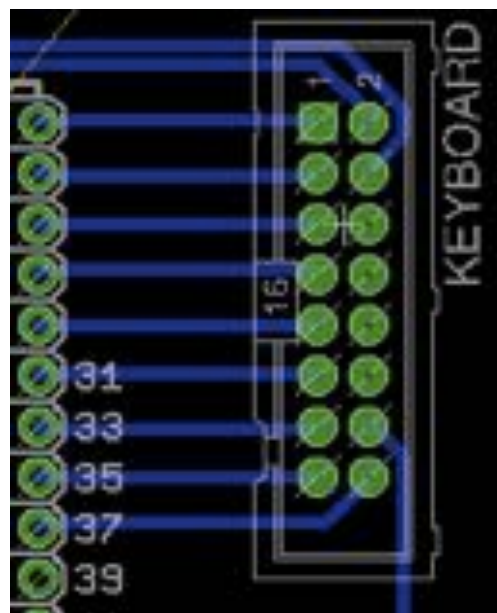
Figur 19: Udformning af tastatur & LCD

El diagram

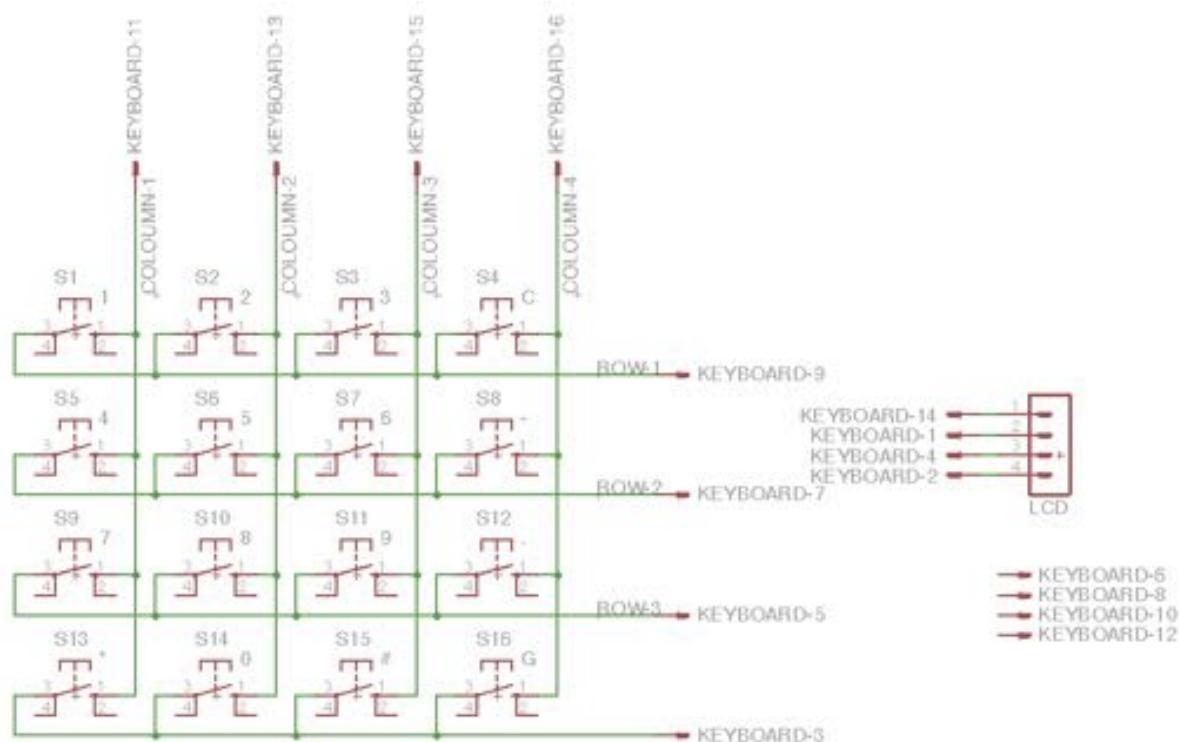
På figur 21 ses el diagrammet for matrix tastaturet samt LCD. For et LCD er der normalt mange flere pins der skal kobles op, men ved brug af en I2C bus skal der kun bruges fire pins: 5V, GND, SDA og SCL. SDA er den serielle forbindelse, hvor der sendes data, mens SCL er den serielle forbindelse der bestemmer hvornår data sendes.

Det bemærkes måske, at rækker og kolonner ikke hele tiden forbindes med den næste pin. Det skyldes, at det skal være nemt på connector shieldet, hvor det er svært at få plads på begge sider, og da få forbindelse til fladkabel. Derfor følger de ordenen der ses på figur 20, så det passer med connector shieldet.

De fire pins i højre hjørne på el diagrammet er de overskydende pins fra fladkablet.



Figur 20: Tilkobling af fladkabel til tastatur og LCD



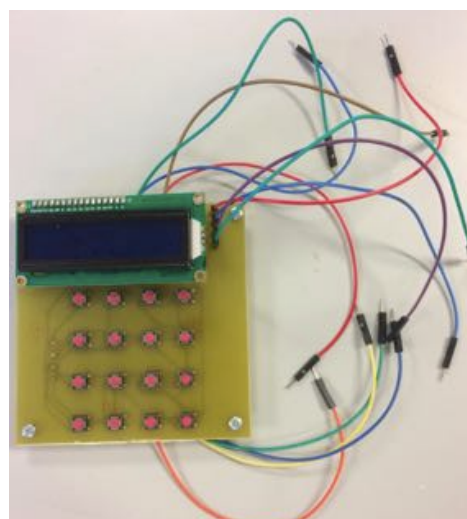
Figur 21: El diagram for tastatur

Fysisk udformning af panel

Da de færdiglavet matrix tastaturer er opbygget meget kompakt, vil det være vanskeligt for brugeren at ramme den rigtige tast, samt holde styr på, hvad hver kontakt har af respons. Oprindeligt var der også en anden årsag til, at der blev konstrueret et tastatur fra bunden. Grunden var, at det det dermed kunne undgås at have pins til at stikke opad, som det gør på figur 22. Det resulterede dog i, en masse ledninger, som der ikke var meget styr over, som det kan ses på figur 23. Da det senere blev besluttet at forbinde til et connector shield (forklaret under afsnittet connector) blev der placeret et fladkabel på printet, hvormed der alligevel kom til at sidde ledninger på oversiden, dog meget mere overskueligt, se figur 19.



Figur 22: Færdiglavet tastatur (27)

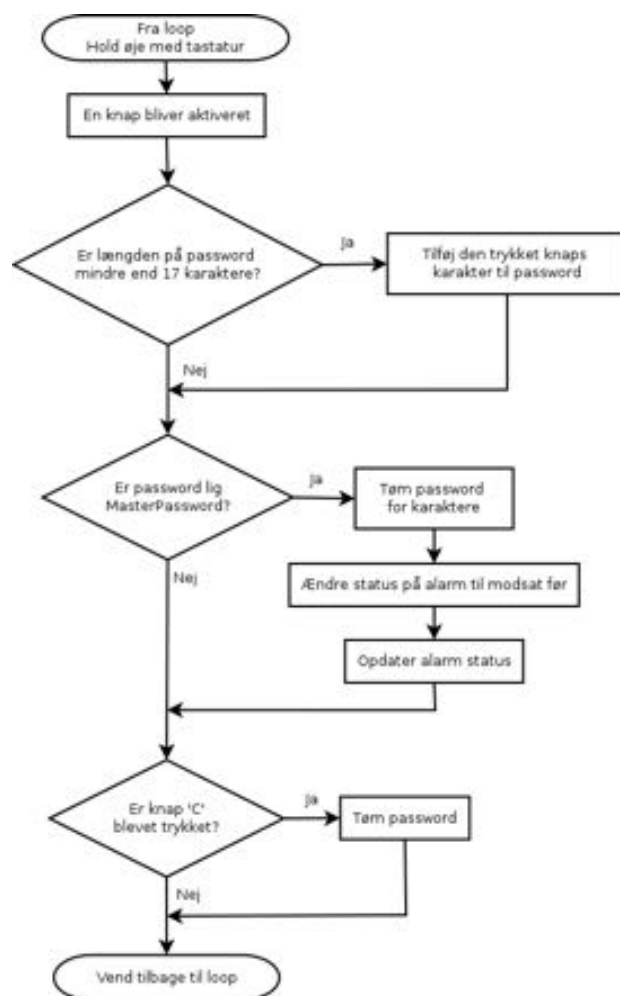


Figur 23: Uoverskuelige ledning til tastatur

Beskrivelse af funktionen for tastaturet

Tastaturet er opbygget så det følger flowchartet på figur 24. Programmet kører ved, at der først sker noget, hvis en knap bliver aktiveret. Denne knaps karakter bliver da tilføjet til en streng (password) såfremt der ikke er mere end 17 karakter i password. Denne grænse passer med de 16 karakter, som displayet kan vise. Overskrides dette antal må man slette indholdet af password ved tryk på 'C' (se evt. figur 19).

Hvis password strengen er magen til MasterPassword strengen, der er angivet til at være "584739G", hvor 'G' er godkend-knappen. Godkend-knappen er et udtryk for, at man skal vælge at det er korrekt, det man har indtastet (ligesom hvis man logger ind på hjemmesiden skal man trykke på login). Det er ikke muligt, at ændre masterkoden.



Figur 24: Flowchart for tastatur

Beskrivelse af switch statment

Programmet for LCD'et er opbygget gennem en switch statement. En switch statement virker meget på samme måde som en if funktion, dog med undtagelser. Det gode ved switch statement er, at den optager færre kodelinjer, mens den samtidigt er hurtigere at udføre, da den kun

```
switch (state) {  
  case 0:  
    lcd.print("Alarm OFF");  
    break;
```

Figur 25: Oversigt over kode for state 0, med switch statement

skal tjekken en statement, mod - kørte man med en if funktion - at tjekke alle cases. Switch statementen holder en variabel op mod forskellige cases (1). En case er den værdi som variabel kan tage. Passer værdien med en case bliver der udført en handling, som er unik for denne case. I programmet er der fem mulige cases: 0, 1, 2, 3 og 4.

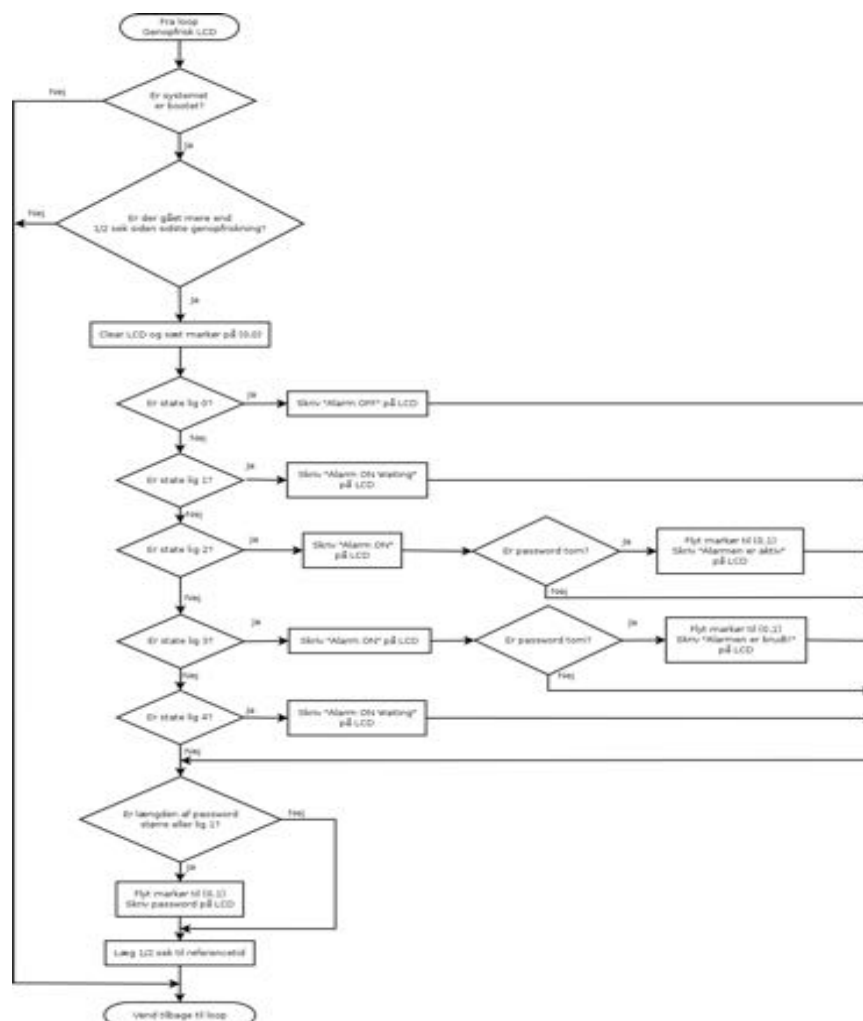
På figur 25 og 26 kan det sammenlignes hvor meget koder der skal til for at køre en if mod en switch (for programmets state 0). I koden, hvor der er implementeret switch statement er de kommandoer, der gør sig gældende for alle cases placeret uden for funktionen – indtastning af password.

```
if (state == 0) {
  if (password.length() == 0) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Alarm OFF");
  }
  if (password.length() >= 1) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Alarm OFF");
    lcd.setCursor(0, 1);
    lcd.print(password);
  }
}
```

Figur 26: Oversigt over kode for state 0, med if funktion

Flowchat for LCD

Alt hvad der skal skrives på LCD'et følger flowchartet på figur 27. Programmet er bygget således, at det bliver opdateret hvert halve sekund.



Figur 27: Flowchart for LCD

Tests

For både tastaturet og LCD'et blev tests gennemført. Udførelsen fandt sted samtidigt med tests af sikring af døre og vinduer, der blev beskrevet i tidligere afsnit. Det følger derfor samme princip, med andre hændelser. Testen, der fokuserede på hvorvidt displayet viste det den skulle på de rigtige tidspunkter, samt hvorvidt der også blev skrevet det indtastet password ned på displayet. Testene havde tilfredsstillende resultatet, efter der blev skruet op for kontrasten, der i begyndelsen var meget lav.

Logik analyse af I2C



Figur 28: Logik analyse af I2C

Idet der bruges SDA og SCL til at overføre data til LCD'et kan man lave en analyse på, om den faktisk sender det den skal. Dette kan gøres med en logik analysator, der kan udskrive hvilke hexadecimal der bliver sendt over. Samtidigt kan man se hvornår den gør det og hvor meget den sender. Et eksempel på dette kan ses på figur 28. Figuren viser boot sekvensen, hvor det tager cirka 2 ½ sekunder før den begynder at skrive til displayet, og da skriver for hvert 500 ms – cirka. Den lille afvigelse skyldes, at der ikke bliver brugt en tid, men en tæller. Tællerens opgave er, at tælle en op, for hver gang funktionen forløber – hvormed det kan ses, at det altså tager lidt længere tid at køre igennem end de 500ms. Det er da også forventet, da opdateringen af LCD bliver kørt fra loop, som har en del funktioner den skal igennem. Det kan faktisk, dog med undvigelser tilnærmelsesvis bestemme hvor lang tid det tager for loopet at køre igennem, ud fra logik analysatoren.

På figur 28 kan det ses at det tog 508.9 ms mellem to overførelser til displayet, men derudover skal der også pålægges tiden det tager at overføre. Dette udregnes med udtrykket under, hvor n er antallet af hexadecimaler. Der tages udgangspunkt i den overførelse, der sker ved startmarkeringen på figur 28 ved de 4 sekunder.

$$t = 508.9\text{ms} + \sum_{i=0}^n 0.0273009\text{ms} \cdot n$$

hvor de 0.0273 ms er det tid det tager at overføre en hexadecimal, samt der er 108 pakker. Dette giver dermed 511.85 ms.

Det skal dog nævnes, at dette kun gælder for overførelse for teksten "Alarm OFF", når der for eksempel i case 2 skal sendes som på figur 29, ja da vil det tage lidt længere tid.

```
29 case 2: //Tilfælde 2: Alarmen er t  
30   lcd.print("Alarm ON");  
31   if (password.length() == 0) {  
32     lcd.setCursor(0, 1);  
33     lcd.print("Alarmen er aktiv");
```

Figur 29: Sendelse til LCD

Det skal dog nævnes, at de hexadecimaler, der overføres, ikke kan oversættes direkte til til karakter, som der bliver vist på LCD'et. Det er I2C bussen der oversætter hexadecimalerne til de korrekte karaktere. At LCD'et så faktisk viser det korrekte er testet ved at kigge på displayet, hvorefter det kan konstateres at der faktisk skrives det korrekte.

Ethernet Shield

Ethernet shieldet er det tredje shield, som bliver placeret på Arduino Mega mikrocontrolleren. Det er placeret således, at det sidder oven på GSM shieldet. Ethernet shieldet er nok en af de vigtigste komponenter i dette projekt. Uden den ville være ikke være nogen forbindelse til hjemmesiden. Dermed ville der mangle nogle indstillinger for systemet samt muligheden for at (de)aktivere alarmen samt kunne se hvor, og hvornår, der har været indbrud.



Figur 30: Billede af ethernet shield (26)

I de følgende afsnit bliver de forskellige funktioner, som der forbinder Arduinoen med internettet beskrevet. For dette ethernet shield er det vigtigt, at man ikke samtidigt kobler noget på pins 10, 11, 12 og 13 samt 50, 51, 52 og 53. Disse pins er i princippet de samme, men grundet Arduino Megas udformning skal begge sæt pins forblive reserveret til shieldet (2).

Klient

Hver gang der oprettes forbindelse til serveren gøres dette gennem en klient. En klient er et stykke software (kan også være hardware) der sender en forespørgsel til et andet program, som i dette system ligger på en server. For eksempel laves der en forespørgsel i funktionen PrepareAlarm, der kan ses på figur 31. På figur 31 er forespørgslen dog opdelt i flere stadier, der tilsammen henter hjemmesiden Read_Alarm.php.

```
48 | if (client.connect(server, 80)) { //Kan der oprettes forbindelse til serveren?
49 |   client.print("GET /HX-15-c-el/glen0930/Read_Alarm.php"); //Indlæs siden <--
50 |   client.println(" HTTP/1.1");
51 |   client.print("HOST: ");
52 |   client.println(server);
53 |   client.println("User-Agent: arduino-ethernet");
54 |   client.println("Connection: close");
55 |   client.println();
```

Figur 31: Eksempel på forespørgsel

Beskrivelser af funktioner

Der er, som nævnt, forskellige funktioner, som hvert har sit formål. For at introducerer dem kort kan der i tabel 4 ses en oversigt over navnet på funktionerne samt hvad deres funktion, kort beskrevet er. En uddybende forklaring følger i senere afsnit

Funktionsnavn	Funktion
PrepareAlarm	Opretter forbindelse til http://htx-dev.ucholstebro.dk/HX-15-c-el/glen0930/Read_Alarm.php , hvis der ikke er forbindelse.
ReadEthernetSite	Læser hvad der står på ovenstående hjemmeside.
AlarmStatusCheck	Analysere resultatet fra ReadEthernet-Site.
UpdateAlarmStatus	Opdaterer status på alarmerne på hjemmesiden gennem førnævnte hjemmeside.
UpdateLog	Sender en værdi til hjemmesiden http://htx-de.ucholstebro.dk/HX-15-c-el/glen0930/Update_log.php , der angiver hvilken alarm er blevet brudt.
UpdateFilename	Indlæser hjemmesiden http://htx-dev.ucholstebro.dk/HX-15-c-el/glen0930/UpdateNameFTP.php

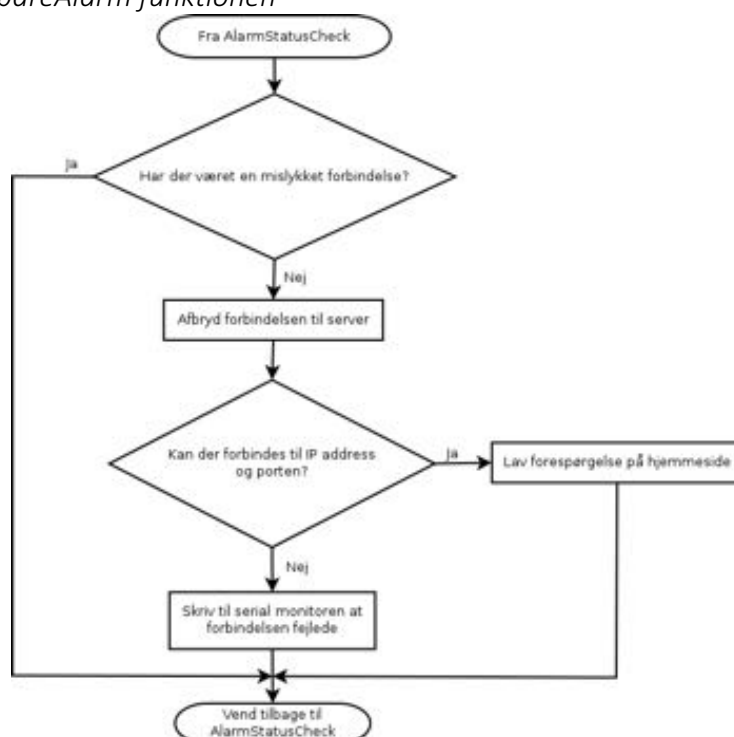
Tabel 4: Tabel over funktioner til ethernet shieldet

Beskrivelse af PrepareAlarm funktionen

PrepareAlarm funktionen bliver kaldt fra AlarmStatusCheck, hvor den også returnerer til, når dens opgave er fuldført.

PrepareAlarm funktionen har til formål at klargøre forbindelsen til server, således at funktionen ReadEthernetSite kan læse, hvad der står på hjemmesiden. Dog med en undtagelse. Har der været en mislykket forbindelse fra UpdateAlarmStatus funktionen skal der ikke klargøres en forbindelse. UpdateAlarmStatus skal nemlig have mulighed for først at opdatere status på hjemmesiden, da den ændring man har udført på det fysiske kontrolpanel vil blive slettet, hvis man læser indholdet fra hjemmesiden.

Flowchart for PrepareAlarm funktionen



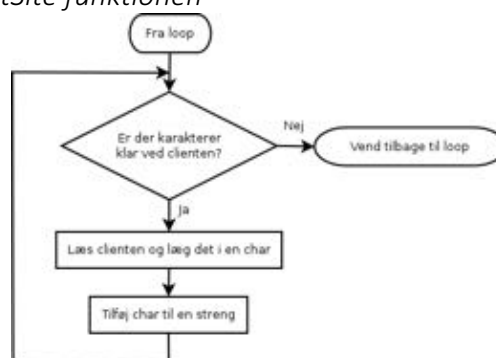
Figur 32: Flowchart for PrepareAlarm funktionen

I bilag 5 findes koden for funktionen for ovenstående flowchart under funktionen PrepareAlarm.

Beskrivelse af ReadEthernetSite funktionen

ReadEthernetSite funktionen har det ene formål, at så længe klienten har nogle karakterer, som kan læses, læser denne funktion dem. Den lægger derefter disse karakterer i en streng, som senere kan behandles i AlarmStatusCheck funktionen.

Flowchart for ReadEthernetSite funktionen



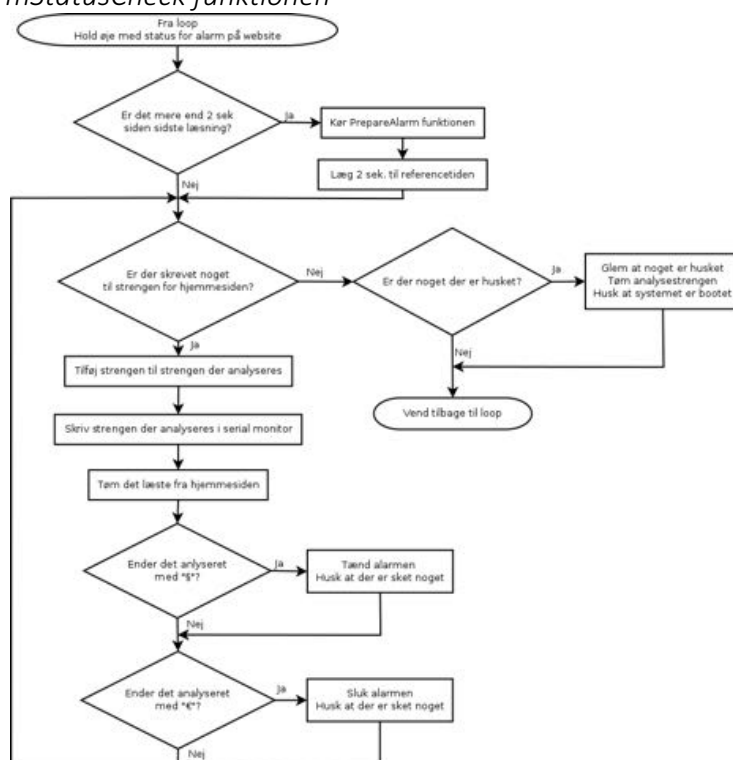
Figur 33: Flowchart for ReadEthernetSite funktionen

I bilag 5 findes koden for funktionen for ovenstående flowchart under funktionen ReadEthernetSite.

Beskrivelse af AlarmStatusCheck funktionen

Denne funktion tjekker om alarmen er høj eller lav ifølge hjemmesiden. Den gør dette ved, at analyserer den streng, som bliver hentet fra ReadEthernetSite funktionen. Afhængig af outputtet af denne streng bliver alarmen tændt eller slukket. Slutter strengen med et § tændes alarmen. Slutter strengen derimod med et € slukkes alarmen.

Flowchart for AlarmStatusCheck funktionen



Figur 34: Flowchart for AlarmStatusCheck funktionen

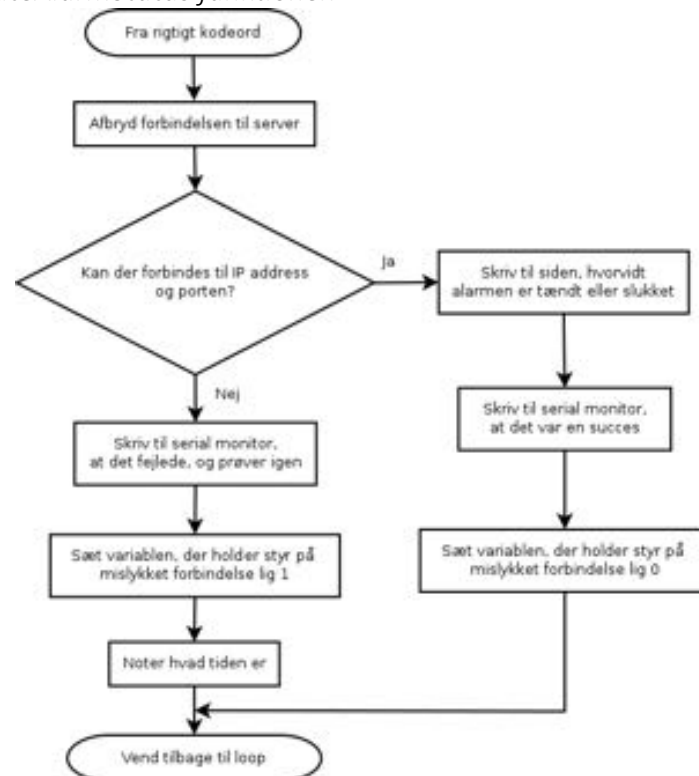
I bilag 5 findes koden for funktionen for ovenstående flowchart under funktionen AlarmStatusCheck.

Beskrivelse af UpdateAlarmStatus funktionen

Denne funktion har til opgave at opdatere status for alarmer på hjemmesiden. Den gør dette ved at blive kaldt når man har indtastet den korrekte kode. Derefter forbinder den til en IP adresse, som den kan skrive til. Der skriver den enten at alarmeren er tændt eller slukket, selvfølgelig afhængig af tilfældet.

Det kan på flowchartet bemærkes, at der er en variabel, som ændres til 1, hvis det mislykkes at oprette forbindelse. Dette gøres for at undgå, at systemets status bliver opdateret fra hjemmesiden næste gang loopet bliver kørt, da det ikke vil have ændret sig, hvis forbindelse i UpdateAlarmStatus mislykkes. Denne funktion skal derimod have lov til, igen, at opdatere alarmens status på hjemmesiden.

Flowchart for UpdateAlarmStatus funktionen



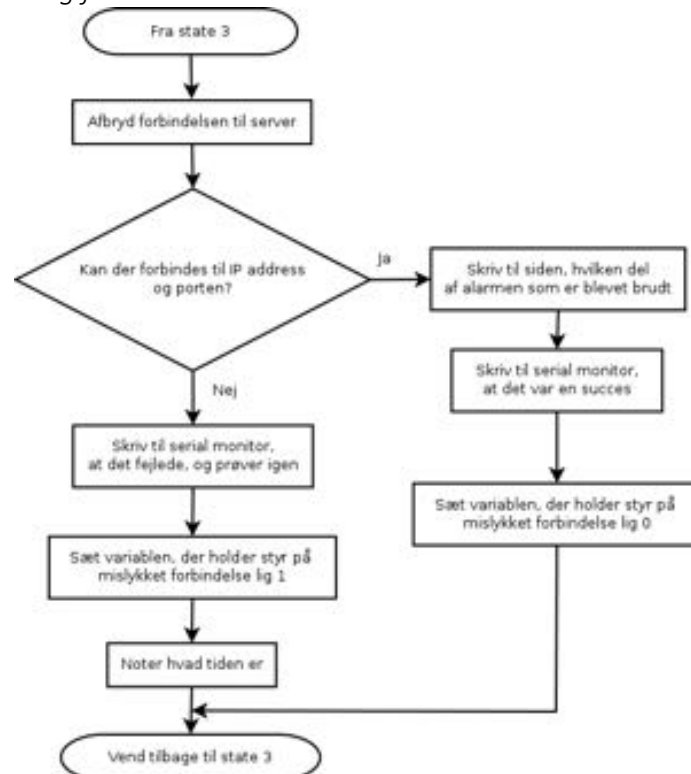
Figur 35: Flowchart for UpdateAlarmStatus funktionen

I bilag 5 findes koden for funktionen for ovenstående flowchart under funktionen UpdateAlarmStatus.

Beskrivelse af UpdateLog funktionen

UpdateLog funktionen fungerer stort set, som UpdateAlarmStatus. Forskellen er, at i stedet for at ændre alarmens status, som UpdateAlarmStatus gjorde, skal denne funktion fortælle serveren hvilken del af alarmen der er brudt.

Flowchart for UpdateLog funktionen



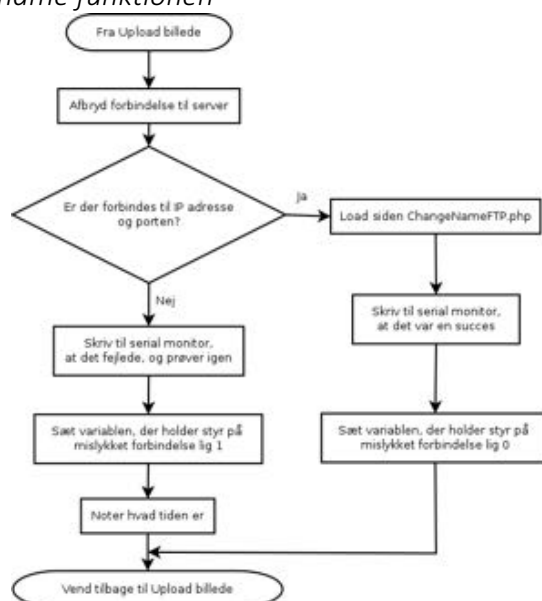
Figur 36: Flowchart for UpdateLog funktionen

I bilag 5 findes koden for funktionen for ovenstående flowchart under funktionen UpdateLog.

Beskrivelse af UpdateFilename funktionen

UpdateFilename har til formål at forbinde til siden ChangeNameFTP. Når dette sker bliver der nemlig opdateret navnet på den fil, der er uploadet fra SD-kortet.

Flowchart for UpdateFilename funktionen



Figur 37: Flowchart for UpdateFilename funktionen

I bilag 5 findes koden for funktionen for ovenstående flowchart under funktionen UpdateFilename.

Komplikationer

Der har igennem de mange timer, der er brugt på at få kommunikationen mellem Arduinoen og internettet, været nogle forhindringer, hvor mange af dem, dog har været muligt at finde en løsning på. Det første problem der opstod var, at der slet ikke kunne forbindes til internettet. Det viste sig imidlertid, at er der isat et SD-kort i ethernet shieldet kan der ikke samtidigt oprettes forbindelse, medmindre SD biblioteket er indlæst, hvilket det på daværende tidspunkt ikke var. Skulle det have virket uden indlæsning af SD biblioteket, og SD-kortet stadig var isat burde man specifikt vælge, at pin 4 er et output og desuden sætte den høj. Det fremgår af dokumentationen (2), at det bunder i, at både SD og W5100 (ethernet chippen) anvender SPI bus gennem ICSP pins, hvilket ikke kan kommunikere med begge moduler på samme tid.

Polling

I koden bliver der brugt polling, hvormed brugeren ikke oplever en så lang ventetid.

Princippet i polling er, at loopet skal rundt så hurtigt som muligt. Man spørger da forskellige elementer om der er sket en ændring i forhold til det de skal hold øje med. På den måde kan brugeren ikke se at der spørges efter mange ting efter hinanden, derimod ligner det, at det kører samtidigt. (3)

Et eksempel på dette i koden kan ses gennem funktionerne ReadEthernetSite, PrepareAlarm og AlarmStatusCheck. Disse tre funktioner kunne faktisk være slået sammen til én funktion, men da den deles op, vil funktionerne løber hurtigere, idet PrepareAlarm opretter forbindelsen, ReadEthernetSite finder det der står på siden og sidst bliver det analyseret i AlarmStatusCheck.

Timeout

Det har nogle gange været et problem, at det tog Arduinoen for lang tid, at timeoutte, såfremt den ikke havde mulighed for at få forbindelse til serveren. Sker dette bliver alle funktioner, der ellers burde forløbe sat i mente. Altså mister man mulighed for at indtaste kode, bryde alarmen eller sende en SMS. Det er meget upraktisk, da det i givet fald vil betyde, at bliver en af kontakterne brudt, imens der laves en henvendelse til serveren vil det ikke blive opfattet, at der faktisk sker et indbrud. Af denne grund er der ændret i indstillingerne for ethernet biblioteket. De ændrede indstillinger er fundet gennem Arduino forummet (4). På forummet bliver der angivet, at ved at ændre i filen w5100.cpp kan man indstille sin egen timeout tid. Det gøres gennem de to linjer kode på figur 38.

```
43   setRetransmissionTime(0x07D0);  
44   setRetransmissionCount(3);
```

Figur 38: Kode til ændring af timeout indstillinger

Ved at sætte værdierne som overstående fås der cirka 600ms tid til at forbinde til serveren, da hvis man definerer værdien til 1 er det lig 100µs, så 0x07D0 bliver lig 2000, som igen er lig 200ms. Dernæst bliver der defineret, hvor mange gange den skal prøve, at forbinde, nemlig 3 gange. Ud fra databladet for W5100 (5) bliver det beskrevet, at setRetransmissionTime som standard er disse 200ms, men der bliver ikke nævnt noget om setRetransmissionCount, og man må dermed formode, at det er denne variable, der har den store betydning for timeouttet.

Der er dog en ulempe ved denne metode. Ulempen er, at der ændres i de grundlæggende indstillinger for biblioteket, i stedet for, som man bør, ændre variabelen i selve koden. Problemet vil sandsynligvis først opstå i fremtiden, hvis modulet kræver, at der er en længere timeout tid.

Test

Der har i løbet af hele projektet blev testet mange gange, for om der nu også kan forbindes med internettet og udføre de dertil egnede funktioner. I starten har der været mange problemer, som er blevet færre og færre desto mere af projektet blev gennemført. Desuden har det vist sig, at have styring over hvad der sker på hjemmesiderne er meget vigtig, for at det kan fungere med Arduinoen og vice versa; altså har koden i Arduino ikke styr på, hvad der sker på hjemmesiderne kan der hurtigt opstå fejl.

GSM shield

Da alarmsystemet skal have mulighed for at informere administratoren, hvis der sker et indbrud, via SMS, er det nødvendigt at have et GSM modul. I stedet for et modul ved siden af mikrocontrolleren er der brugt et shield. Shieldet kan ses på figur 39. Der er en ulempe ved GSM shieldet. Shieldet skal manuelt tændes for ved tryk på selve modulet.



Figur 39: Billede af GSM modul (25)

AT kommandoer

GSM shieldet bruger AT kommandoer. AT kommandoer kan med et GSM eller GPRS modul bruges til følgende (6):

- Information og konfiguration af modulet og SIM kort
- SMS
- MMS
- Fax
- Data og lyd over det mobile netværk.

Shieldet, der bliver brugt, bruger chippen SIM900, som understøtter både GSM 7.05 og 7.07 AT kommandoer. I det følgende bliver de anvendte kommandoer forklaret i forhold til GSM 7.07 AT kommandoerne (7).

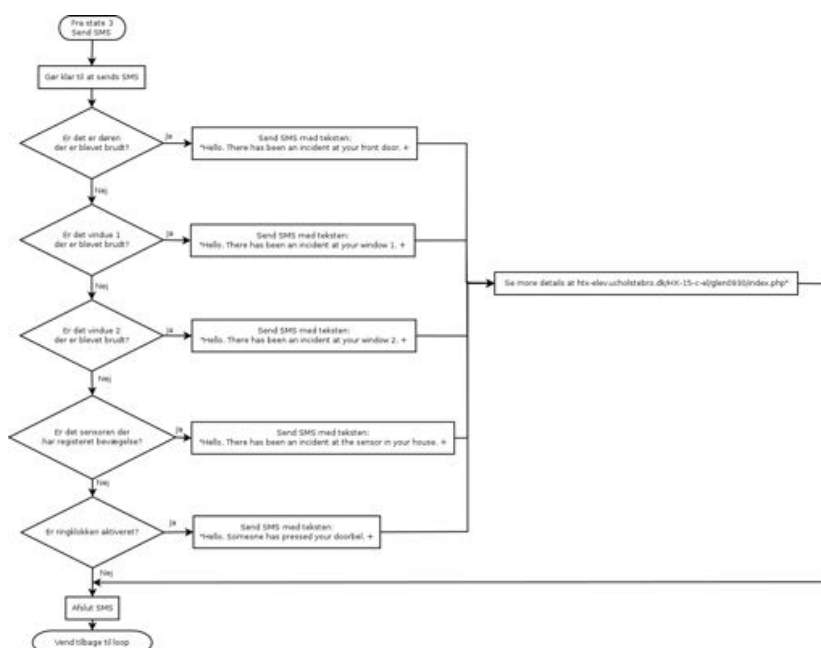
Den første AT kommando, der bliver anvendt, er kommandoen AT+CMGF, der vælger hvilket format Sms'en skal sendes i. Der findes to valgmuligheder: tekst og PDU³, hvor der naturligvis bruges tekst, der vælges ved angivelse af et 1-tal. (7)

Efterfølgende bruges kommandoen AT+CMGS, hvor der først angives hvilket nummer Sms'en skal sendes til, og derefter hvad Sms'en skal indeholde. (7)

Der afsluttes med karakter nummer 26, der i IRA teksttypen står for ctrl-z, hvilket sender beskeden.

Flowchart

På figur 40 ses flowchart for GSM shieldet. Programmet er opbygget ved at der bliver sendt en SMS når en af kontakterne eller sensoren er brudt eller der er trykket på ringklokken. Derfra er det mest et spørgsmål om indholdet i Sms'en⁴. Der er fire valgmuligheder for hvad der kan stå i Sms'en, men de slutter alle med samme tekst, som det også fremgår af flowchartet.



Figur 40: Flowchart for GSM shield

I bilag 6 findes koden for funktionen for ovenstående flowchart under funktionen SendSMS.

³ PDU (Protocol data unit) er en metode, hvor der også kan sendes kontrol informationer om adressen og andet information om brugeren.

⁴ Bemærk der er en stavefejl koden i form af "Se more [..]", som selvfølgelig skulle være "See more [..]"

Komplikationer

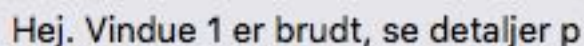
Det er måske blevet bemærket at teksten i Sms'en står på engelsk. Det skyldes en manglende mulighed for at sende æ, ø og å i Sms'en. Ud fra dokumentationen bruger GSM shieldet IRA koder når den sender en SMS, hvori der ikke kan skrives disse tegn.

Dette burde dog være muligt ved brug af AT kommandoen CSCS, hvor der kan bruges både hexadecimal og et defineret dansk / norsk karaktertabel, men af en ukendt årsag vil dette ikke fungerer. Ved test blev det klart, at den kan registrere, at der skiftes karaktertabel, men karakterne æ, ø og å bliver ikke håndteret korrekt. Der bliver derimod skrevet det, der ses på figur 41. Figuren er et udklip af den kommunikation der sker til og fra GSM shieldet på RX og TX benene (tilsluttet Arduino Mega på henholdsvis pin 8 og 7).

```
AT+CMGF=1
AT+CSCS=PCDN
AT+CMGS = "+4521521508"
Hello √¶√¶√•. There has been an incident at your window 1. Se more details at htx-
elev.ucholstebro.dk/HX-15-c-el/glen0930/index.php
```

Figur 41: Eksempel på fejl ved anvendelse af dansk / norsk karaktertabel

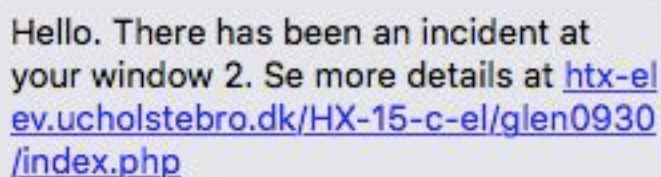
Dette er dog bedre, end et tidligere resultat, hvor al tekst efter et æ, ø eller å i sms'en bliver fjernet. Resultatet derved ses på figur 42.



Hej. Vindue 1 er brudt, se detaljer p

Figur 42: Eksempel på SMS uden æ, ø og å

Ved at skrive det på engelsk får man derimod det rigtige resultat, da det selvfølgelig ikke indeholder hverken æ, ø eller å.

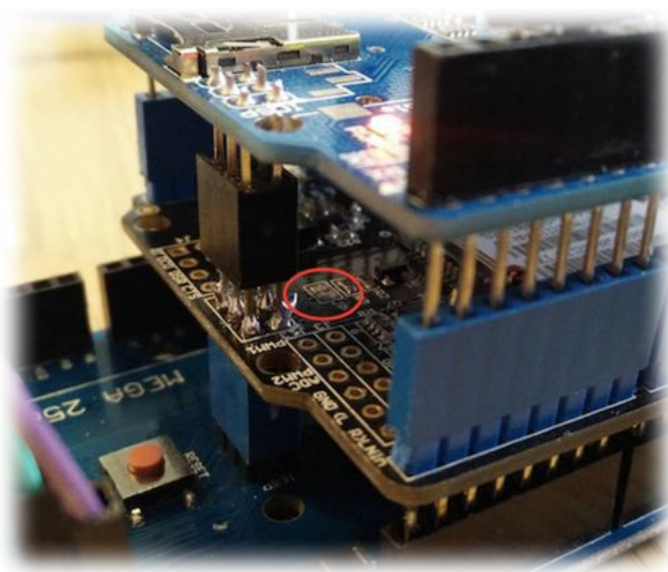


Hello. There has been an incident at
your window 2. Se more details at htx-elev.ucholstebro.dk/HX-15-c-el/glen0930/index.php

Figur 43: Eksempel på SMS på engelsk

Fejlen sker formentlig i processen, hvor Arduino skal oversætte æ, ø eller å til en af de andre karaktersæt. Hvor i denne proces fejlen sker, er dog uvist.

Tidligere blev den nævnt at shieldet skulle tændes manuelt ved tryk på en knap. Det er dog blevet programmeret således, at det kan gøres gennem koden. Dokumentationen skriver, at det kræver lodning på JP (jumper). Som standard skal der ikke være lodning på shieldet, men i dette tilfælde ligner det, at faktisk er tilfældet. Da det alligevel er muligt bruges denne metode. Hovedårsagen til valget, skyldes at man ellers



Figur 44: Markering af JP på GSM shield

som bruger ville være nødsaget til, at have en mere dybdegående viden om, hvor de forskellige komponenter sidder.

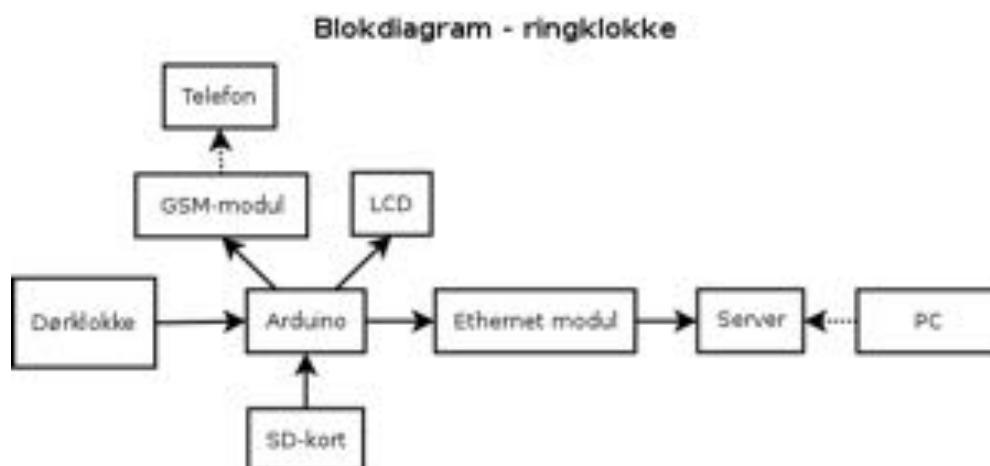
Test

De førnævnte komplikationer blev alle fundet igennem løbende tests af programmet, og til slut kunne det konstateres at det virker efter hensigten.

Som man kender det fra mobiltelefoner kan man først få forbindelse til GSM netværket efter lidt tid, og det er da også tilfældet med dette shield. Der kan derfor være et øjeblik, der ikke er mulighed for at sende en sms. Dette burde dog ikke være noget stort problem, da det kun er tilfældet når systemet genstartes / resettes – hvilket der ikke burde gøres.

Ringklokke

I systemet er der konstrueret en ringklokke, der skulle have den funktion, at når der blev ringet på døren, da ville der blive taget et billede af personen der stod foran døren, som da kunne tilgås fra hjemmesiden. Dertil ville der være behov for et kamera, som kan tage dette billede. Meningen var da, at der skulle opkobles et sådan kamera, men grundet den store kompleksitet i et kamera og samtidigt skulle have de andre dele systemet til at virke bliver der ikke taget et billede. Derimod ligger der et billede på SD-kortet, som er isat ethernet shieldet. Desuden skal der sendes en SMS til admin om, at der har været en at ringe på, dog kun hvis alarmen er tændt. Det vil jo være unødvendigt at få en SMS om det, hvis man er hjemme. Dermed har ringklokken et blokdiagram som på figur 45.



Figur 45: Blokdiagram for ringklokke

Funktionen med at uploade fra SD-kort til serveren sker gennem FTP. Koden der gør dette er taget fra Arduinos hjemmeside (8), og det er derfor ikke dybdegående dokumentation af denne funktion. Dog er de overordnede ideer bag koden forstået. Der er desuden lavet nogle ændringer i koden, så det passer sammen med resten af systemet, samt tilføje nogle enkelte funktioner.

En af de tilføjelser der er lavet er på linje 71-74, hvor den sender en SMS, såfremt alarmen er tændt. Denne stump kode ses på figur 46.

```
71 Doorbell_pressed = 1; //Husk at dørklokken er trykket
72 if (state > 0) { //Hvis alarmen er tændt
73     SendSMS(); //Send SMS
74 }
```

Figur 46: Stump koder der kører funktionen SendSMS

Når der oprettes forbindelse til FTP serveren kræves der selvfølgelig et brugernavn og en adgangskode. Denne adgangskode ligger i en fil for sig, som ikke er medtaget i bilag⁵. Konstruktionen for kodeordet er som der ses på figur 47, dog med et personligt password i stedet for der, hvor der står kodeord, som er skrevet i filen.

```
client.println(F("PASS kodeord"));
```

Figur 47: Linje kode der skriver kodeord

⁵ Findes dog i det elektroniske bilag. Der bedes om diskretion om denne kode, da det er en personlig kode for gruppemedlemmet Glenn Herping.

```
94 | if (!Rcv()) return 0;  
95 | client.println(F("USER glen0930"));  
96 | if (!Rcv()) return 0;  
97 | //Personligt password til FTP server - ikke inkluderet i rapporten  
98 | //(dog som elektronisk bilag)  
99 | #include "PasswordFile.h"
```

Figur 48: Stump kode, der logger ind på FTP serveren

Der er selvfølgelig også bestemt hvor den uploader til på FTP serveren, nemlig på den offentlige del i mappen Doorbell_IMG.

Da programmet ikke kan køre andre funktioner mens der uploades til FTP skrives der på LCD'et en besked om, at der uploades. På den måde kan man undgå forvirring fra brugers side, da det forklarer hvorfor fx tastaturet ikke reagerer i dette tidsrum. Oprindeligt blev der uploadet en fil på cirka 520kB, hvilket tog cirka 25 sekunder at uploade. Dette vil gøre systemet mindre sikkert, da man på den måde vil kunne ringe på klokken, bryde døren op og lukke døren igen og dermed undgå at bryde alarmen. Derfor er der valgt et meget mindre billede, der fylder 65kB, der tager cirka 2 sekunder.

Den sidste ændring i koden er derfor en ændring af referencetiden for hvornår status fra hjemmesiden skal tjekkes. Ellers ville programmet efterfølgende spørge hjemmesiden et antal gange i træk.

Komplikationer

I starten kunne koden fra Arduino Playground ikke uploade en fil korrekt, den tilføjede for hver 64 bytes en ekstra byte, og dermed ødelagde filen. Hvorfor programmet præcist gjorde dette er uvist, men efter udskiftning af SD-kortet til et nyt, er der ikke længere problemer med det. Filen kunne sagtens vises på computer, hvis man satte den direkte i, og havde da også den rigtige størrelse (i bytes), når man læste den fra Arduinoen. Problemet kom først i det øjeblik man begyndte at uploade.

PIR-bevægelsessensor

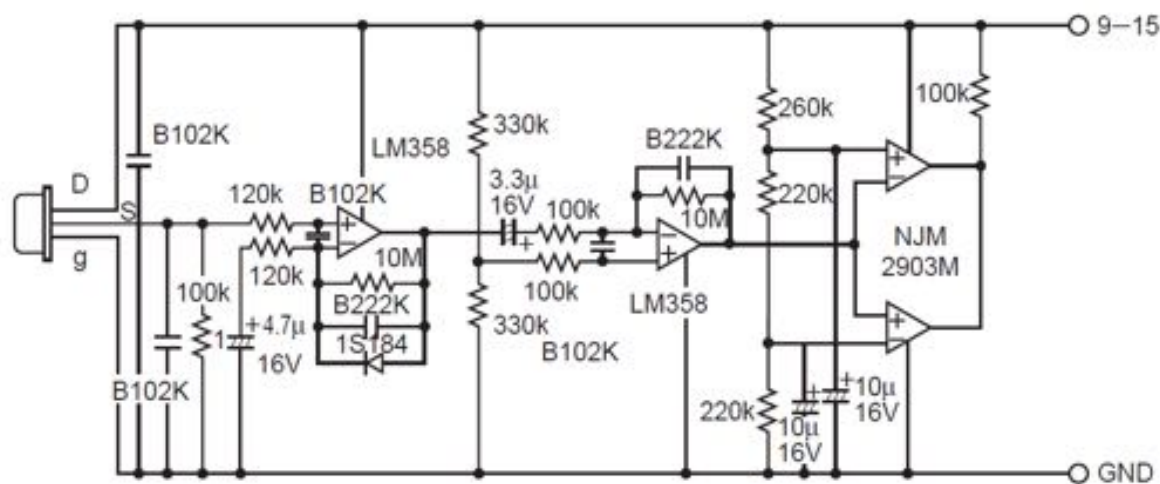
Bevægelsessensoren vi bruger i projektet er en såkaldt PIR-sensor (*passive infrared sensor*). Det er altså helt grundlæggende et elektrisk apparat der kan måle det infrarøde lys (IR), som objekter udsender inden for sensorens rækkevidde. Dette kan man udnytte til forskellige formål, hvor PIR-baserede bevægelsesdetektorer, eller normalt bevægelsessensorer, er det mest kendte.

Principielt kan det lade sig gøre, fordi levende organismer og i dette tilfælde mennesker udsender varme i form af infrarød stråling. Strålingen kan ikke ses med det blotte øje, fordi bølgelængden ligger over det synlige lysspektrum, der ligger fra 380 til 750 nm, hvor infrarød ligger fra 750 til ca. 1000 nm. Den passive infrarød sensor opfanger derfor den varme som udsendes af forbipasserende.

Selve sensoren virker ved at opfange lyset på en sensorflade, som er belagt med et film af såkaldt pyroelektriske materiale, der på givet vis leder strøm når de udsættes for varme-stråling. Filmen er typisk af gallium nitrit, caesium nitrat, polyvinyl flourider forurennet med pheylpyrinid og kobalt phtalocyanin (9). Sensoren er meget følsom og er derfor ty-pisk integreret i et kredsløb med forskellige filtre og forstærkende funktionsdele. På den måde kan man behandle den lille ændring i spænding, der fremkommer når sensoren udsættes for infrarød stråling.

Sensorkredsløb

En PIR-bevægelsessensor kan opstilles i adskillige konfigurationer med forskellige funktioner. Herunder ses det anvende eksempel af sådan et kredsløb, der kan give et læsebart signal.

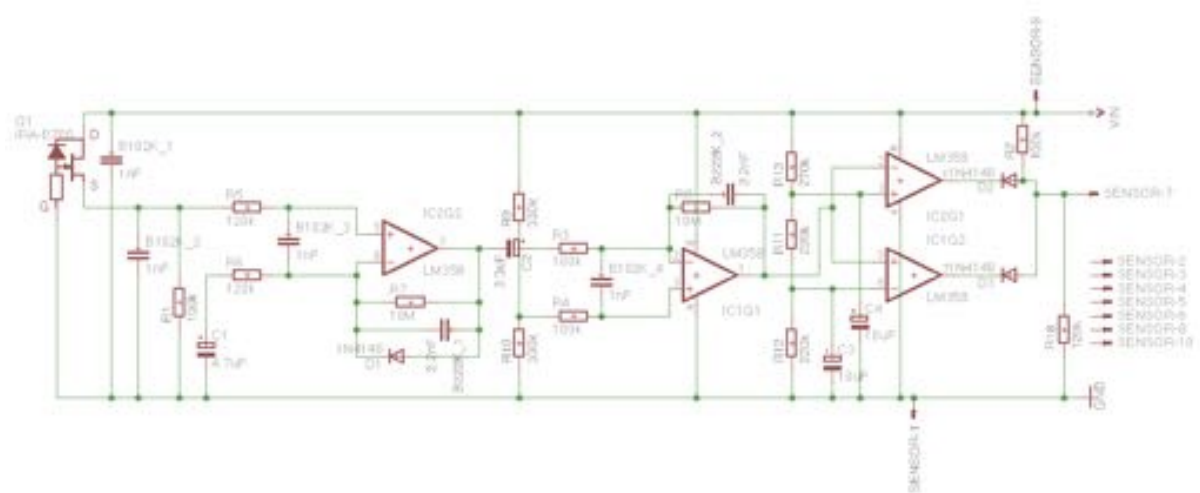


Figur 49: El-diagram for konfiguration af bevægelsessensor (10)

El diagram i Eagle

Fra det givne el-diagram fra kilden (10) er der lavet nogle små ændringer, så det passer med resten af Alarmduinoen.

For PIR-sensoren ser el diagrammet dermed ud, som på figur 50. De forskellige elementer og komponenter i kredsløbet er forklaret i de videre afsnit.



Figur 50: El diagram for sensor

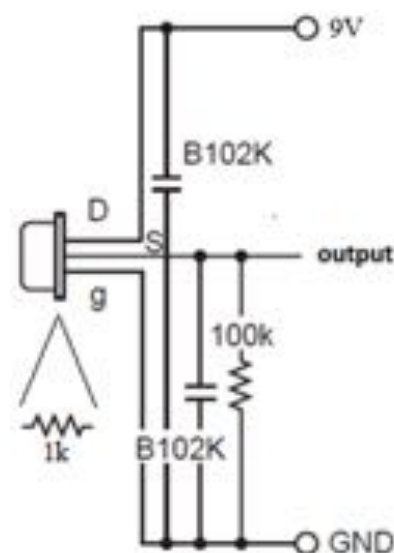
Beskrivelse af kredsløbets funktionelle dele

Sensor low-pass filter

Kredsløbet består af forskellige funktionelle dele, som hver især behandler signalet analogt, hvor der bl.a. sidder nogle forskellige filtre. Filtrene skal frafiltrere støj i forskellige frekvenser. I starten af kredsløbet sidder sensoren, som har udgangene D, S og g hvor D og g er forbundet med hhv. 5V og ground. S-benet er her, hvor det forstyrrede signal kommer ud som output. Signalet bliver behandlet undervejs af en konstruktion der virker som et lowpass-filter.

Begrænsningsfrekvensen eller knækfrekvensen opnås ved det punkt, hvor vekselstrømsmodstanden eller reaktansen X_C i 1nF kondensator er lig med den anslåede modstand i sensoren på 1k. Den 1k modstand i sensoren er ikke nogen tabelværdi, men derimod et kvalificeret bud. Frekvensen f_c opfylder:

$$f_c = \frac{1}{2\pi \cdot R \cdot C}$$



Figur 51: Low-pass filter (udsnit)

Det skyldes at reaktansen eller vekselstrømsmodstanden i en kondensator er frekvensafhængig. Knækfrekvensen f_c , bliver i denne seriekobling beregnet til:

$$f_c = \frac{1}{2\pi \cdot 1k\Omega \cdot 1nF} = 159160\text{Hz} \approx 159\text{kHz}$$

Når der samtidigt gælder for reaktansen X_C og modstanden:

$$1k\Omega \sim R \sim X_C = \frac{1}{2\pi \cdot 159\text{kHz} \cdot 1nF}$$

Det betyder derfor, at vekselstrømsmodstanden er omvendt proportional med frekvensen, og kondensatoren derfor vil begynde at kortslutte signalet og afdæmpe spændingen, når frekvensen kommer op over knækfrekvensen på 159 kHz. Men det er en teoretisk forudsætning af, at udgangsspændingen U_{output} er lig summen af spændingerne over modstanden U_R og over kondensatoren U_C . Frekvensen på 159 kHz er høj, men det er alligevel lavt nok til at frasortere størstedelen af radiobølgerne, så de ikke forstyrrer signalet. Radiofrekvenser ligger i området fra 3 kHz til 400 GHz.

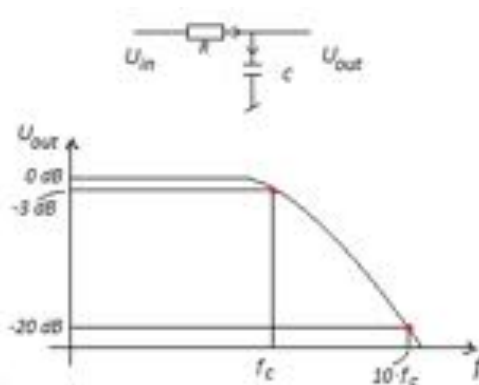
I virkeligheden forholder det sig ikke helt sådan, da de to vekselspændingerne over de to komponenter er forskudt i tid. I virkeligheden vil f_c ligge ved ca. -3 dB, som vist på diagrammet til højre, hvor akserne er angivet i en logaritmisk skala. Spændingen i dB er angivet som:

$$\text{dB} = 20 \cdot \log_{10} \left(\frac{U_{\text{out}}}{U_{\text{in}}} \right)$$

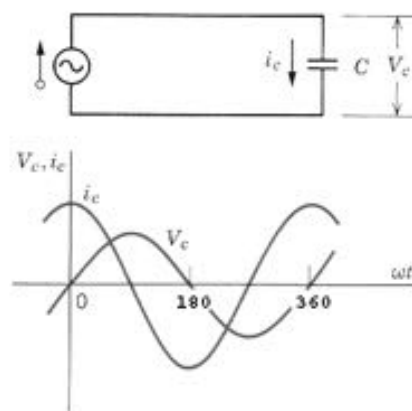
Grunden til at f_c ikke vil ligge ved 0 dB, findes i kondensatorens elektriske egenskaber. Når strømmen veksler hen over kondensatoren, vil spændingen

U_C ligeledes veksles med en tidsforskydning på $\frac{\pi}{2}$ radianer eller 90 grader i forhold til strømmen I_C .

Det skyldes at der under opladning af kondensatoren går en ladestrøm. Efterhånden vil strømmen under tilbageløb falde og spændingen over kondensatoren stige. Med andre ord vil den løbende strøm være 0, når hældningen og dermed ændring i spænding er 0.



Figur 52: Frekvenskarakteristik for lavpass filter



Figur 53: U og I vekselstrøm over kondensator

(30)

Det ses også illustreret på figur 54 til højre, hvordan spændingen er forskudt med $\frac{1}{4}$ periode i forhold til strømmen.

Pga. faseforskydning virker seriekoblingen som en spændingsdeler, derfor må U_{out} være lig:

$$U_{out} = I \cdot X_C = \frac{X_C}{\sqrt{X_C^2 + R^2}} \cdot U_{in}$$

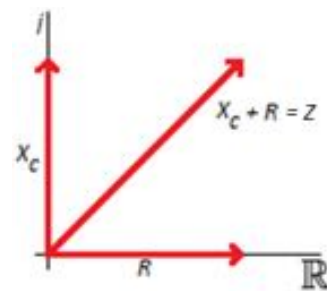
Pythagorasledet under brøkstregen skyldes, at reaktansen er en kompleks værdi og sumvektoren er impedansen i det komplekse plan. Dvs. den samlede modstand og impedans Z er ikke summen, men størrelsen af det komplekse tal, som reaktansen og resistansen udgør. Skulle man opstille et udtryk for udgangsspændingen, skulle man gange med den imaginære vektor j :

$$U_{out} = \frac{\frac{1}{2\pi \cdot f \cdot C \cdot j}}{\frac{1}{2\pi \cdot f \cdot C \cdot j} + R} \cdot U_{in}$$

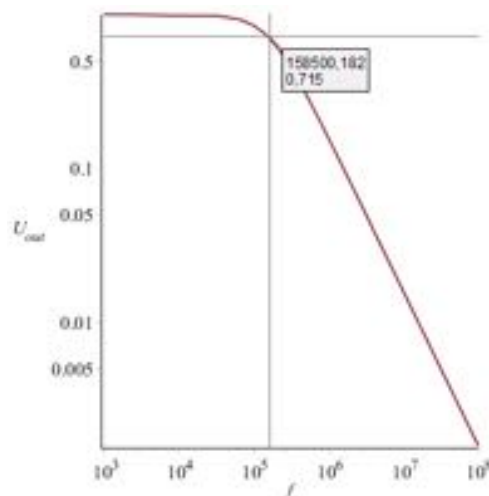
Opstiller man et udtryk for dette, hvor U_{in} sættes til 1, og tager den absolutte værdi, vil man få samme frekvenskarakteristika som før, hvor

knækfrekvensen ses markeret på grafen på figur 55. Det passer med at man ved knækfrekvensen vil have en udgangsspænding der er lig:

$$U_{out} = \frac{(U_C + U_R)}{\sqrt{2}}$$



Figur 54: Reaktansen i det komplekse plan



Figur 55: Frekvenskarakteristika af low-pass filter

Første bånd-pass filter med tilbagekobling

Den næste funktionelle del i kredsløbet er signalforstærkende modkobling, bestående af en operationsforstærker, som forstærker et bånd af frekvenser – heraf bånd-pass filter. Modkobling virker som en ikke-inverteret forstærkning, dvs. signalspændingen bliver forstærket positivt i forhold til modkoblingens modstand. Udgangsspændingen kan teoretisk set udtrykkes:

$$U_{out} = U_{in} \cdot \left(1 + \frac{R_M}{R_1}\right)$$

Forstærkningsgraden på $\frac{R_M}{R_1}$ kommer af forholdet

mellem modstanden i den ikke-inverterede ind-

gang og modkoblingsmodstanden, mens den 1 gangs forstærkning skyldes at spændingsforskellen på den ikke-inverterede og inverterede stort set er 0. Forstærkningsgraden A regnes til at være:

$$A = 1 + \frac{10M\Omega}{120k\Omega} = 84\frac{1}{3}$$

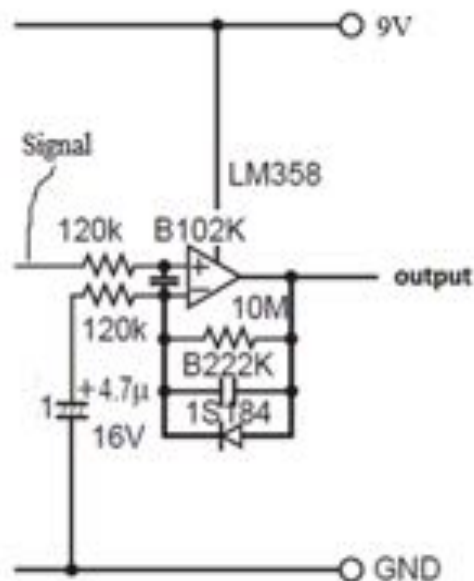
I pass-båndet har man altså en AC-forstærkning på ca. 84 gange. Den store forstærkning er nødvendig for de "svage" ændringssignaler fra IR-sensoren, som ifølge databladet har en "responsivity" på omkring 4,3mV ved 25 °C og 1 Hz. Ser man bort fra variationerne ved frekvensafhængighed og kondensatorenes virkning har man et niveau på omkring:

$$U_{out} = 4,3mV \cdot \left(1 + \frac{10M\Omega}{120k\Omega}\right) \approx 0,363V$$

Skal dette gælde, må reaktanserne være ca. lig modstandene, som den er omkring pass-båndet. Forstærkningens frekvensafhængighed afgøres af kondensatorenes reaktans og modstandenes resistans. Starter man med at se på den serieforbundede kondensator og resistor på det inverterende forstærkerben, kan man som udgangspunkt beregne knækfrekvensen for denne:

$$f_{c1} = \frac{1}{2\pi \cdot R_1 \cdot C_1} = \frac{1}{2\pi \cdot 120k\Omega \cdot 4,7\mu F} \approx 0,28Hz$$

I modkoblingen må kvækfrekvensen for den parallelforbundede kondensator og resistor være:



Figur 56: Første bånd-pass filter (udsnit)

$$f_{C2} = \frac{1}{2\pi \cdot R_2 \cdot C_2} = \frac{1}{2\pi \cdot 10M\Omega \cdot 2,2 nF} \approx 7,23 Hz$$

Dvs. passbåndet ligger fra ca. 0,28 Hz til 7,23 Hz, hvor AC-forstærkningen er ca. 83 gange. Ved at opstille en frekvenskarakteristika, kan man se hvordan forstærkningen afhænger af frekvensen. Forstærkningen A var:

$$A = 1 + \frac{R_M}{R_1}$$

Men fordi reaktansen indgår i beregningen af en forstærkningen uden for bånd-passet, må man opstille udtrykket for impedansen Z . Det huskes fra tidligere, at impedansen var sumvektoren i det komplekse plan.

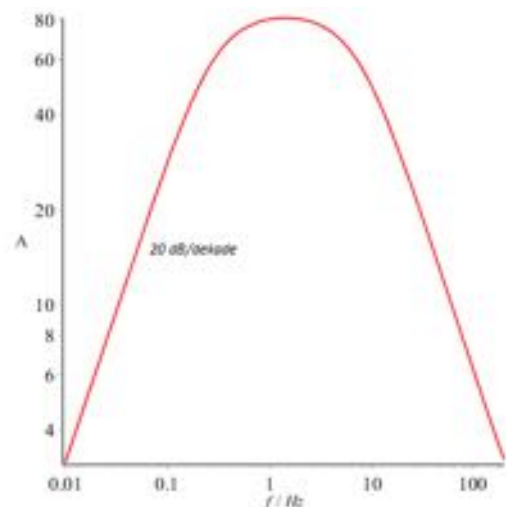
$$Z = R + X_C \cdot j$$

Hvor j er den komplekse forskydning på 90 grader mod uret. Indsættes impedansen og huskes det at R_M udgør en modstand i parallelkobling, må der for forstærkningen gælde:

$$A = 1 + \frac{\left(\frac{X_{C,B222K_1} \cdot j \cdot R_7}{X_{C,B222K_1} \cdot j + R_7} \right)}{X_{C1} \cdot j + R_6}$$

Men da reaktansen er frekvensafhængig med kapacitansen:

$$\begin{aligned} X_C &= \frac{1}{2\pi \cdot f \cdot C} \rightarrow \\ A &= 1 + \frac{\left(\frac{\frac{1}{2\pi \cdot f \cdot C_{B222K_1}} \cdot j \cdot R_7}{\frac{1}{2\pi \cdot f \cdot C_{B222K_1}} \cdot j + R_7} \right)}{\frac{1}{2\pi \cdot f \cdot C_1} \cdot j + R_6} \\ &= \frac{\frac{1}{2} \cdot j \cdot R_7}{\pi f C_{B222K_1} \left(\frac{\frac{1}{2} \cdot j}{\pi f C_{B222K_1}} + R_7 \right) \cdot \left(\frac{\frac{1}{2} \cdot j}{\pi f C_1} + R_6 \right)} \end{aligned}$$



Figur 57: Frekvenskarakteristika af første bånd-pass filter

Det svarer til at forstærkningen stiger eller falder med ca. 20 dB pr. dekad. Men det bemærkes også, at pga. impedansen vil forstærkningen i teorien aldrig nå helt op på de 84 gange, men snarere 80 gange, hvor den perfekte frekvens ligger omkring 1-2 Hz.

Anden bånd-pass filter

Signalet fra det første bånd-pass filter løber ind i endnu et bånd-pass filter. Ser man bort fra impedansen bliver den AC-forstærkningen ved frit genne-
nemløb 101 gange:

$$A = 1 + \frac{10M\Omega}{100k\Omega} = 101$$

Regner man igen med de ca. 0,363V ved sensor-responsivity på 4,3mV, svarer det til et udgangssignal:

$$U_{out} = \left(1 + \frac{10M\Omega}{100k\Omega}\right) \cdot 0,363V = 33,97V \approx 34V$$

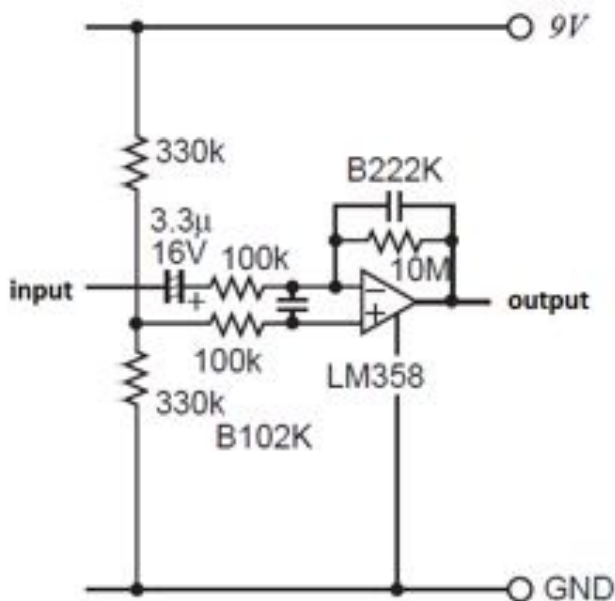
Men dette vil altså være den ultimative ændring, og det er langt fra de udløsende spændingsniveauer ved den efterliggende comparator. Det betyder derfor, at sensorkredsløbet i virkeligheden er bygget til at kunne føle de mindste ændringer i IR. Knækfrekvenserne for bånd-pass filteret beregnes uden hensyn til impedansen:

$$f_{c,1} = \frac{1}{2\pi \cdot 100k\Omega \cdot 3.3\mu F} \approx 0,482Hz$$

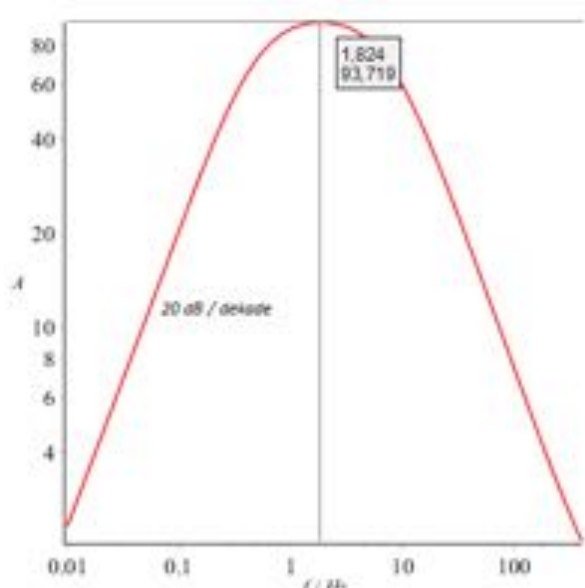
$$f_{c,2} = \frac{1}{2\pi \cdot 10M\Omega \cdot 2.2nF} \approx 7,234Hz$$

Bånd-passet ligner nogenlunde det samme som den forgangne, hvor den maksimale forstærkning ligger omkring 1-2 Hz.

Opbygningen af anden tilbagelkoblingen er anderledes fra det første bånd-pass filter, fordi der ikke sidder nogen diode i parallellforbindelse med kondensatoren og resistoren. Det betyder at strømmen har frit løb begge veje i tilbagelkoblingen, hvilket gør at udgangen "hurtigere" indfinder spændingsniveauet på de 4,5 V fra



Figur 58: anden bånd-pass filter (udsnit)



Figur 59: Frekvenskarakteristika af anden bånd-pass filter

spændingsdeleren. Derved udlignes de forstærkede signaler også hurtigere, så comparatoren reagerer meget kortvarigt.

Comparator "sammenligner"

Den sidste del af kredsløbet består af en såkaldt comparator. En comparator eller "sammenligner" er en elektrisk enhed der kan sammenligne to analoge indgangsspændinger eller strømme og derudfra sende et digitalt signal ud. En comparator består af et kredsløb med en operationsforstærker, som set på figur 60 til højre.

Selve ideen med comparatoren er at se på forskellen mellem de to indgangsterminaler, hvor den ene er referencespænding og den anden en indgangsspænding. På en comparator, som den på figur 60, ligger referencespændingen på den ikke-inverterede terminal og den variable indgangsspænding på den inverterede terminal. Det betyder at når output går lavt, når den variable indgangsspænding er højere end referencespændingen. På den måde kan man styre et udgangssignal ud fra én referencespænding.

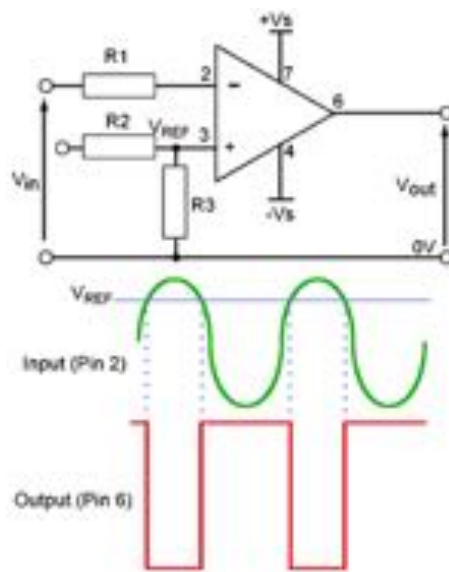
I kredsløbet anvendes der en comparator, for at give det rigtige output-signal, når sensoren reagerer på IR-stråling. Det ses på figur 61 at man i stedet for én forstærker har to, så der altså bliver to referencespændinger. På den måde er det muligt at styre outputspændingen ud fra et interval, der ligger mellem spændingen over R12 og R12+R11. Spændingen kan derfor beregnes ud fra Ohms lov:

$$U_{R12} = I \cdot R_{12}$$

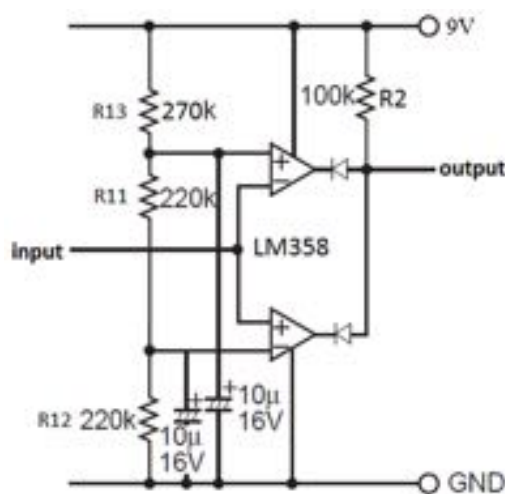
Men da strømmen I kan omskrives:

$$I = \frac{U_{acc}}{R_{12} + R_{11} + R_{13}} \rightarrow$$

$$U_{R12} = \frac{R_{12}}{R_{12} + R_{11} + R_{13}} \cdot U_{acc}$$



Figur 60: Udgangssignalers afhængighed
(28)



Figur 61: Dobbelt comparator med to reference-spændinger (udsnit)

Da forsyningsspændingen er 9V og modstandene er givet, må spændingen over R12 være:

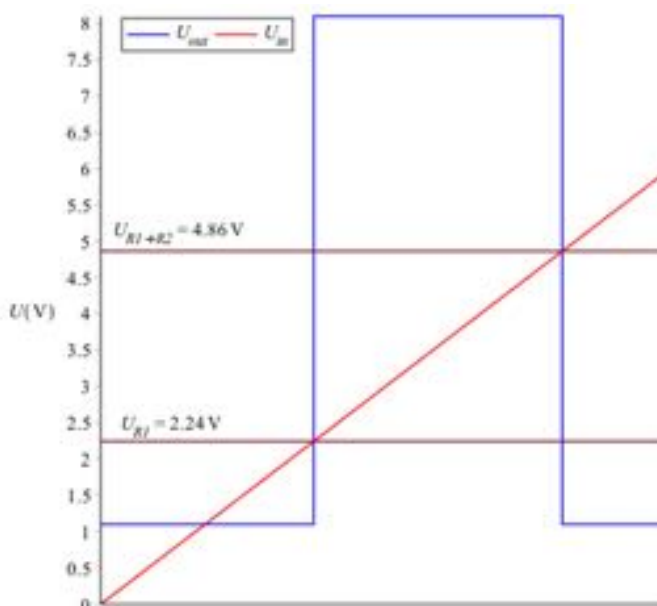
$$U_{R12} = \frac{220K\Omega}{220K\Omega + 220K\Omega + 270K\Omega} \cdot 9V = 2,7887V$$

Det samme kan gøres for spændingen over R12+R11:

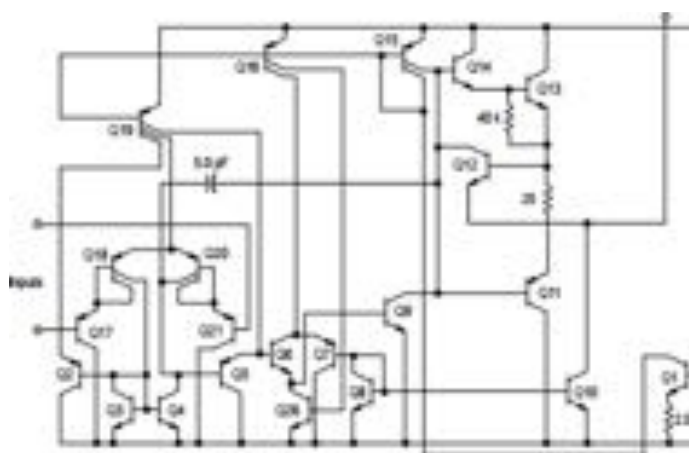
$$U_{R12+R11} = \frac{220K\Omega + 220K\Omega}{220K\Omega + 220K\Omega + 270K\Omega} \cdot 9V = 5,5775V$$

Efter beregningerne af de to referencespændinger betyder det, at udgangsspænding må gå høj, når indgangsspændingen ligger mellem ca. 2,79V og 5,58V. Teoretisk set vil udgangsspændingen nå op på 9V, såfremt der ikke løber nogen strøm i R2. Men den vil aldrig trække helt lavt, da der efter comparatoren ligger en diodespænding på ca. 0,7V. Der vil altså gælde for udgangsspændingen $9V \approx U_{out} > 0,7V$, når $2,79V < U_{in} < 5,58V$. I tilfælde af at indgangsspændingen ligger uden for intervallet, vil der gælde $0,7V \approx U_{out}$ for $2,79V > U_{in} \vee U_{in} > 5,58V$.

Man ønsker altså at opfange "forstyrrelserne" fra sensoren, således at den foregående tilbagekobling forstærker afgivelsen fra spændingen efter spændingsdeleren på ca. 4,5V. Når de forstærkede forstyrrelser fra sensoren resultere i en afvigelse fra intervallet, vil der gælde $9V \approx U_{out} > 0,7V$ som nævnt ovenover.



Figur 62: Graf for operationsforstærker



Figur 63: Komponentdiagram for operationsforstærkerne (29)

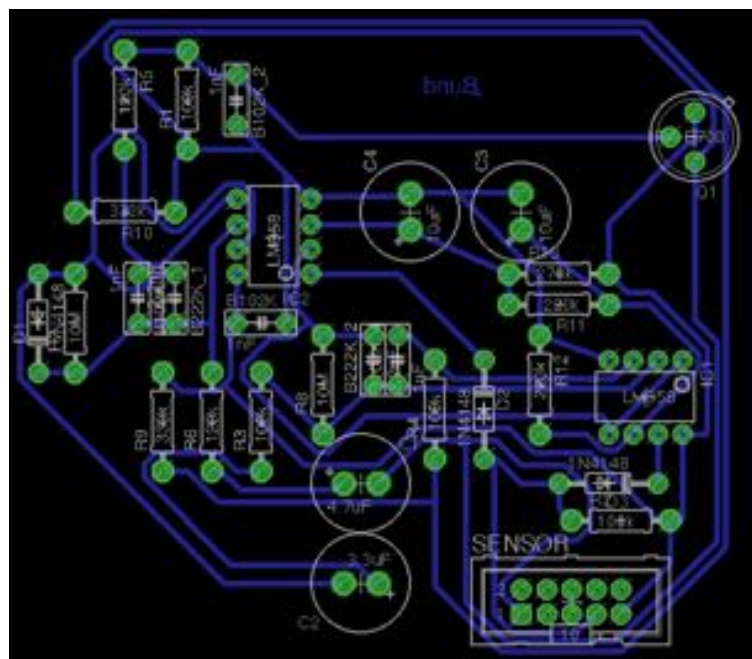
Målinger for ind- og udgangsspænding

For at bekræfte teorien lavede man sammenhængende målinger for ind- og udgangsspændingerne i comparatoren. Herved kunne man se, hvornår comparatoren havde sine omskiftningspunkter i forhold til teorien. På figur 62 ses graferne fra målingerne. Det bemærkes at omskiftningen sker tidligere end forventet. Dette kan skyldes, at operationsforstærkerne ikke kan trække helt høj, fordi der sker et spændingsfald over komponentens transistorer på omkring 0,4V. Det samme for U_{out} , som heller ikke kommer helt op på de 9V.

El-diagrammet for operationsforstærkeren LM358 ses nederst til højre. Her kunne man skyde på, at der var et mindre spændingsfald over nogle af transistorerne, som gør at udgangen ikke kan trække helt høj.

Fysisk udformning af bevægelsessensor

Ud fra el-diagrammet for sensor-kredsløbet lavede man et print-diagram i Eagle, som kan ses på figur 64.



Figur 64: Print-diagram for bevægelsessensor

Det bemærkes dog, at den 270k modstand R_{13} i virkeligheden skulle være en 260k ifølge konfigurationen fra sensorens data-blad. Det har ikke nogen betydelig indflydelse, men ændrer en smule på spændingsintervallet i comparatoren (her henvises til afsnittet "Comparator "sammenligner"). Dvs. det nedsætter begge spændinger. Prinet er forsynet med 9V fra Arduinoens Vinport, hvor den målte spænding er 8,54 V. Det betyder derfor at Arduinoen som udgangspunkt ikke er nogen "optimal" forsyningskilde til netop denne konfiguration af kredsløbet.

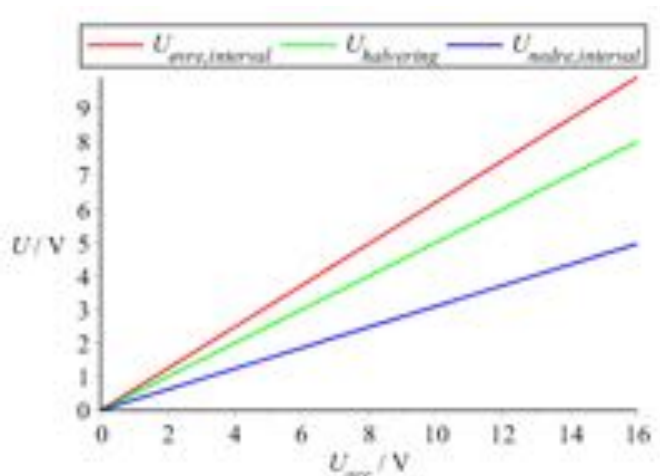


Figur 65: Billedet viser det færdige print af sensorkredsløbet

Regulering af følsomhed for sensor

Den anbefalede forsyningspænding er 9-16V, og man er faktisk under minimum her. Desto mindre en forsyningspænding, desto mere "følsom" bliver sensoren, fordi den halverede spænding nærmer sig grænserne for de udløsende spændingsniveauer i comparatoren. Der skal derfor gradvist mindre udsving fra sensoren til at comparatoren går lav.

Det kan illustreres ud fra graferne på figur 66.



Figur 66: Grafer for interval og indgangsniveau

$$U_{halvering} = \frac{U_{acc}}{2}$$

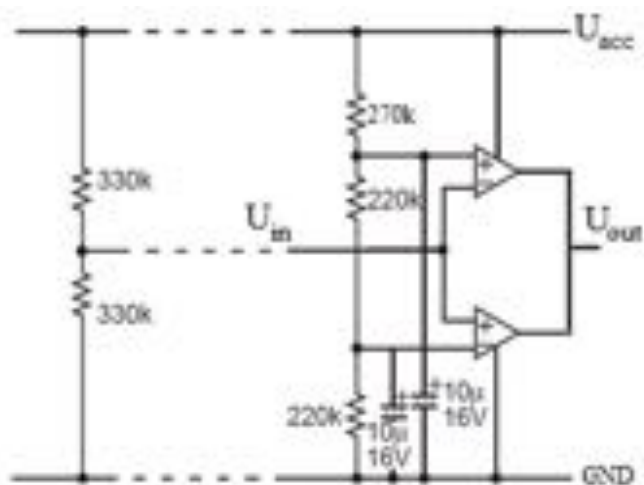
$$U_{nedre} = \frac{220K\Omega \cdot U_{acc}}{220K\Omega + 220K\Omega + 270K\Omega}$$

$$U_{øvre} = \frac{(220K\Omega + 220K\Omega) \cdot U_{acc}}{220K\Omega + 220K\Omega + 270K\Omega}$$

Dette bliver yderligere til et problem, fordi de målte niveauer i intervallet faktisk er lavere end de teoretiske, som er beregnet ovenover. En måde man kunne regulere følsomheden, ville være ved at spændingsdeleren før anden bånd-pass filter, således at man ikke fik den halve forsynings-spænding, men en der var lidt lavere end dette.

En anden del man kunne regulere er spændingsdeleren ved comparatoren, hvor man har de to 220k og en 270k modstand. Her kunne man ændre intervallet for, hvornår udgangssignalet går lavt og dermed gøre den mindre

følsom. Her er det nævneværdigt at bemærke, at man taler om følsomheden overfor spændingsniveauet og ikke frekvens.



Figur 67: Spændingsdeler og comparator (anden bånd-pass filter imellem)

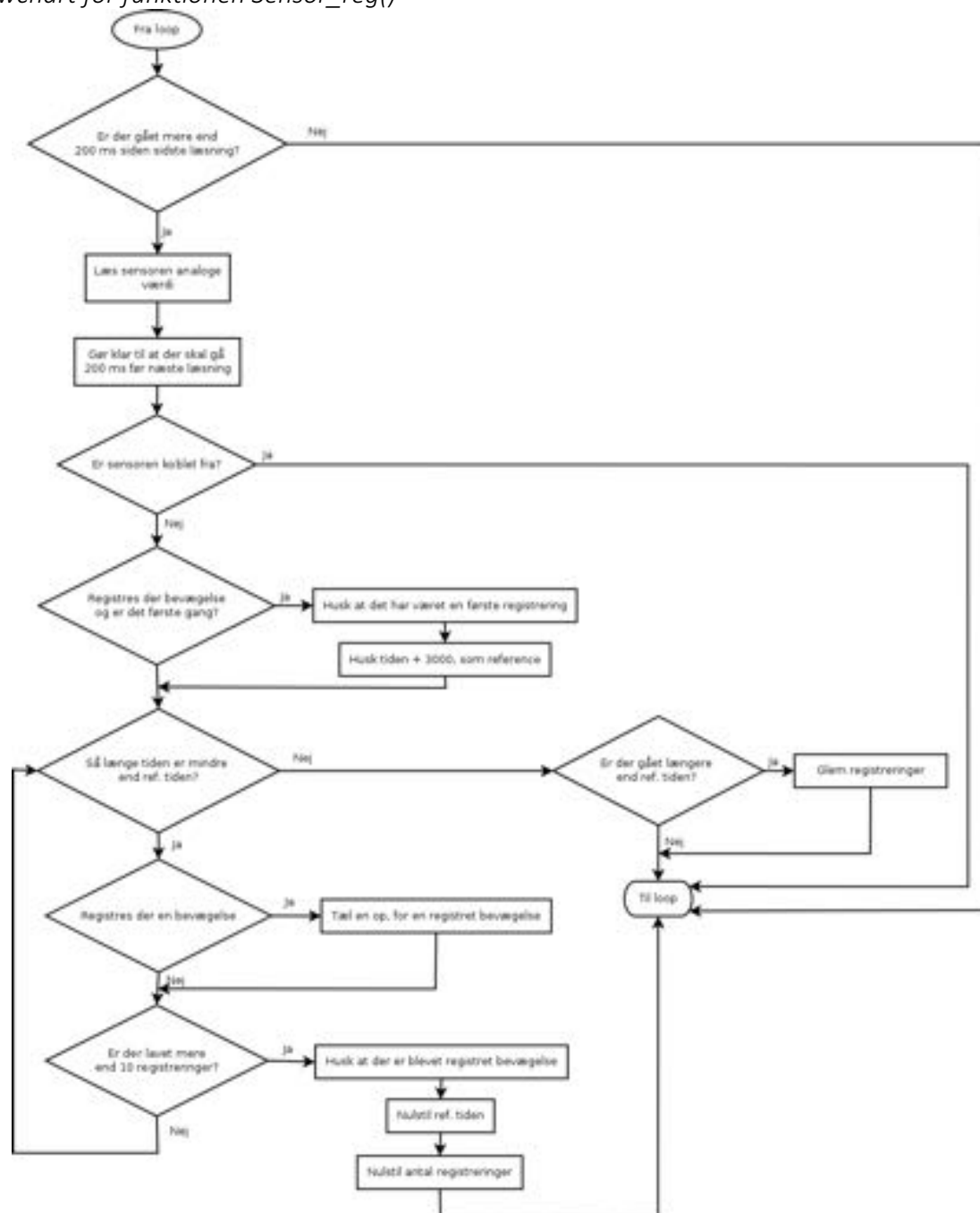
Beskrivelse af funktionen Sensor_reg()

Denne funktion, Sensor_reg, bliver kørt direkte fra loopet og har til formål at tjekke om der har været registreret af bevægelser foran sensoren. Den gør dette ved at måle på, hvad den analoge værdi, som Arduinoen får ind fra pin A15 ligger på. Såfremt der er registreret en bevægelse vil denne værdi ligge på cirka 212. Det er dermed denne værdi Arduinoen søger efter skal være korrekt. Der er dog det problem, som blev uddybet tidligere, at når der ikke er nogen bevægelser foran sensoren er den alligevel ikke helt stille og der kan forekomme nogle hændelser, hvor den alligevel ryger ned på disse 212, og dermed vil opføre sig som om der var sket en registrering. I programmet er derfor implementeret den specifikation, at for at der skal være registreret en bevægelse kræver det, at det er sket mere end 10 gange inden for 3 sekunder før det registreres som en endelig bevægelse, hvormed enkelte udsving bliver elimineret.

Funktionen forløber således, at der først tjekkes hvorvidt der er gået mere end 200 ms siden sidste læsning, passer dette fortsætter funktionen, ellers returner den tilbage til loop. Er det derimod sandt læses den analoge, og kan man ud fra denne værdi se, at sensoren er koblet fra, da returner den også. Er sensoren derimod ikke koblet fra bliver der tjekket om, hvorvidt der har været en bevægelse, og om det er første gang. Såfremt dette er sandt huskes der, at det er første gang, samt en reference tid på 3 sekunder. Dernæst

følger funktionen en while-løkke, som kører så længe der ikke er gået disse 3 sekunder. I while-løkken bliver der igen tjekket om der har været bevægelser, hvis dette er sandt tæller den op, indtil den når 10, hvor den da tolker dette som et entydigt svar på, at der har været en bevægelse, hvori den gør resten af programmet opmærksom på det og samtidigt nulstiller antal registreringer og reference tiden. Går der mere end 3 sekunder, hvor den ikke får disse 10 registreringer glemmer den det hele og vender tilbage til loop.

Flowchart for funktionen *Sensor_reg()*



Figur 68: Flowchart for sensoren

I bilag 8 findes koden for funktionen for ovenstående flowchart under *Sensor_reg()*.

Connector

For at gøre det nemt at samle alle delene fra både sensor, keypad, LCD og sikring af døre og vinduer er der konstrueret et shield, hvor der er placeret flere fladkabelforbinder, som da forbinder ud til diverse moduler.

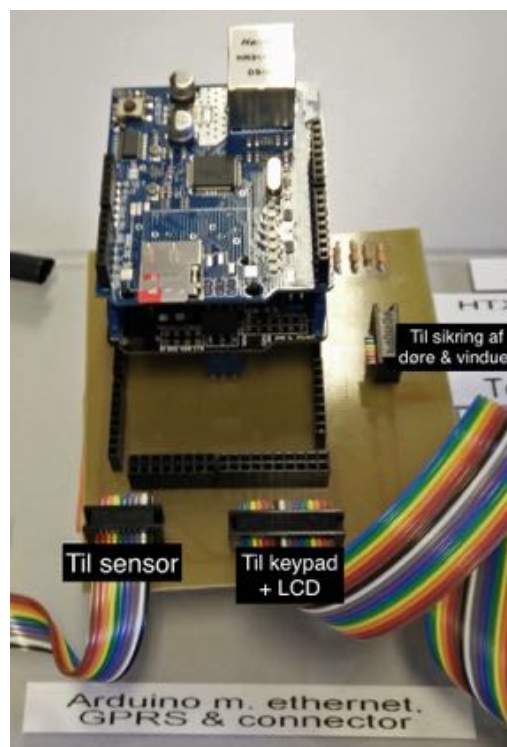
På figur 69 ses et billede af dette print, hvor der også er markeret, hvilket fladkabel der forbinder til hvilket modul.

Det har dog været nødvendigt at lave nogle små rettelser på printet, efter det blev konstrueret, da nogle forhold har ændret sig. En af disse ændringer er i stedet for, at forbinde sensoren til 5V skal den sluttes til 9V, som bliver leveret gennem strømforsyning, og ud i pin VIN (Volt in). Denne ændring kan ses på figur 70 (markeret med grøn) Da dette er tilfældet, men Arduinoen stadig betragter 5V som et højt signal skal der, når der kommer signal tilbage fra sensoren tilkobles en spændingsdeler, for at komme under de 5V igen. Da der fra figur 49 på side 40 kan ses, at der i forvejen er en modstand op til de 9V, på $100k\Omega$ behøver der kun en modstand mod GND. Denne modstand har størrelsen $120k\Omega$, hvormed indgangsspændingen til Arduinoen bliver:

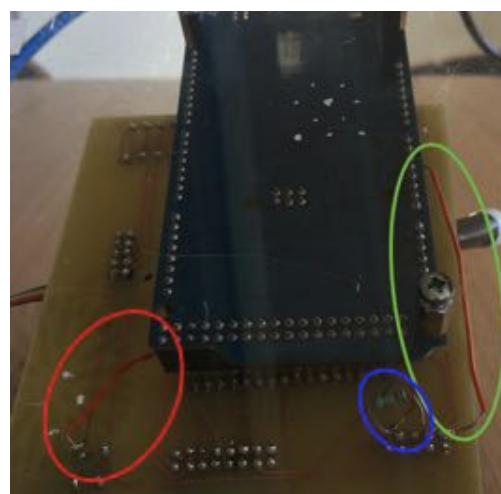
$$U_{in} = 9V \cdot \frac{100k\Omega}{100k\Omega + 120k\Omega} = 4.09V$$

På figur 70 (markeret med blå) kan denne modstand ses.

Idet sensoren blev koblet til 9V, og dette ben samtidigt var forbundet til dørklokke kontakten, er der desuden blevet lavet en lus hvor dørklokken bliver koblet på 5V.



Figur 69: Udformning af connector shieldet

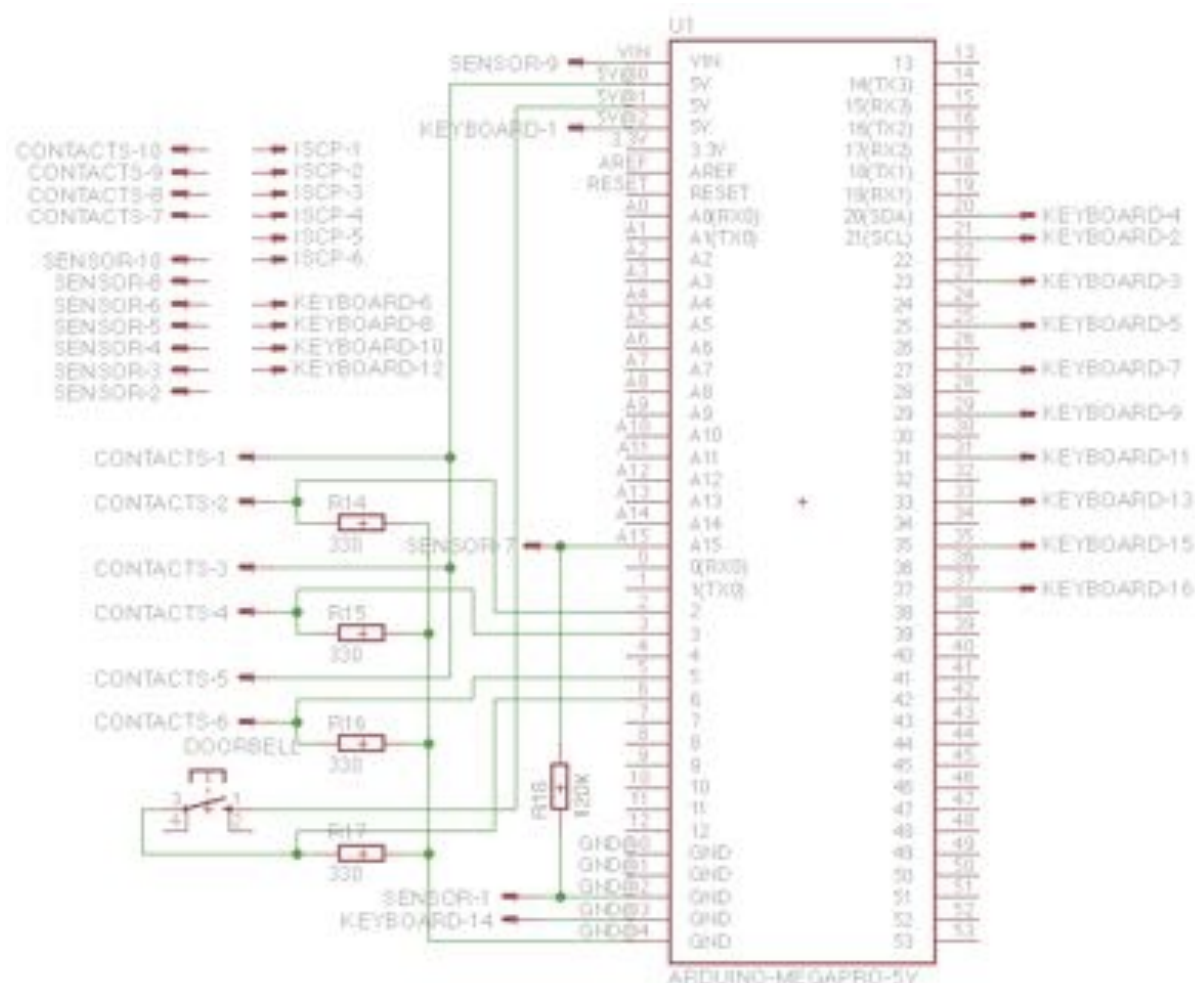


Figur 70: Ændringer på printet

El diagram

For connector shieldet er følgende el diagram gældende. På diagrammet kan det bemærkes at der i øverste venstre hjørne er en del overskydende pins fra fladkablerne. Skulle der laves et nyt print kunne der for bruges fladkabler med færre pins, hvis man ville formindske printet.

ISCP pins, der er placeret i øvre venstre hjørne er de pins, der sidder midt Arduino Mega, og bliver koblet op igennem de forskellige shields. Det er med på diagrammet, da de skal placeres på det diagram, der er lavet for printet.



Figur 71: El diagram for connector

Kontrolpanel (hjemmeside)

Index filen, er defineret som login siden til system. Man skal altså først igennem den, før man kan få adgang til systemet. På login siden er det eneste man skal bruge et brugernavn og en adgangskode. Dog kan man ikke oprette sig på systemet, hvis man ikke har en adgangskode. Det er derfor, at der følger et "master" brugernavn og adgangskode med til sin Alarmduino⁶. Hvis det rigtige brugernavn og adgangskode bliver indtastet, gemmer koden at man er blevet logget ind. Dvs. at man ikke kan tilgå de undersider der er på hjemmesiden før man har været igennem valideringens fasen. Når man så har været igennem den, og er blevet godkendt, så har brugeren fuld adgang til kontrolpanelet over alarmen. Man har mulighed for at oprette nye brugere, slette brugere, se status på alarmen, se en detaljeret log over hvornår alarmen er blevet brudt og tænde / slukke for alarmen.

Krav til loginsystemet

- Oprettelse af bruger
 - Form for kryptering af brugerens adgangskode. (evt. Sha1, Md5 eller andre algoritmer)
 - Der skal være en "salt", der skal lægges til brugerens adgangskode
 - To "salts", én i php-koden, én i databasen.
 - "Salten" skal være unik for hver enkel bruger, og være autogenereret.
 - Man skal være registeret på systemet for at kunne lave nye brugere.
- Hvis brugeren ikke er registeret på databasen skal den blive smidt tilbage på login-siden.
- Man skal ikke kunne tilgå siderne før man er logget ind.

Struktur på siderne

På figur 72 ses et overordnet billede af strukturen på websitet. Man ser login panelet øverst, og man kan altså først tilgå undersiderne, når man har været igennem valideringsfasen på den side.



Figur 72: Illustration af website-strukturen for kontrolpanelet.

⁶ For at logge ind, brug da: brugernavn: 15_c_el_glen0930 og kode: ub4h3

Database

Databasestrukturen indeholder 5 tabeller, *Alarm_status* der skal holde styr på, hvilken tilstand alarmeren er i, samt hvilket tidspunkt alarmeren er blevet tændt. *Log*, der indeholder alle de gange hvor alarmeren er blevet brudt. Den får opdateringer fra Arduinoen. *Log_status* holder øje med hvilke tilstande sensorerne er i. *Pictures* indeholder billede navnene på de filer der bliver uploadet til FTP'en, samt hvornår de enkelte billeder er blevet uploadet. Den sidste tabel i databasen er *members*, der skal holde styr på hvem der er oprettet på systemet, hvad de hedder, samt deres krypterede loginoplysninger. Strukturen for databasen kan ses på figur 73.



Tabel	Handling	Rækker	Datatype	Tegnsæt (sortering)	Størrelse	Overhead
alarm_status	Vis Struktur Søg Indsæt Tøm Slet	~1	InnoDB	utf8_general_ci	32 K.B	-
log	Vis Struktur Søg Indsæt Tøm Slet	~46	InnoDB	utf8_general_ci	16 K.B	-
log_status	Vis Struktur Søg Indsæt Tøm Slet	~4	InnoDB	utf8_general_ci	16 K.B	-
members	Vis Struktur Søg Indsæt Tøm Slet	~4	InnoDB	utf8_general_ci	16 K.B	-
pictures	Vis Struktur Søg Indsæt Tøm Slet	~2	InnoDB	utf8_general_ci	16 K.B	-
5 tabeller	Sum	57	InnoDB	utf8_general_ci	96 K.B	0 B

Figur 73: Illustration af tabellerne på databasen.

Alarm_status

På figur 74 ses et udsnit af strukturen af tabellen Alarm_status. Denne variable kan skifte mellem 0 og 1. Hvor 0 er hvis alarmeren er slukket, og 1 er hvis alarmsystemet er tændt. Denne status kan ændres enten fra arduinoen selv ved at indtaste den korrekte kode, eller på hjemmesiden "login.php". Dette giver brugeren mulighed for at være i stand til at styre sin alarm, selvom personen ikke er hjemme. Eller hvis den pågældende person er på vej ud af huset, kan tænde for alarmeren. Når alarmeren tændes, så sættes der også et unix timestamp ind, således at systemet kan sammenligne hvornår alarmeren er blevet tændt, så plantegningens lamper kan være korrekte.

status	tidspunkt
1	1461668569

Figur 74: Tabel for status på alarmeren

Log

Figur 75 er et udsnit af tabellen log, der har informationer om hvor, samt hvornår der er sket et brud på alarmen. Der er blevet valgt, at datatypen datetime (11) skal bruges, da den bruger den nuværende tid for nyeste opdatering. Status_id bruges til at kontrollere lamperne på plantegningen. Tid er fra når arduinoen uploader til databasen, så sættes der også et timestamp derfra, således der kan sammenlignes om tiden der står på databasen er større end den tid fra den anden database, der indeholder timestamp for hvornår alarmen er blevet tændt. Den er skrevet som en unix kode, hvilket er defineret som sekunder siden d. 1. januar 1970.

Location	status_id	Datetime	Tid
Dør	3	2016-04-15 14:10:22	1460722222
Vindue 1	1	2016-04-15 14:11:34	1460722294
Vindue 2	2	2016-04-15 14:14:59	1460722499
sensor	4	2016-04-15 14:15:01	1460722501
sensor	4	2016-04-15 14:17:09	1460722629
sensor	4	2016-04-15 14:17:10	1460722630
sensor	4	2016-04-15 14:17:10	1460722630

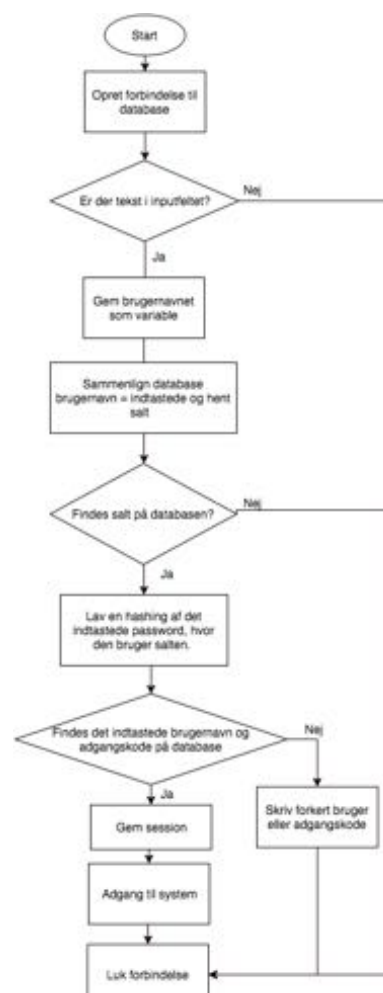
Figur 75: Viser log over brud på alarmen

Index



Figur 76: Udsnit af index.php

På selve loginsystemet er det vigtigt at tjekke, om brugeren overhovedet findes på databasen, samt om den adgangskode som brugeren indtaster matcher med den der står på databasen. Her er det lidt kompliceret, for den adgangskode der står på databasen er en krypteret kode, som ikke direkte kan sammenlignes. Man er derfor nødt til at hente den 'salt' der står ud for brugeren, og derved anvende den til at hashe det indtastede password, hvorefter den sammenligner den med det som står på databasen. Denne valideringsproces gør, at man ikke kan gætte sig direkte til adgangskoden, da man skal kunne kende den salt der står på databasen. Figur 77 viser flowchartet for loginsystemet. I bilag 9 kan den fungerende kode ses for login systemet.



Figur 77: Flowchart for loginsystemet

Opret bruger

The image shows a web form titled 'Kontrolcenter' on a light brown background. In the top left corner, there is a small link that says 'Tilbage' with a left-pointing arrow. The main heading of the form is 'Register bruger på alarmsystemet:'. Below this heading, there are five input fields, each preceded by a label: 'Brugernavn:', 'Fornavn:', 'Efternavn:', 'Adgangskode:', and 'Adgangskode igen:'. At the bottom of the form, there is a button labeled 'Opret!'.

Figur 78: Udsnit af opret.php

For at kunne oprette nye brugere på loginsystemet kræves det, at man allerede har adgang til sitet. Dette gør, at folk udefra ikke kan få adgang til alarmen. Det er lavet således, at hvis man eksempelvis skal på ferie, og gerne vil have naboen til at tjekke op på ens hus, så kan en bruger der allerede har adgang til systemet tilføje naboen således at han kan få adgang til systemet.

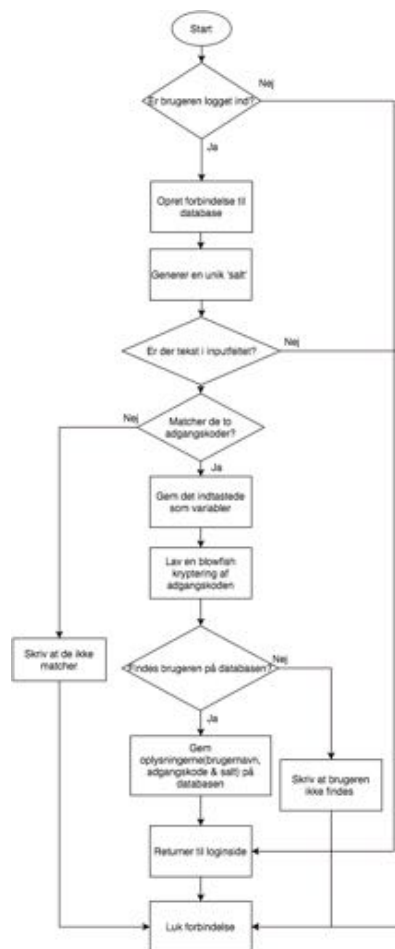
For at oprette en ny bruger, kræves det at brugeren skal indtaste et brugernavn, fornavn, efternavn og adgangskode. Brugernavnet er det som skal bruges når brugeren vil tilgå sitet, samt adgangskoden. Når brugeren er oprettet, vil man blive smidt tilbage til login sitet. Når brugeren så logger ind på kontrolpanelet, vil han/hun blive mødt af en velkomst, hvor fornavnet vil indgå, som på figur 79.

Du er nu logget ind på systemet: Glenn

Figur 79: Velkomstmeddelelse

For at uploade til databasen anvendes 'INSERT' metoden (12). Her indsættes fem parametre, idet både brugernavn, adgangskode, salt, fornavn og efternavn skal gemmes. Dette gør, at der bliver oprettet en ny række på tabellen i databasen, således at systemet kan aflæse de oplysninger for den enkelte bruger.

Figur 80 viser flowchartet for programmet, der skal oprette brugeren. Først generere den altså denne unikke salt for brugeren, og tjekker om der faktisk er noget indtastet i feltet. Hvis der ikke er, så skal databaseforbindelsen lukke. Hvis der er, så skal den gemme det som brugeren har indtastet som en variable, og lave en kryptering af den indtastede adgangskode. Efterfølgende tjekker den, om brugeren allerede findes på databasen, hvis den gør, så skal den skrive til siden, at brugeren allerede findes, og lukke forbindelsen. Hvis den ikke gør, så gemmer den oplysningerne på databasen, og smider brugeren tilbage til index.php.



Figur 80: Flowchart for oprettelse af bruger

```

require 'databaseconnect.php'; //Php-koden kan kun køre, hvis databaseconnect-filen er inkluderet.
$salt = strtr(base64_encode(mcrypt_create_iv(16, MCRYPT_DEV_URANDOM)), '+', '.'); //Genererer en tilfældig værdi, som skal lægges til passwordet der bliver hashet.
$value = 10; //Sætter en værdi.
$salt = sprintf("%020d", $value) . $salt; //laver oplysninger om hashen, så Php kan bekræfte den senere. " $ 2a $ " betyder, at vi bruger en algoritme der hedder blowfish.
if (isset($_POST['user']) && isset($_POST['pass'])) { //Hvis det indtastede i user og password feltet så,
    $user = $_POST['user']; //Set variable med det indtastede
    $pass = $_POST['pass']; //Set variable med det indtastede
    $fornavn = $_POST['fornavn'];
    $efternavn = $_POST['efternavn'];
    $pass2 = $_POST['pass2'];
    $hash = crypt($pass, $salt); //laver et hash af det indtastede password hvor den bruger salten fra tidligere igennem blowfish algoritmen.

    $query = mysql_query($mysqlConnection, "SELECT * FROM members WHERE username='$user'"); //Laver en query på brugernavn, for at tjekke om den findes.
    if (mysql_num_rows($query) > 0) { //Hvis brugeren allerede findes på databasen:
        echo "<script type='text/javascript'>alert('Brugeren findes allerede');</script>";
    } else { //Hvis ikke, så
        mysql_query($mysqlConnection, "INSERT INTO members (username, password,salt,fornavn,efternavn) VALUES ('$user', '$hash', '$salt', '$fornavn', '$efternavn')");
        //lav en ny bruger på databasen, med værdierne der er indtastede, dog bruger den det hashede password.
        header("location:index.php"); //Redirecter til index siden.
    }
}
mysql_close($mysqlConnection);
?>

```

Figur 81: Udsnit af koden; oprettelse af bruger

Koden på figur 81, viser hvordan håndteringen af nye brugere bliver udført. Koden anvender 'POST' metoden vha. http (Hypertext transfer protocol) til at kunne kommunikere med serveren og hente det som brugeren har indtastet i feltet. Til krypteringen af adgangskoden bruges 'crypt' funktionen (13), der kan anvende forskellige algoritmer til at hashe med. Det eneste den kræver er at have en streng for at kunne hashe. Dog kan den

også anvende en 'salt', men dette er kun hvis programmøren ønsker det. Der anvendes 'salt' til kryptering af brugerens adgangskode på siden. I bilag 14 kan den fungerende kode ses for oprettelse af bruger.

Kontrolpanel



Figur 82: Udsnit af alarm slukket



Figur 83: Udsnit af alarm tændt

Kontrolpanel (Login.php) skal sørge for, at man kan holde øje med hvilken tilstand alar-men er i. Derudover skal man kunne ændre alarmens tilstand, således at man ikke behø- ver være i hjemmet for at tænde / slukke for sin alarm. Siden anvender POST-metoden til at sende informationer til databasen. Den fungerer også som et samlepunkt for alle un- dersiderne. Det er vigtigt, at man ikke kan tilgå denne side, før man har været igennem login-fasen. Ellers vil uvedkommende kunne styre tilstanden for alarmen, hvilket vil være problematisk. For at holde styr på, om brugeren er logget ind, anvendes 'Sessions' (14), der er en php metode. På undersiderne tjekkes der ligeledes om brugeren er logget ind, på samme metode.

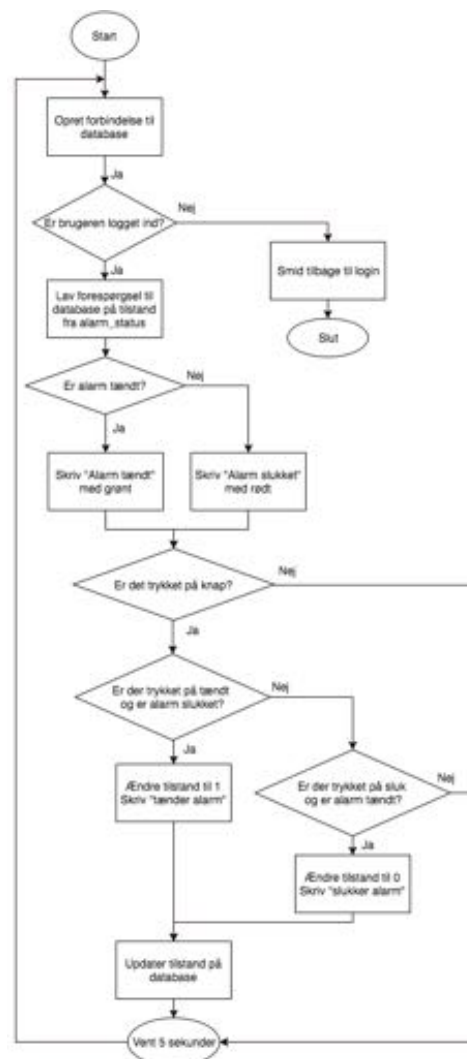
```
session_start();  
if($_SESSION["Login"] != "YES"){  
    header("Location: index.php");  
}
```

Figur 84: Anvendelse af Sessions

Hvis sessionen ikke er lig YES, så smider den brugeren tilbage til login siden. Dette ko- destykke anvendes på alle sider, hvor det kræves at man er logget ind.

Det er blevet valgt, at siden skal genindlæses hvert femte sekund. Dette gøres for at hele tiden få den rigtige tilstand for alarmen fra databasen. Dog er der en ulempe ved, at den skal genindlæse hele siden konstant, i stedet for et enkelt element. Det giver ikke særlig god brugervenlighed, da nogle browsere ikke fungerer optimalt med denne løsning.

Det også kun muligt at trykke på ON-knappen hvis alarmen er slukket, og ligeledes hvis OFF-knappen bliver trykket er alarmen tændt. Dette skyldes, at der ikke er grund til at kunne tænde for alarmen, når den allerede er tændt. I bilag 12 kan den fungerende kode ses for kontrolpanelet.



Figur 85: Flowchart for login.php

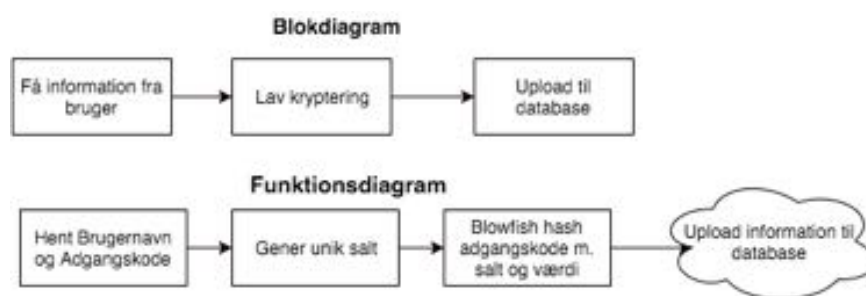
Kryptering

Langt største delen, hvis ikke alle IT-firmaer bruger dagligt kryptering til at sikre deres fortrolige oplysninger, men hvor sikre er disse systemer overhovedet, og hvorfor anvendes disse teknikker? Krypteringen handler basalt set om en række matematiske teknikker, som kan forme algoritmer. Disse algoritmer kan lave tal, eller sågar, tekst om til en masse uforståelige tegn, som ikke kan læses af den almindelige bruger. Dog er computere begyndt at blive så avancerede og har en processorkraft der uden problemer kan bryde mange af de kendte krypteringsmetoder. Dette kan blive problematisk for de persondata som et firma kan have på en pågældende person. For eksempel var der i 2012 nogle svenske hackere der brød ind på det danske CPR-register (15), og fik fat i mange fortrolige oplysninger om danskerne.

Det er vigtigt, når man laver et login system, at brugere der ikke har adgang til alarmen, kan bryde igennem sikkerhedssystemet, og få informationer omkring systemet. Derfor er

det ikke nok at gemme brugerens adgangskode i en database som ren tekst. Man er nødt til at lave en kryptering af adgangskoden. Til dette system er der blevet valgt, at lave en blowfish kryptering på adgangskoden. Dvs. at man først generer en unik 'salt', som der- efter bruges i addition af ens hash.

Når brugerens information bliver uploadet til den database, der skal holde styr på hvem der er oprettet, og hvem der ikke er oprettet, bliver deres unikke 'salt' også uploadet som tilføjelse af deres id.



Figur 86: Krypteringsfasen.

Figur 87 viser database strukturen, hvor man kan se, at hver enkel bruger får deres eget id, hvor man kan se brugernavnet, det krypterede adgangskode, salt, fornavnet og efternavnet på brugeren. Fornavnet bliver brugt til at lave en velkomstmeddelelse på kontrolpanelet.

id	username	password	salt	fornavn	efternavn
45	15_c_el_glen0930	\$2a\$10\$BtNlu2K89cYWqcVXIS64eu5Jwbc1A/xAzvXVZ4F9QkqO1.m.q2s5a	\$2a\$10\$BtNlu2K89cYWqcVXIS64ew==	Glenn	Herping
46	chri808q	\$2a\$10\$XzWQkfUfhUu17P4dHQL1xuGicINpC.Eqqar6FAQFbMedUoyYS5tdG	\$2a\$10\$XzWQkfUfhUu17P4dHQL1xw==	Christian	Henriksen
47	Master_alarm1	\$2a\$10\$nnx/bbxW/ebCkS0c3L7NtuKx.vbYeYBq0mWmLmm21LftgE35fB9C	\$2a\$10\$nnx/bbxW/ebCkS0c3L7Ntw==	Master_alarm1	Master_alarm1
48	Master_alarm2	\$2a\$10\$bG5lQXXKEIEZzqxWcUHy5OVrrJMcncXoAj2gV.uRqg5oIdcAQzbcC	\$2a\$10\$bG5lQXXKEIEZzqxWcUHy5Q==	Master_alarm2	Master_alarm2

Figur 87: Udsnit af tabellen members

Forklaring på blowfish algoritmen

Blowfish algoritmen er en blok kode, der kan tage op til 56 bytes (448 bits) og kryptere dem. Den gør dette ved at opdele i blokke af 64 bits, der igen opdeles i venstre og højre, som den gennemløber 16 gange med en XOR-funktion. Der er fem subkeys: et 18 element P-array og fire 256 element S-bokse.⁷

Hver runde består af fire handler. Den første handling er, at den tager venstre del og XOR med det P'te element den er kommet til. Dernæst bruger den XOR'ed venstre del som input for Blowfishs F-funktion og XOR det returnerede med den højre del. Efterfølgende

⁷ En S-boks er en substitutions-boks der udskifter nogle bits til bits der der tæt på, men ikke nødvendigvis magen til den oprindelige bit.

XOR den igen højre side, men denne gang med næstfølgende P'te element. Sidst bruger den den højre del som input i F-funktionen og XOR den med venstre side. Når den har gjort dette 8 gange (16 XOR runder) XOR den venstre del med P'te 17'ende element og den højre del med P'te 18'ende element. Sidst bytter den rundt på venstre og højre del. Kodemæssigt ser det ud som følgende: (16)

```
void encrypt (uint32_t & L, uint32_t & R) {  
    for (int i=0 ; i<16 ; i += 2) {  
        L ^= P[i];  
        R ^= f(L);  
        R ^= P[i+1];  
        L ^= f(R);  
    }  
    L ^= P[16];  
    R ^= P[17];  
    swap (L, R);  
}
```

Den før omtalte F-funktion kan ses under. F-funktionens formål er, at flytte rundt på placeringerne af 0 og 1 i hver bit, med bitwise metoden. (16)

```
uint32_t f (uint32_t x) {  
    uint32_t h = S[0][x >> 24] + S[1][x >> 16 & 0xff];  
    return ( h ^ S[2][x >> 8 & 0xff] ) + S[3][x & 0xff];  
}
```

Eksempel på XOR

Lad os sige, at vi har decimal 109 der i binary er givet som: 01101101, som værende den venstre side, og skal XOR med P[0]: 10110100⁸ der er decimal 180 fås følgende:

$$\begin{aligned} &01101101 \text{ XOR } 10110100 \\ &= 11011001 \end{aligned}$$

Rediger bruger

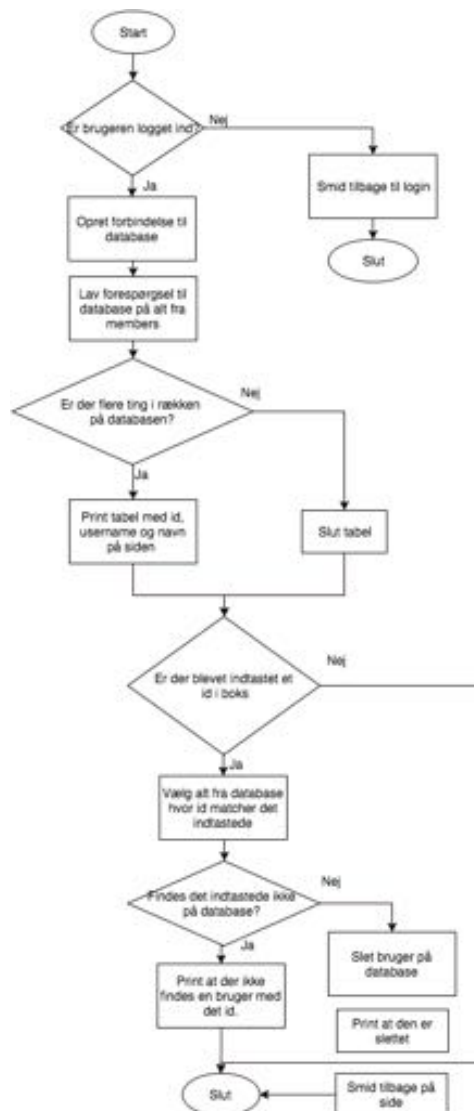


Figur 88: Udsnit af rediger.php

På systemet er det muligt at redigere de brugere der er oprettet på systemet. Dog er det lavet således, at der kun er mulighed for at slette brugere på systemet indtil videre. Der er en detaljeret liste over de brugere der er registreret på systemet, med ID, Brugernavn,

⁸ Ikke nødvendigvis rigtig, da værdien for P[0] ikke vides.

fornavn & efternavn. Hvis det ønskes at fjerne en bruger, aflæses ID'et på den pågældende bruger, og indtastes i feltet, hvorefter der trykkes på 'fjern bruger'-knappen. Det er også på denne side, man har mulighed for at tilgå undersiden for at oprette en ny bruger.



Figur 89: Flowchart for rediger.php

I bilag 13 kan den fungerende kode for redigeringen af brugerne ses.

Database connect

I stedet for at have gentagende linjer kode, der sørger for at skulle connecte til databasen, er der skrevet filen, databaseconnect.php, som gør det samme, som der så i stedet henvises til i de andre filer. På figur 90 ses et udsnit af koden der tilslutter til databasen.


```
<?php
$host = "localhost"; // Host navn
$username = "15_c_el_glen0930"; // Mysql brugernavn
$password = ""; // Mysql password
$db_name = "15_c_el_glen0930"; // Database navn
$table_name = "members"; // Tabel-navn
$mysqlConnection = mysqli_connect($host, $username, $password); //connect til databasen
if (!$mysqlConnection){ //Hvis den ikke kan etablere en forbindelse til databasen, så skal den printe:
    echo "<script type='text/javascript'>alert('Kan ikke skabe en sikker forbindelse til databasen');</script>";
} else { //Ellers, hvis den har skabt forbindelse, så skal den printe:
    mysqli_select_db($mysqlConnection,$db_name);
}
mysqli_report(MYSQLI_REPORT_ERROR);
?>
```

Figur 90: Kode fra databaseconnect.php

Hvis ikke den kan tilkoble sig databasen, kommer den med en fejlmeddelelse. Der er også blevet indsat funktionen 'mysqli_report' (17), der sørger for at komme med fejlmeddelelser på siden, hvis der er evt. syntaxfejl mm.

Hvis man inkludere denne fil på de andre sider medfører det, at man ikke behøver skrive den samme kode flere gange. Dog anvendes 'require' (18) funktionen, således at det de andre filer ikke kan åbnes hvis ikke den kan tilslutte sig databasen.

```
require 'databaseconnect.php';
```

Figur 91: Anvendelse af 'require' funktionen

Log ud

Idet der bliver anvendt sessions til at holde styr på om brugeren rent faktisk er logget ind på vores system, så er det eneste der skal gøres for at logge ud, er at slette den session som browseren har. Sessionen forbliver aktiv indtil browseren bliver lukket ned, eller man logger ud.

```
<?php
session_start(); // Begynder session
session_unset(); // Uddeklarer alle sessioner
session_destroy(); // Sletter session
header( "refresh:2;url=index.php" );
?>
```

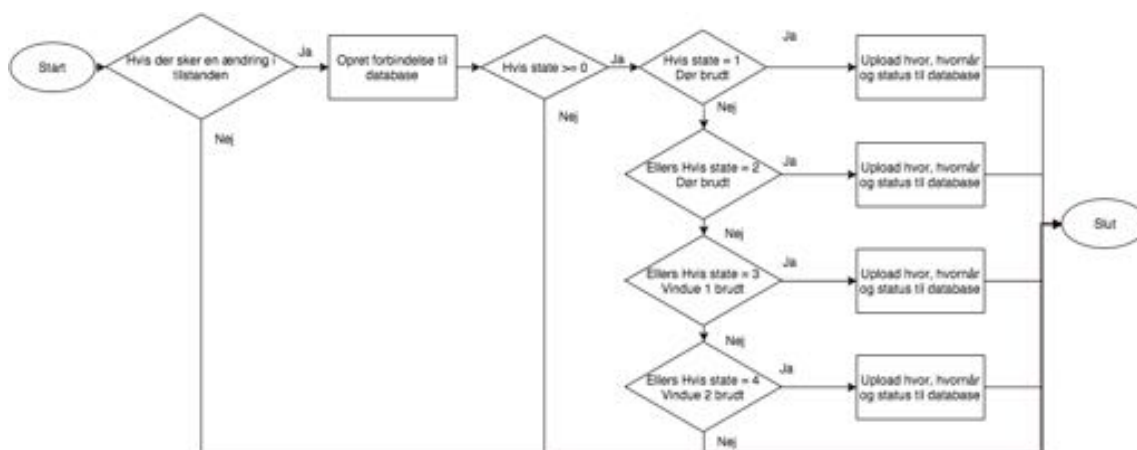
Figur 92: Udsnit af kode der logger brugeren ud

Sammenspil med Arduino

For at få arduinoen til at kommunikere med vores webbaserede kontrolpanel, kræves det anvendelse af GET-metoden fra php. Denne GET-metode bliver anvendt i Arduino funktionerne; PrepareAlarm, UpdateAlarmStatus, UpdateLog og UpdatefileName.

Update log

Update log filen, sørger for at opdatere tilstanden på hvor der har været et brud på alarmen. De informationer der bliver opdateret er lokation og tiden for bruddet. Den får tilstandene fra Arduinoen, som hhv. kan være 1, 2, 3 og 4. Hvis tilstanden er 1, betyder det at døren er brudt, 2 vindue 1 er brudt, 3 vindue 2 er brudt og 4 er PIR-sensoren er brudt.



Figur 93: Flowchart for Update_log.php

Koden holder øje med om tilstanden fra Arduinoen er større end 0, såfremt det er sandt gennemløbes en 'if-statement', hvor, der alt efter hvilken tilstand Arduinoen returnerer, så uploader den informationerne til databasen. Hvis ikke den får en tilstand der er over 0, så stopper processen.

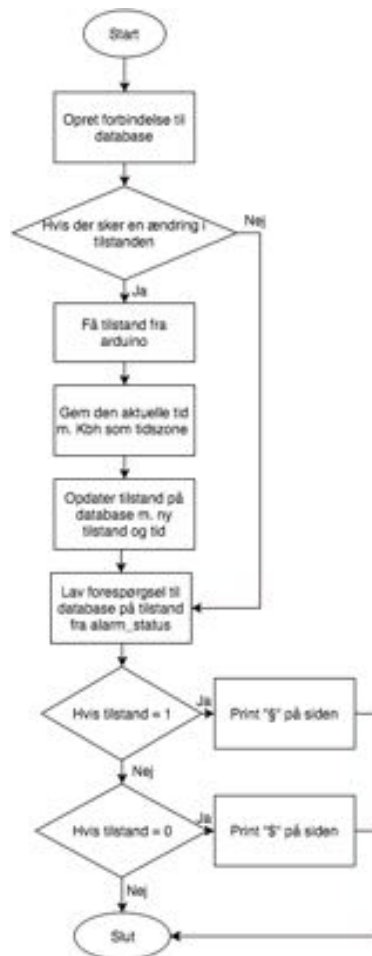
```
$state = $_GET['state']; //Andre værdien 'tjek'
if ($state >= 0) {
    if ($state == 1) { //Hvis der er brudt,
        $stateinsert = "Dør";
        mysql_query(mysqlConnection,"INSERT INTO log (Location,Tid, status_id) VALUES ('$stateinsert','$tid', 3)"); //Indsæt Lokation, tid og status
        mysql_query(mysqlConnection,"UPDATE log_status SET log_status = '1' WHERE sted='Dør'"); //Updater log_status med 1 ved døren
    } else if ($state == 2) { //Vindue1
        $stateinsert = "Vindue 1";
        mysql_query(mysqlConnection,"INSERT INTO log (Location,Tid,status_id) VALUES ('$stateinsert','$tid',1)"); //Indsæt Lokation, tid og status
        mysql_query(mysqlConnection,"UPDATE log_status SET log_status = '1' WHERE sted='Window_one'"); //Updater log_status med 1 ved vindue 1
    } else if ($state == 3) { //Vindue2
        $stateinsert = "Vindue 2";
        mysql_query(mysqlConnection,"INSERT INTO log (Location,Tid,status_id) VALUES ('$stateinsert','$tid',2)"); //Indsæt Lokation, tid og status
        mysql_query(mysqlConnection,"UPDATE log_status SET log_status = '1' WHERE sted='Window_two'"); //Updater log_status med 1 ved vindue 2
    } else if ($state == 4) { //Sensor
        $stateinsert = "Sensor";
        mysql_query(mysqlConnection,"INSERT INTO log (Location,Tid,status_id) VALUES ('$stateinsert','$tid',4)"); //Indsæt Lokation, tid og status
        mysql_query(mysqlConnection,"UPDATE log_status SET log_status = '1' WHERE sted='Sensor'"); //Updater log_status med 1 ved Sensor.
    }
}
```

Figur 94: Udsnit af kode; Udatering af log

I bilag 16 kan den fungerende kode ses for opdatering af loggen.

Read Alarm

Read alarm filen har til formål at læse hvilken tilstand alarmen er i, og smide den op på databasen. Tilstanden kan enten være 0, som repræsenterer at alarmen er slukket, og 1 så er alarmen tændt. Derudover printer den også den nuværende status på siden. Dog skriver den hhv. '\$' og '\$' alt efter hvilken tilstand der er gældende, således at Arduinoen kan læse det korrekt. Det gøres idet, at tilstanden for alarmen også kan ændres fra det web-baserede kontrolpanel og fra Arduinoen.



Figur 95: Flowchart for Read_alarm.php

Figur 95 viser et flowchart for sammenspillet mellem arduinoen og databasen. Her er det tilstanden og tiden for alarmerne der bliver opdateret. Den tjekker også, om tilstanden fra databasen ændre sig, og hvis den så gør det, ændres det symbol som læses af arduinoen, og det ville kunne ses på arduinoen.

Alarm log



Figur 96: Alarm_log.php; Status: tændt



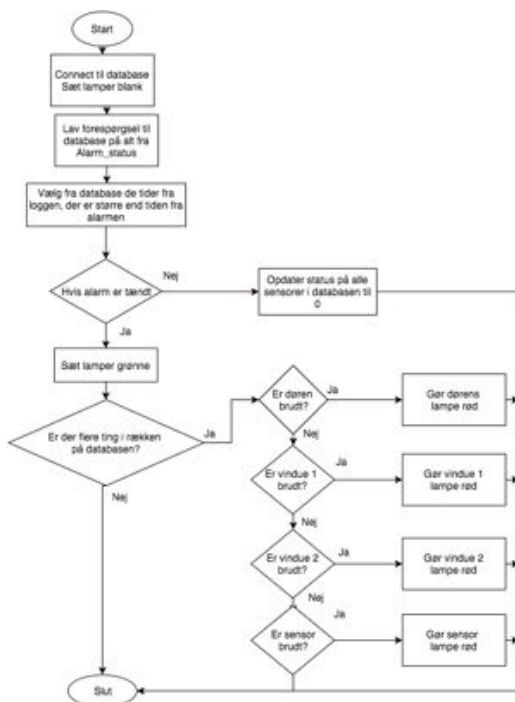
Figur 97: Alarm_log.php; Status: brudt

Siden Alarm_log.php er den sider, der viser loggen over alle sensorerne. På siden er der placeret en stor plantegning (19) for et hus. Oven på den er der anbragt fire lamper (20), som repræsenterer de fire sensorer, der er tilsluttet det pågældende alarmsystem. Når alarmen er slukket, hvilket betyder at tilstanden er 0 på databasen, så er lamperne illustreret som blanke. Med det samme tilstanden ændre sig til 1, som indikere alarmen er tændt, bliver de omtalte lamper grønne. Idet, at der på databasen bliver ændret i tabellen 'log_status', således at der er sket et brud, bliver lampen på det sted bruddet er sket rød. Se figur 97.

```
if ($row['status'] == 1){ // hvis alarm er tændt:  
/*Lampe;grønfarve*/  
$lightbulb_door = "pics/Lightbulbgreen.png";  
$lightbulb_window_one = "pics/Lightbulbgreen.png";  
$lightbulb_window_two = "pics/Lightbulbgreen.png";  
$lightbulb_sensor = "pics/Lightbulbgreen.png";  
  
while ($row_log = mysqli_fetch_assoc($result_log)) {  
    if($row_log['status_id'] == 1){ //Hvis vindue1 er brudt  
        $lightbulb_window_one = "pics/lightbulbred.png"; //Gør lampe rød  
    } else if ($row_log['status_id'] == 2){ //Ellers hvis vindue2 er brudt  
        $lightbulb_window_two = "pics/lightbulbred.png"; //Gør lampe rød  
    } else if ($row_log['status_id'] == 3){ //Ellers hvis dør er brudt  
        $lightbulb_door = "pics/lightbulbred.png"; //Gør lampe rød  
    } else if ($row_log['status_id'] == 4){ //Ellers hvis sensor er brudt  
        $lightbulb_sensor = "pics/lightbulbred.png"; //Gør lampe rød  
    }  
}
```

Figur 98: Udsnit af kode; Lamper

Lige meget hvilken tilstand lamperne er i, kan man ved at klikke på dem, få en oversigt over hvornår der sidst har været et brud på det pågældende sted. Se figur 98.



Figur 99: Flowchart for Alarm_log.php



Figur 100: Flowchart 2 for Alarm_log.php

Figur 100 viser et flowchart for den funktion der viser tidsloggen for de enkelte steder. Den tjekker, hvilke lamper der bliver trykket på, og alt efter hvilken der bliver trykket på, så henter den tiden fra tabellen 'log' på databasen. I dette tilfælde, er det blevet valgt at anvende "case-metoden", (21) så koden er i stand til at tjekke hvilke objekter der er blevet valgt.

```

switch ($log_state) {
    case "1": //Hvis døren er valgt
        $log_query = "SELECT Tid FROM log WHERE status_id = '3'";
        $result_door = mysqli_query($mysqlConnection, $log_query); //Laver forespørgslen.
        /*Laver en tabel. */
        echo '<table class="log_time">';
        echo '<tr><th class="log"> Tidspunkt for brud for døren: </th></tr>';
        while ($row = mysqli_fetch_array($result_door)) { //Så længe at $row er ligmed et resultat række som et array, så
            echo '<center>';
            echo '<tr><td class="log">';
            $tid = date("m/d/Y H:i:s", $row['Tid']); //Laver tidsformatering
            echo $tid; //Indsætter tiden i tabellen
        }
        echo '</table>';
        break;
}

```

Figur 101: Udsnit af kode; Switchstatement

I bilag 17 kan den fungerende kode for alarm_log ses.

Camera log

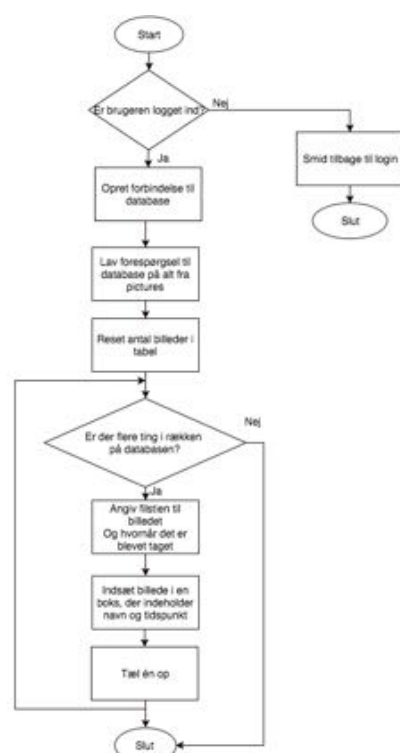
Camera_log.php har til funktion at uploade de billeder der ligger på serveren til siden. Dog skal de filer håndteres korrekt. For der kommer hele tiden nye filer til, så det kræves det at man har en database, der opbevarer filnavnet på de filer der skal uploades. Derudover, så holder den også øje med hvornår billedet er taget, og hvis man klikker på et af billederne, så kommer det op i en større version med tidsnotering for hvornår det er taget.

Til indsættelse af billederne, bruges variablen 'Source' til at finde den rigtige fil som skal anvendes. Den tager det filnavn der står på databasen, og indsætter i filstien på serveren, således at HTML-koden kan håndtere billedet korrekt. Dog skal den hente alle billeder der skulle ligge på serveren, så derfor er der blevet anvendt en 'while-løkke' (22), for at kunne hente alle filnavne fra databasen. Se figur 102 og 103.

```
$i=0;
while($row = mysqli_fetch_array($result_pic)) {
    /*Sætter filstien til billedet ud fra filnavnet på database*/
    $source = "http://htx-elev.ucholstebro.dk/HX-15-c-el/glen9930/Doorbell_IMG/" . $row['fileName'];
    /*Tidsformatering*/
    date_default_timezone_set("Europe/Copenhagen");
    $tid = date("m/d/y H:i:s", $row['time']);
    echo "<div style='display: inline-block;'>"; //Laver div
    echo "<img src='".$source.">"; //Skriver tiden for det pågældende billede
    echo "<br>";

    /*Laver billede med link, så når man trykker på den kommer der et overlay*/
    echo "<a href='#popup'." . $i . "><img class='camera_log' src='".$source.">/></a>";
    echo "</div>";
    /*Laver overlay der er skjult, indeholder stor version af billede samt tiden*/
    echo "<div id='popup'." . $i . " class='overlay'>";
    echo "<div class='popup'>";
    echo "<a class='close' href='#'>&times;</a>";
    echo "<div class='content'>";
    echo "<img src='".$source.">";
    echo "<p>Tidspunkt billedet er taget: '".$tid."</p>";
    echo "</div>";
    echo "</div>";
    $i++; //Lægger én til
}
```

Figur 103: Udsnit af kode; while loop



Figur 102: Flowchart for Camera_log.php

Persondataloven

I forhold til persondataloven § 6 stk. 1 må man ikke, uden samtykke, offentliggøre portrætbilleder af andre på internettet. (23) Der er selvfølgelig undtagelser, men som udgangspunkt for at offentliggøre portrætbilleder skal man have samtykke fra personen. Dette vil kunne være et problem for Alarmduino, idet den vil skulle kunne tage billeder af personer der ringer på ens dør. Når det sker bliver billedet lagt op på en offentlig FTP server, hvor har man den rigtige adresse kan tilgå billederne direkte. En løsning på dette kunne være, at man havde billederne til at ligge privat, og kun når siden blev indlæst kan man tilgå billederne, og dermed slettes fra den offentlige del igen når siden forlades. Dette vil ikke fuldstændigt gøre det lovligt, men det vil gøre det mere sikkert for den portrætterede, da det kun vil være i få øjeblikke, at billederne faktisk ligger offentlige.

Videreudvikling

Der er selvfølgelig plads til forbedring af systemet. I det følgende vil der beskrives nogle overvejelser for, hvad disse forbedringer kunne være. Først nogle overordnet punkter, og dernæst videreudvikling for hjemmesiden.

Det største mangler er nok, at systemet ikke kan fungere, hvis det er offline fra netværket. I sådanne tilfælde vil man ikke kunne uploade billeder, opdatere loggene, slå alarmen til eller fra udefra. Dermed vil man ikke kunne have et klart overblik over alarmen gennem hjemmesiden, da den ikke er opdateret. Dog vil man godt kunne opfange, at nogen har brudt ind, da man får en SMS, hvis nogen bryder ind. Implementerede man en offline mode, ville man kunne opbevare data fra alarmen på SD kortet, og når der på et tidspunkt igen er forbindelse til internettet ville man kunne uploade disse data – eller helt tage SD kortet ud af systemet og læse direkte derfra.

Derudover kunne man producere systemet således, at de forskellige moduler fx sensor og kontakter kunne sælges separat, og dermed have mulighed for selv, at vælge hvor mange sensorer man skal have i huset. Huset er jo forskelligt bygget og der vil ikke være megen mening i en alarm der ikke kan holde øje med alle vinduer og døre. For at dette kan virke vil man samtidigt skulle have mulighed for at fortælle systemet, at man vil tilkoble et modul mere. Det kunne for eksempel gøres gennem internettet, hvor man vælger de forskellige moduler og styre dem.

Dertil kunne man gøre disse moduler trådløse, så der ikke skal trækkes ledninger rundt i hele huset for eksempel ved, at hvert modul havde sin egen internet forbindelse over Wi-Fi eller med Bluetooth.

Derudover kunne man tilkoble overvågningskameraer i forskellige rum, som begynder at optage når der sker et brud på alarmen. På den måde vil man kunne følge gerningsmanden rundt og samtidigt se hvad de går efter. Dette vil selvfølgelig også skulle uploades til databasen og også være muligt at hente ned, så man kan videregive det til politiet i en eventuel sag.

Desuden vil det være godt, hvis man kunne styre alarmen mere. Dertil tænkes der de forskellige sensorer og knapper. For eksempel vil det ikke være rart hvis der er 30 grader en varm sommernat, og ikke have mulighed for at åbne et vindue, da alarmen vil tolke dette som brudt.

Det er måske set, at den kode man bruger til at tænde og slukke alarmen via tastaturet er prædefineret. Kunne man selv ændre kode ville det gøre systemet mere sikkert, da

man dermed ikke vil have en kode der er ens for alle alarmsystemer – hvis det blev masseproduceret.

Alarmduino er tilsluttet en 9V adapter, der sættes i en stikkontakt. Dette kan være problematisk, idet, at hvis der sker en strømafbrydelse i huset, vil alarmsystemet slukke. Der vil i realiteten kunne ske indbrud, uden at alarmen opfanger det under strømsvigt. Det kan derfor udvikles en form for batteri-backup der automatisk tilsluttes, hvis denne adapter pludseligt ikke giver forsyning til systemet længere. Dette vil kunne være noget så simpelt som et 9V batteri, samt et lille kredsløb der holder øje med adapteren.

Som Alarmduino er opbygget nu, kan flere alarmer ikke være brudt samtidigt. Dette skyldes, at Arduinoen under et brud går i tilstand 3, hvilket den ikke har mulighed for at gøre gentagende gange. På hjemmesiden er det muligt, at på plantegningen kan der ske flere brud, dermed ligger bristet hovedsageligt i softwaren på Arduinoen.

Videreudvikling af hjemmeside

For det online kontrolpanel er der gjort overvejelser for, hvad der kunne forbedre systemet. Disse er beskrevet i det følgende.

- Administratoren på sitet, skal have mulighed for at deklarere rettigheder til forskellige brugere. F.eks. skal det ikke være muligt, at alle brugere kan oprette nye brugere. Det kan også være, at naboen kun skal kunne se, om alarmen er gået, men ikke kunne slukke for den.
- Mulighed for, når man opretter bruger, at tilmelde sig systemets SMS-service. Det kan eksempelvis være, hvis administratoren på alarmen gerne vil have, at sin nabo også er tilknyttet denne service. Dette kan være af forskellige årsager, f.eks. hvis man ikke har mulighed for at tjekke op på alarmen selv.
- Mere mobilvenlig (Responsive (24)), evt. en separat Application. Dette gør det mere brugervenligt for brugere af alarmen, da de kan tilgå alarmen fra deres smartphone. Dette skal gøres, idet mange altid har deres telefoner på sig, så det er derfor essentielt at have denne egenskab. Det nuværende site har ikke mulighed for at tilpasse størrelsen til mobiler, tablets mm. til optimal opløsning.
- Mulighed for at ændre brugerens oplysninger. Det kan være, at de gerne vil have ændret deres adgangskode til alarmen, eller at efternavnet skal ændres. Noget er mere nødvendigt end andet, men muligheden skal være der. Dertil kunne det også være interessant at kunne deklarere hvilken relation personen er.

- Kontrolpanelet opdaterer kun enkelte elementer i stedet for hele siden. Nuværende kode opdatere hele kontrolpanelet hvert femte sekund. Dette giver ikke særlig brugervenlighed, da mange browsere reagerer designmæssigt på dette.

Konklusion

Ud fra arbejdet med emnet "velfærds-elektronik i hjemmet" og udformningen af alarm-systemet "Alarmduino" kan det konkluderes, at man er kommet frem til et velbesvarende projektresultat – Man har lavet et fungerende alarmsystem, som i store træk opfylder de krav og planer, der blev stillet ved projektets begyndelse. Fagligt og konstruktionsmæssigt er man nået et tilfredsstillende niveau, som indeholder flere kompetencer inden for faget computer- og el-teknik, hvor der ligeledes er inddraget tværfaglighed fra matematik og fysik. Herunder har man anvendt adskillige kompetencer som tidsmæssig planlægning, skitsering, blok- og funktionsdiagrammer, flowcharts, el-diagrammer, programmering, kryptering, print-fremstilling og beregninger på komponentniveau, der indebærer forståelsen af el-tekniske fagligheder som DC- og AC-spænding, komplekse tal, faseforskydning og frekvenser vedr. elektriske filtre.

Alt sammen som en del af den lærerige proces det er, at udføre et projekt som dette - ikke mindst fordi man støder på forskellige problematikker, der åbner op for nye udfordringer og læremuligheder. Når det er sagt, så er resultatet ikke fejlfrit på nogle punkter, og der er naturligvis plads til forbedringer og optimering på flere af projektets dele. Her er konstruktionen af et fungerende kamera, en mere stabil sensor og optimering af software blot nogle eksempler. Alligevel kan man, som tidligere nævnt, sige at man har opnået et tilfredsstillende resultat, der fint illustrere idéen bag sådanne projekter samt den teknologiske udvikling af et produkt som "Alarmduino".

Litteraturliste

1. Holstebro HTX. Switch - Kommunikation-IT Holstebro HTX. [Online].; 2015 [cited 2016 april 6. Available from: <http://htx-elev.ucholstebro.dk/wiki/index.php?title=Switch>.
2. Arduino. Arduino Ethernet Shield. [Online].; 2016 [cited 2016 april 27. Available from: <http://www.arduino.cc/en/Main/ArduinoEthernetShield>.
3. Holstebro HTX. Polling - Kommunikation-IT Holstebro HTX. [Online].; 2014 [cited 2016 april 11. Available from: <http://htx-elev.ucholstebro.dk/wiki/index.php?title=Polling>.
4. Oily The Otter. Arduino forum: Ethernet, connect(), timeout. [Online].; 2010 [cited 2016 april 11. Available from: <http://forum.arduino.cc/index.php?topic=49401.0>.
5. WIZnet CO. Datasheet W5100. [Online].; 2008 [cited 2016 april 11. Available from: https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf.
6. EngineersGarage. AT Commands. [Online].; 2012 [cited 2016 april 25. Available from: <http://www.engineersgarage.com/tutorials/at-commands>.
7. Seedstudio. AT Commands Set. [Online].; 2010 [cited 2016 april 25. Available from: http://garden.seeedstudio.com/images/a/a0/SIM900_ATC_V1_00.pdf.
8. S. Arduino Playground. [Online].; 2015 [cited 2016 april 4. Available from: <http://playground.arduino.cc/Code/FTP>.
9. Wikipedia. [Online].; 2016. Available from: https://en.wikipedia.org/wiki/Passive_infrared_sensor.
10. Murata Manufacturing Co., Ltd. Murata.com. [Online].; 2005. Available from: <http://htx-elev.ucholstebro.dk/el/komponent/IRA-E700.pdf>.
11. Mysql Dev. The DATE, DATETIME, and TIMESTAMP Types. [Online].; 2009 [cited 2016 5 28. Available from: <http://dev.mysql.com/doc/refman/5.7/en/datetime.html>.
12. W3Schools. PHP Insert Data Into MySQL. [Online].; 2016 [cited 2016 5 28. Available from: http://www.w3schools.com/php/php_mysql_insert.asp.
13. W3Schools. PHP crypt() Function. [Online].; 2016 [cited 2016 5 28. Available from: http://www.w3schools.com/php/func_string_crypt.asp.

- 14 W3Schools. PHP 5 Sessions. [Online].; 2016 [cited 2016 5 28. Available from:
. http://www.w3schools.com/php/php_sessions.asp.
- 15 Astrup S. Hackere har angrebet cpr-register. [Online].; 2013 [cited 2016 4 30.
. Available from:
<http://politiken.dk/forbrugogliv/digitalt/internet/ECE1989690/hackere-har-angrebet-cpr-register/>.
- 16 Wikipedia. Wikipedia. [Online].; 2016 [cited 2016 april 30. Available from:
. [https://en.wikipedia.org/wiki/Blowfish_\(cipher\)](https://en.wikipedia.org/wiki/Blowfish_(cipher)).
- 17 Php.net. Mysql Report. [Online].; 2016 [cited 2016 4 28. Available from:
. <http://php.net/manual/en/function.mysql-report.php>.
- 18 Php.net. Require. [Online].; 2016 [cited 2016 4 25. Available from:
. <http://php.net/manual/en/function.require.php>.
- 19 Nordhuse. Plantegninger. [Online].; 2016 [cited 2016 3 18 [Billede]. Available from:
. http://nordhuse.dk/upload_dir/pics/Plantegninger/Mellem/Box-165-1-mellem.jpg.
- 20 Podolsky A. Light Bulb. [Online].; 2016 [cited 2016 5 12 [Billede]. Available from:
. <https://thenounproject.com/term/light-bulb/263994/>.
- 21 W3Schools. PHP 5 switch Statement. [Online].; 2016 [cited 2016 5 28. Available
. from: http://www.w3schools.com/php/php_switch.asp.
- 22 W3Schools. PHP 5 while Loops. [Online].; 2016 [cited 2016 5 28. Available from:
. http://www.w3schools.com/php/php_looping.asp.
- 23 Datatilsynet. Billeder på internettet. [Online].; 2015 [cited 2016 maj 1. Available
. from: <https://www.datatilsynet.dk/borger/internettet/billeder-paa-internettet/>.
- 24 W3Schools. Responsive Web Design - Introduction. [Online].; 2016 [cited 2016 5
. 28. Available from: http://www.w3schools.com/css/css_rwd_intro.asp.
- 25 Seedstudio. GPRS Shield V2.0 - Wiki. [Online].; 2016 [cited 2016 april 6. Available
. from: http://www.seeedstudio.com/wiki/GPRS_Shield_V2.0.
- 26 Holstebro HTX. W5100.png. [Online].; 2016 [cited 2016 april 27. Available from:
. <http://htx-elev.ucholstebro.dk/el/modul/W5100.png>.
- 27 HTX Holstebro. Komponenter og datablade for moduler. [Online].; 2016 [cited 2016
. april 4. Available from: <http://htx-elev.ucholstebro.dk/index.php?p=102>.

- 28 Learnabout Electronics. Learnabout-electronics.org. [Online].; 2016. Available from:
. <http://www.learnabout-electronics.org/Amplifiers/amplifiers62.php>.
- 29 Motorola. motorola.com. [Online].; 1996. Available from: <http://htx-elev.ucholstebro.dk/el/komponent/LM358.pdf>.
- 30 nde-ed.org. [Online]. Available from: <https://www.nde-ed.org/EducationResources/CommunityCollege/EddyCurrents/Physics/circuitsphase.htm>.

Figurliste

Bemærk, at alle billeder, hvor der ikke er anført kilde er egne billeder.

Figur 1: Oversigt over bagtanke for Alarmduino-systemet.....	5
Figur 2: Foregående funktionsdiagram for sikring af dør og vinduer	7
Figur 3: Foregående blokdiagram for sikring af dør og vinduer.....	7
Figur 4: Foregående funktionsdiagram for ringklokke.....	8
Figur 5: Foregående blokdiagram for ringklokke	8
Figur 6: Foregående funktionsdiagram for bevægelsessensor	8
Figur 7: Foregående blokdiagram for bevægelsessensor	8
Figur 8: Oprindeligt Gantt-diagram	9
Figur 9: Blokdiagram for sikring af dør og vinduer	10
Figur 10: Blokdiagram for sensor.....	11
Figur 11: Blokdiagram for ringklokke	11
Figur 12: Det reviderede Gantt-diagram	12
Figur 13: Fysisk udformning af Alarmduino.....	13
Figur 14: Samlet el diagram for Alarmduino	14
Figur 15: Blokdiagram for sikring af døre og vinduer	16
Figur 16: El diagram til sikring af døre og vinduer.....	17
Figur 17: Fysisk udformning af sikring af døre og vinduer	18
Figur 18: Statediagram for dør, sensor og vinduer.....	19
Figur 19: Udformning af tastatur & LCD.....	20
Figur 20: Tilkobling af fladkabel til tastatur og LCD	20
Figur 21: El diagram for tastatur	21
Figur 22: Færdiglavet tastatur (27).....	21
Figur 23: Uoverskuelige ledning til tastatur	21
Figur 24: Flowchart for tastatur.....	22
Figur 25: Oversigt over kode for state 0, med switch statement.....	22
Figur 26: Oversigt over kode for state 0, med if funktion	23
Figur 27: Flowchart for LCD.....	23
Figur 28: Logik analyse af I2C	24
Figur 29: Sendelse til LCD	25
Figur 30: Billede af ethernet shield (26)	25
Figur 31: Eksempel på forespørgsel.....	26
Figur 32: Flowchart for PrepareAlarm funktionen.....	27
Figur 33: Flowchart for ReadEthernetSite funktionen.....	28
Figur 34: Flowchart for AlarmStatusCheck funktionen.....	28
Figur 35: Flowchart for UpdateAlarmStatus funktionen.....	29
Figur 36: Flowchart for UpdateLog funktionen.....	30
Figur 37: Flowchart for UpdateFilename funktionen.....	31
Figur 38: Kode til ændring af timeout indstillinger	32
Figur 39: Billede af GSM modul (25)	33
Figur 40: Flowchart for GSM shield	34
Figur 41: Eksempel på fejl ved anvendelse af dansk / norsk karaktertabel	35
Figur 42: Eksempel på SMS uden æ, ø og å.....	35
Figur 43: Eksempel på SMS på engelsk	35
Figur 44: Markering af JP på GSM shield	36
Figur 45: Blokdiagram for ringklokke	37
Figur 46: Stump koder der kører funktionen SendSMS.....	37
Figur 47: Linje kode der skriver kodeord	37

Figur 48: Stump kode, der logger ind på FTP serveren	38
Figur 49: El-diagram for konfiguration af bevægelsessensor (10).....	39
Figur 50: El diagram for sensor	40
Figur 51: Low-pass filter (udsnit)	40
Figur 52: Frekvenskarakteristik for lavpass filter	41
Figur 53: U og I vekselstrøm over kondensator (30).....	41
Figur 54: Reaktansen i det komplekse plan	42
Figur 55: Frekvenskarakteristika af low-pass filter	42
Figur 56: Første bånd-pass filter (udsnit)	43
Figur 57: Frekvenskarakteristika af første bånd-pass filter	44
Figur 58: anden bånd-pass filter (udsnit).....	45
Figur 59: Frekvenskarakteristika af anden bånd-pass filter	45
Figur 60: Udgangssignalers afhængighed (28).....	46
Figur 61: Dobbelt comparator med to referencespændinger (udsnit)	46
Figur 62: Graf for operationsforstærker.....	47
Figur 63: Komponentdiagram for operationsforstærkerne (29)	47
Figur 64: Print-diagram for bevægelsessensor	48
Figur 65: Billedet viser det færdige print af sensorkredsløbet	49
Figur 66: Grafer for interval og indgangsniveau.....	49
Figur 67: Spændingsdeler og comporator (anden bånd-pass filter imellem).....	50
Figur 68: Flowchart for sensoren.....	51
Figur 69: Udformning af connector shieldet.....	52
Figur 70: Ændringer på printet	52
Figur 71: El diagram for connector	53
Figur 72: Illustration af website-strukturen for kontrolpanelet.....	54
Figur 73: Illustration af tabellerne på databasen.	55
Figur 74: Tabel for status på alarmen.....	55
Figur 75: Viser log over brud på alarmen	56
Figur 76: Udsnit af index.php	56
Figur 77: Flowchart for loginsystemet.....	56
Figur 78: Udsnit af opret.php	57
Figur 79: Velkomstmeddelelse	57
Figur 80: Flowchart for oprettelse af bruger.....	58
Figur 81: Udsnit af koden; oprettelse af bruger	58
Figur 82: Udsnit af alarm slukket Figur 83: Udsnit af alarm tændt.....	59
Figur 84: Anvendelse af Sessions.....	59
Figur 85: Flowchart for login.php.....	60
Figur 86: Krypteringsfasen.....	61
Figur 87: Udsnit af tabellen members	61
Figur 88: Udsnit af rediger.php.....	62
Figur 89: Flowchart for rediger.php.....	63
Figur 90: Kode fra databaseconnect.php.....	64
Figur 91: Anvendelse af 'require' funktionen.....	64
Figur 92: Udsnit af kode der logger brugeren ud	64
Figur 93: Flowchart for Update_log.php.....	65
Figur 94: Udsnit af kode; Updatering af log	65
Figur 95: Flowchart for Read_alarm.php.....	66
Figur 96: Alarm_log.php; Status: tændt.....	67
Figur 97: Alarm_log.php; Status: brudt	67

Figur 98: Udsnit af kode; Lamper	67	
Figur 99: Flowchart for Alarm_log.php	Figur 100:Flowchart 2 for Alarm_log.php...	68
Figur 101: Udsnit af kode; Switchstatement.....	68	
Figur 102: Flowchart for Camera_log.php	69	
Figur 103: Udsnit af kode; while loop.....	69	

Tabelliste

Tabel 1: Liste over ind- og udgange på Arduino	15
Tabel 2: Liste over filer og hvilket modul de tilhører	16
Tabel 3: Tabel over tests til dør, sensor og vinduer	19
Tabel 4: Tabel over funktioner til ethernet shieldet	26

Bilag:

Bilag 1: kode Controlpanel.ino

```
1  /*Inkluder disse biblioteker:*/
2  #include <Wire.h>
3  #include <LiquidCrystal_I2C.h>
4  #include <Keypad.h>
5  #include <SD.h>
6  #include <SPI.h>
7  #include <Ethernet.h>
8  #include <SoftwareSerial.h>
9
10  SoftwareSerial GPRSSerial(7, 8);
11  LiquidCrystal_I2C lcd(0x27, 16, 2); //Størrelsen på LCD displayet
12
13  /*Liste over variabler der skal tjekkes globalt.*/
14  int state = 0;
15  int StateAlarm = 0;
16  int PendingChangeAlarm = 0;
17  int PendingChangeLog = 0;
18  int PendingChangefileName = 0;
19  int BOOT = 0;
20  int BOOT_SD = 0;
21  int Doorbell_pressed = 0;
22  int SensorReg = 0;
23  unsigned long PendingTimerAlarm = 0;
24  unsigned long PendingTimerLog = 0;
25  unsigned long PendingTimer = 0;
26  unsigned long PendingTimerfileName = 0;
27
28  //Variabler til password
29  String password;
30  String MasterPassword = "584739G";
31  int WhichContact = 0;
32
33  //Liste over filer med kode:
34  #include "GSM.h"
35  #include "EthernetSHIELD.h"
36  #include "Door_and_window.h"
37  #include "Matrix.h"
38  #include "Display.h"
39  #include "Doorbell.h"
40  #include "Sensor.h"
41
42  void setup() {
43    Serial.begin(9600);
44    Door_and_windows_setup();
45    Ethernet_setup();
46    GSM_setup();
47    LCDSetup();
48    SD_setup();
49    Sensor_setup();
50  }
51
52  void loop() {
53    PendingTimer = millis();
54    LCDDraw();
55    Door_and_windows();
56    Sensor_reg();
57    Matrix();
58    AlarmStatusCheck();
59    ButtonState_DoorbellContact = digitalRead(DoorbellContact);
60    /* Det følgende sørger for, at der ikke bliver opdateret status på alarmen,
61     * mens der er andre vigtigere funktioner der kører, fx ændring af navn på
62     * billedet der uploades.
63     */
64    if (ButtonState_DoorbellContact == HIGH) {
65      Serial.print(ButtonState_DoorbellContact);
66      doFTP();
```

```
67     } else if (PendingChangeAlarm == 1) {  
68         if (PendingTimer > PendingTimerAlarm + 5000) {  
69             UpdateAlarmStatus();  
70         }  
71     } else if (PendingChangeLog == 1) {  
72         if (PendingTimer > PendingTimerLog + 5000) {  
73             UpdateLog();  
74         }  
75     } else if (PendingChangefileName == 1) {  
76         if (PendingTimer > PendingTimerfileName + 5000) {  
77             UpdateFilename();  
78         }  
79     }  
80     else {  
81         ReadEthernetSite();  
82     }  
83 }
```

Bilag 2: kode Display.h

```
1  #include "Arduino.h"
2
3  unsigned long PendingTimerLCD_stop = 500;
4
5  /*Setup LCD*/
6  void LCDSetup() {
7      lcd.init();
8      lcd.backlight();
9      lcd.begin(16, 2);
10     lcd.setCursor(0, 0);
11     lcd.print("Alarm booting");
12 }
13
14 /*Genopfrisk LCD*/
15 void LCDDraw() {
16     if (BOOT == 1 && BOOT_SD == 1) { //Er systemet bootet?
17 //Er det mere end 500ms siden sidste genopfriskning?
18         if (PendingTimer >= PendingTimerLCD_stop) {
19             //Clear LCD og sæt markør på position (0,0).
20             lcd.clear();
21             lcd.setCursor(0, 0);
22             switch (state) { //Switch funktion.
23                 case 0: //Tilfælde 0: Alarmen er slukket.
24                     lcd.print("Alarm OFF");
25                     break;
26                 case 1: //Tilfælde 1: Alarmen er blevet tændt, og der ventes 20 sekunder.
27                     lcd.print("Alarm ON Waiting");
28                     break;
29                 case 2: //Tilfælde 2: Alarmen er tændt og holder øje.
30                     lcd.print("Alarm ON");
31                     if (password.length() == 0) {
32                         lcd.setCursor(0, 1);
33                         lcd.print("Alarmen er aktiv");
34                     }
35                     break;
36                 case 3: //Tilfælde 3: Alarmen er blevet brudt.
37                     lcd.print("Alarm ON");
38                     if (password.length() == 0) {
39                         lcd.setCursor(0, 1);
40                         lcd.print("Alarmen er brudt!");
41                     }
42                     break;
43                 case 4: //Tilfælde 4: Døren er blevet åbnet, og der ventes 20 sekunder.
44                     lcd.print("Alarm ON Waiting");
45                     break;
46             }
47             //Er der skrevet noget på tastaturet, som skal vises på LCD?
48             if (password.length() >= 1) {
49                 lcd.setCursor(0, 1);
50                 lcd.print(password);
51             }
52             PendingTimerLCD_stop = PendingTimer + 500;
53         }
54     }
55 }
```


Bilag 3: kode Door_and_window.h

```
1  #include "Arduino.h"
2
3  //Definere indgange:
4  const int TriggerDoor = 2;
5  const int TriggerWindow1 = 3;
6  const int TriggerWindow2 = 5;
7
8  //Definere variabler:
9  int ButtonStateTriggerDoor = 1;
10 int ButtonStateTriggerWindow1 = 1;
11 int ButtonStateTriggerWindow2 = 1;
12 unsigned long counter_start = 0;
13 unsigned long counter_stop = 0;
14
15 /*Setup Døre og vinduer*/
16 void Door_and_windows_setup() {
17     pinMode(TriggerDoor, INPUT);
18     pinMode(TriggerWindow1, INPUT);
19     pinMode(TriggerWindow2, INPUT);
20 }
21
22 /*Døre og vinduer*/
23 void Door_and_windows() {
24     //Læser kontakternes tilstand (1 = aktiveret eller 0 = deaktiveret)
25     ButtonStateTriggerDoor = digitalRead(TriggerDoor);
26     ButtonStateTriggerWindow1 = digitalRead(TriggerWindow1);
27     ButtonStateTriggerWindow2 = digitalRead(TriggerWindow2);
28
29     /*Statemaskine for sikring af døre og vinduer*/
30     if (state == 0) { //Tilstand 0, hvor alarmen ikke er tændt
31         if (StateAlarm == 1) { //Er alarmen blevet tændt?
32             state = 1; //Skift til tilstand 1
33             //Sæt tæller til 20 sek.
34             counter_start = millis();
35             counter_stop = counter_start + 20000;
36         }
37         else {
38             counter_start = 0;
39             return;
40         }
41     }
42
43     if (state == 1) { //Tilstand 1, hvor der ventes
44         counter_start = millis(); //Opdater den aktuelle tid
45         if (StateAlarm == 0) { //Er alarmen blevet slukket?
46             state = 0; //Skift til tilstand 0
47         }
48         //Når tiden tælleren, altså de 20 sek.
49         else if (counter_start >= counter_stop) {
50             state = 2; //Skift da til tilstand 2.
51         }
52         else {
53             return;
54         }
55     }
56
57     if (state == 2) { //Tilstand 2, hvor alarmen er tændt og holder øje.
58         if (StateAlarm == 0) { //Er alarmen blevet slukket?
59             state = 0; //Skift til tilstand 0
60         } else if (ButtonStateTriggerDoor == 0) { //Tjek dør-kontakten
61             state = 4; //Skift til tilstand 4.
62             //Sæt tæller til 20 sek.
63             counter_start = millis();
64             counter_stop = counter_start + 20000;
65         } else if (ButtonStateTriggerWindow1 == 0) { //Tjek vindue1-kontakten
66             state = 3; //Skift til tilstand 3
```

```
67     WhichContact = 2; //Husk, at det er vindue1 kontakten der er brudt.
68 } else if (ButtonStateTriggerWindow2 == 0) { //Tjek vindue2-kontakten
69     state = 3; //Skift til tilstand 3
70     WhichContact = 3; //Husk, at det er vindue2 kontakten der er brudt.
71 }
72 else if (SensorReg == 1) { //Tjek sensor
73     state = 3; //Skift til tilstand 3.
74     WhichContact = 4; //Husk, at det er sensoren der er brudt.
75 }
76 else {
77     return;
78 }
79 }
80
81 if (state == 3) { //Tilstand 3, hvor en af alarm-kontakterne er brudt
82     if (StateAlarm == 0) { //Er alarmeren blevet slukket?
83         state = 0; //Skift til tilstand 0
84     }
85     if (WhichContact > 0) { //Hvis der er brudt en kontakt eller sensor:
86         SendSMS(); //Send SMS
87         UpdateLog(); //Opdater log
88         WhichContact = 0; //Glem at der er brudt en kontakt
89     }
90 }
91
92 if (state == 4) { //Tilstand 4, hvor døren er åben.
93     if (StateAlarm == 0) { //Hvis man slukker alarmeren. Skift da til tilstand 0;
94         state = 0; //Skift til tilstand 0
95     }
96     counter_start = millis(); //Tæl op
97     if (counter_start >= counter_stop) { //Når tiden tælleren, altså de 20 sek.
98         state = 3; //Skift til tilstand 3.
99         WhichContact = 1; //Husk at det er døren, der er brudt.
100     }
101 }
102 }
```

Bilag 4: kode Doorbell.h

```
1  /*
2   Denne kode er i store dele kopieret fra http://playground.arduino.cc/Code/FTP,
3   og da det fungerer har man valgt ikke at sætte sig ind i koden, til mindst
4   detalje.
5  */
6  #include "Arduino.h"
7  //Sikrer at ethernet shieldet er inkluderet
8  #include "EthernetSHIELD.h"
9
10 //Kontakt pin til dørklokke:
11 const int DoorbellContact = 6;
12 int ButtonState_DoorbellContact = 0;
13
14 char outBuf[128];
15 char outCount;
16
17 //Filnavn til billede der uploades: (8.3 format)
18 char fileName[13] = "Ring1.png";
19
20 /*Opsætning af SD-kort*/
21 void SD_setup() {
22     pinMode(DoorbellContact, INPUT);
23     digitalWrite(10, HIGH);
24     //Mens SD kort ikke kan indlæses kan systemet ikke boote.
25     while(SD.begin(4) == 0) {
26         BOOT_SD = 0;
27     }
28     BOOT_SD = 1;
29     delay(2000);
30 }
31
32 File fh; //
33
34 void efail() {
35     byte thisByte = 0;
36     client.println(F("QUIT"));
37     while (!client.available()) delay (1);
38     while (client.available()) {
39         thisByte = client.read();
40         Serial.write(thisByte);
41     }
42     client.stop();
43     Serial.println(F("Command disconnected"));
44     fh.close();
45     Serial.println(F("SD closed"));
46 }
47
48 byte eRcv() {
49     byte respCode;
50     byte thisByte;
51     while(!client.available()) delay(1);
52     respCode = client.peek();
53     outCount = 0;
54     while (client.available()) {
55         thisByte = client.read();
56         Serial.write(thisByte);
57         if (outCount < 127) {
58             outBuf[outCount] = thisByte;
59             outCount++;
60             outBuf[outCount] = 0;
61         }
62     }
63     if (respCode >= '4') {
64         efail();
65         return 0;
66     }
67 }
```

```
67     return 1;
68 }
69
70 /*Upload billede*/
71 byte doFTP() {
72     Doorbell_pressed = 1; //Husk at dørklokken er trykket
73     if (state > 0) { //Hvis alarmen er tændt
74         SendSMS(); //Send SMS
75     }
76     Doorbell_pressed = 0; //Glem at dørklokken er trykket
77     client.stop();
78     int SDfail = 0;
79     //Så længe den ikke kan få fat i filen, dog maks fem gange.
80     while(!fh && SDfail < 5) {
81         fh = SD.open(fileName, FILE_READ);
82         SDfail++;
83     }
84     Serial.println(F("SD opened"));
85     if (client.connect(server, 21)) {
86         Serial.println(F("command connected"));
87     }
88     else {
89         fh.close();
90         Serial.println(F("Command connection failed"));
91         return 0;
92     }
93
94     if (!Rcv()) return 0;
95     client.println(F("USER glen0930"));
96     if (!Rcv()) return 0;
97     //Personligt password til FTP server - ikke inkluderet i rapporten
98     //(dog som elektronisk bilag)
99     #include "PasswordFile.h"
100    if (!Rcv()) return 0;
101    client.println(F("SYST"));
102    if (!Rcv()) return 0;
103    client.println(F("Type I")); //Send binært
104    if (!Rcv()) return 0;
105    client.println(F("PASV")); //Gå i passiv mode
106    if (!Rcv()) return 0;
107
108    char *tStr = strtok(outBuf, "(");
109    int array_pasv[6];
110    for (int i = 0; i < 6; i++) {
111        tStr = strtok(NULL, "(");
112        array_pasv[i] = atoi(tStr);
113        if (tStr == NULL) {
114            Serial.println(F("Bad PASV Answer"));
115        }
116    }
117
118    //Tildeling af port til upload på FTP
119    unsigned int hiPort, loPort;
120    hiPort = array_pasv[4] << 8;
121    loPort = array_pasv[5] & 255;
122    Serial.print(F("Data port: "));
123    hiPort = hiPort | loPort;
124    Serial.println(hiPort);
125
126    int dclient_fail = 0;
127    while(!dclient.connect(server, hiPort) && dclient_fail < 5) {
128        dclient_fail++;
129        dclient.connect(server, hiPort);
130        if (dclient.connect(server, hiPort)) {
131            Serial.println(F("Data connected"));
132            dclient_fail = 0;
133        }
134    }
135    if (dclient_fail >= 5) {
136        Serial.println(F("Data connection failed"));
```

```
137     client.stop();
138     fh.close();
139     return 0;
140 }
141 client.print(F("STOR ")); //FTP kommando for gem.
142 //Adressen hvor der skal gemmes
143 client.print("/HX-15-c-el/glen0930/Public/Doorbell_IMG/");
144 //filnavnet der skal gemmes (opretter filen på FTP serveren)
145 client.println(fileName);
146
147 if (!Rcv()) {
148     dclient.stop();
149     return 0;
150 }
151
152 /*Upload delen:*/
153 Serial.println(F("Writing"));
154 byte clientBuf[64]; //Definer, at der uploades i pakker med 64 byte i hver.
155 int clientCount = 0;
156 /*Skriv til LCD at der uploades.*/
157 lcd.clear();
158 if (state >= 1) { //Hvis alarmeren er tændt
159     lcd.print("Alarm ON"); //Skriv <--
160     lcd.setCursor(0, 1); //Flyt markør
161     lcd.print("Uploader til FTP"); //Skriv <--
162 }
163 if (state == 0) { //Hvis alarmeren er slukket
164     lcd.print("Alarm OFF"); //Skriv <--
165     lcd.setCursor(0, 1); //Flyt markør
166     lcd.print("Uploader til FTP"); //Skriv <--
167 }
168 while (fh.available()) { //Så længe der er adgang til filen på SD-kort
169     clientBuf[clientCount] = fh.read(); //Læs filen (en byte ad gangen)
170     clientCount++; //Tæl op på antallet af bytes
171     if (clientCount > 63) { //Hvis der er læst mere end 63 bytes
172         dclient.write(clientBuf, 64); //Skriv disse 64 bytes til FTP
173         clientCount = 0; //Nulstil antal
174     }
175 }
176 if (clientCount > 0) dclient.write(clientBuf, clientCount);
177 /*Afbryd forbindelse til FTP og SD*/
178 dclient.stop();
179 Serial.println(F("Data disconnected"));
180 if (!Rcv()) return 0;
181 client.println(F("QUIT"));
182 if (!Rcv()) return 0;
183 client.stop();
184 Serial.println(F("Command disconnected"));
185 fh.close();
186 Serial.println(F("SD closed"));
187 UpdateFilename(); //Opdater database + filnavn
188 //Reset ref. tid for opdatering af status.
189 lastReadStateCounter = millis() + 2000;
190 return 1;
191 }
```

Bilag 5: kode EthernetSHIELD.h

```
1  #include "Arduino.h"
2  #ifndef EthernetSHIELD
3  #define EthernetSHIELD
4
5  byte mac[] = { 0x00, 0xAA, 0xCB, 0xEC, 0xAF, 0x03 }; //MAC adresse for ethernetet
6
7  //Skolens setup, IP, DNS, Gateway og link
8  IPAddress ip(10, 85, 5, 10);
9  IPAddress Dns(10, 26, 0, 11);
10 IPAddress GW(10, 85, 0, 1);
11 char server[] = "htx-dev.ucholstebro.dk";
12
13 //Klienter til serveren:
14 EthernetClient client;
15 EthernetClient dclient;
16
17 //Til læsning af side:
18 char c;
19 String c_c;
20 String c_read = "";
21
22 //Tidstagnning
23 unsigned long lastReadStateCounter = 2000;
24
25 //Variabel til, at huske at der har været en ændring
26 int remember = 0;
27
28 /*Setup Ethernet*/
29 void Ethernet_setup() {
30     //Start ethernet og setup serveren.
31     delay(1000);
32     Ethernet.begin(mac, ip, Dns, GW);
33 }
34
35 /*Står der noget på Read_Alarm.php?*/
36 void ReadEthernetSite() {
37     while (client.available()) { //Mens det er muligt at få forbindelse til klienten:
38         c = client.read(); //Læs en karakter fra klienten.
39         c_c.concat(c); //Tilføj karakteren til en streng
40     }
41 }
42
43 /*Kør PrepareAlarm funktionen*/
44 void PrepareAlarm() {
45     if (PendingChangeAlarm == 0) { //Har der været en mislykket forbindelse?
46         if (!client.available()) {
47             client.stop(); //Stop forbindelse
48             //Kan der oprettes forbindelse til serveren?
49             if (client.connect(server, 80)) {
50                 client.print("GET /HX-15-c-el/glen0930/Read_Alarm.php"); //Indlæs siden <--
51                 client.println(" HTTP/1.1");
52                 client.print("HOST: ");
53                 client.println(server);
54                 client.println("User-Agent: arduino-ethernet");
55                 client.println("Connection: close");
56                 client.println();
57             }
58             else { //Ellers, skriv at forbindelsen fejlede.
59                 Serial.println("connection failed");
60             }
61         }
62     }
63 }
64
65 /*Hold øje med status for alarm på website*/
66 void AlarmStatusCheck() {
```



```
67 //Er det mere end 2 sek siden sidste læsning?
68 if (PendingTimer >= lastReadStateCounter) {
69     PrepareAlarm(); //Kør funktionen PrepareAlarm
70     lastReadStateCounter += 2000; //Læg 2 sek. til referencetiden
71 }
72 while (c_c.length() >= 1) { //Er der skrevet noget til strengen for hjemmesiden?
73     c_read.concat(c_c); //Tilføj strengen til strengen der analyseres
74     //Skriv strengen der analyseres i serial monitor (bruges til debugging)
75     Serial.println(c_read);
76     c_c = ""; //Tøm det læste fra hjemmesiden
77     if (c_read.endsWith("$")) { //Ender det analyseret med $?
78         StateAlarm = 1; //Tænd alarmen
79         remember = 1; //Husk at der er sket en ændring
80     }
81     if (c_read.endsWith("€")) { //Ender det analyseret med €?
82         StateAlarm = 0; //Sluk alarmen
83         remember = 1; //Husk at der er sket en ændring
84     }
85 }
86 if (remember == 1) { //Er der noget, der er husket?
87     remember = 0; //Glem at der husket noget
88     c_read = ""; //Tøm analysestrengen
89     BOOT = 1; //Husk at systemet er bootet
90 }
91 }
92
93 /*Opdater alarm status*/
94 void UpdateAlarmStatus() {
95     client.stop(); //Afbryd forbindelse til klienten
96     if (client.connect(server, 80)) { //Kan der oprettes forbindelse til serveren?
97         //Endre værdien for variabelen på php-siden, tjek, til at stemme overens med
98         //StateAlarm (Alarm ON/OFF)
99         client.print("GET /HX-15-c-el/glen0930/Read_Alarm.php?tjek=");
100         client.print(StateAlarm);
101         client.println(" HTTP/1.1");
102         client.print("HOST: ");
103         client.println(server);
104         client.println("User-Agent: arduino-ethernet");
105         client.println("Connection: close");
106         client.println();
107         Serial.println("Connnection success");
108         PendingChangeAlarm = 0; //Ikke en mislykket forbindelse
109     }
110     else { //Mislykket forbindelse:
111         Serial.println("connection failed trying again");
112         PendingChangeAlarm = 1; //Husk, at det er en mislykket forbindelse
113         PendingTimerAlarm = millis(); //Noter tiden
114     }
115 }
116
117 /*Opdater log over brudte alarmer*/
118 void UpdateLog() {
119     client.stop(); //Afbryd forbindelsen til serveren
120     if (client.connect(server, 80)) { //Kan der forbindes til serveren?
121         /* Ændre værdien for variabelen på php-siden, State, til at stemme overens med
122         * WhichContact der holder styr på hvilken kontakt / sensor der er brudt
123         */
124         client.print("GET /HX-15-c-el/glen0930/Update_log.php?State=");
125         client.print(WhichContact);
126         client.println(" HTTP/1.1");
127         client.print("HOST: ");
128         client.println(server);
129         client.println("User-Agent: arduino-ethernet");
130         client.println("Connection: close");
131         client.println();
132         PendingChangeLog = 0; //Ikke en mislykket forbindelse
133     }
134     else { //Mislykket forbindelse
135         Serial.println("connection failed trying again");
136         PendingChangeLog = 1; //Husk, at det er en mislykket forbindelse
```

```
137     PendingTimerLog = millis(); //Noter tiden
138 }
139 }
140
141 /*Opdater navn på FTP serveren for det uploadet billede*/
142 void UpdateFilename() {
143     client.stop(); //Afbryd forbindelsen til serveren
144     if (client.connect(server, 80)) { //Kan der forbindes til serveren?
145         //Indlæs siden ChangeNameFTP, der kører en funktion,
146         //som omdøber navnet på det uploadet billede
147         client.print("GET /HX-15-c-el/glen0930/ChangeNameFTP.php");
148         client.println(" HTTP/1.1");
149         client.print("HOST: ");
150         client.println(server);
151         client.println("User-Agent: arduino-ethernet");
152         client.println("Connection: close");
153         client.println();
154         PendingChangefileName = 0; //Ikke en mislykket forbindelse
155         Serial.println("Change file name success");
156     }
157     else { //Mislykket forbindelse
158         Serial.println("connection failed trying again");
159         PendingChangefileName = 1; //Husk, at det er en mislykket forbindelse
160         PendingTimerfileName = millis(); //Noter tiden
161     }
162 }
163 #endif
```

Bilag 6: kode GSM.h

```
1  #include "Arduino.h"
2  #include <String.h>
3
4  /*Setup GSM*/
5  void GSM_setup() {
6      GPRSSerial.begin(19200);
7      delay(500);
8      pinMode(9, OUTPUT);
9      digitalWrite(9, LOW);
10     delay(300);
11     digitalWrite(9, HIGH);
12 }
13
14 /*Send SMS*/
15 void SendSMS() {
16     /*Gør klar til at sende SMS*/
17     GPRSSerial.print("AT+CMGF=1\r");
18     delay(100);
19     GPRSSerial.println("AT+CMGS = \""+4521521508"\");
20     delay(100);
21     /* I det følgende finder programmet hvilken kontakt / sensor der er brudt,
22      * eller om der er ringet på døren. Ud fra det sender den en besked med
23      * den rigtige tekst.
24      */
25     if (WhichContact == 1) {
26         GPRSSerial.println("Hello. There has been an incident at your front door. Se more
27         details at htx-elev.ucholstebro.dk/HX-15-c-el/glen0930/index.php");
28     }
29     if (WhichContact == 2) {
30         GPRSSerial.println("Hello. There has been an incident at your window 1. Se more
31         details at htx-elev.ucholstebro.dk/HX-15-c-el/glen0930/index.php");
32     }
33     if (WhichContact == 3) {
34         GPRSSerial.println("Hello. There has been an incident at your window 2. Se more
35         details at htx-elev.ucholstebro.dk/HX-15-c-el/glen0930/index.php");
36     }
37     if (WhichContact == 4) {
38         GPRSSerial.println("Hello. There has been an incident at the sensor in your
39         house. Se more details at htx-elev.ucholstebro.dk/HX-15-c-el/glen0930/index.php");
40     }
41     if (Doorbell_pressed == 1) {
42         GPRSSerial.println("Hello. Someone has pressed your doorbell. Se more details at
43         htx-elev.ucholstebro.dk/HX-15-c-el/glen0930/index.php");
44     }
45     delay(1000);
46     GPRSSerial.print((char)26); //AT kommando for send SMS
47     delay(100);
48     GPRSSerial.println();
49 }
```

Bilag 7: kode Matrix.h

```
1  #include "Arduino.h"
2
3  char key = 0;
4  const byte ROWS = 4; // 4 rækker på matrix
5  const byte COLS = 4; // 4 kolonner på matrix
6
7  // Definerer keymap, hexadecimal
8  char keys[ROWS][COLS] = {
9      {'1', '2', '3', 'C'},
10     {'4', '5', '6', '-'},
11     {'7', '8', '9', '.'},
12     {'*', '0', '#', 'G'}
13 };
14
15 //forbinder rækker og kolonner til pinouts på tastatur
16 byte rowPins[ROWS] = { 29, 27, 25, 23};
17 byte colPins[COLS] = { 31, 33, 35, 37};
18 Keypad customkeypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
19
20 /*Hold øje med tastatur*/
21 void Matrix() {
22     key = customkeypad.getKey(); // Laver en ny character der gemmer værdien fra
23     matrixtastaturet
24     if (password.length() < 17) { //Er længden på password mindre end 17 karaktere?
25         if (key) { //Tilføj den trykket knaps karakter til password
26             password += key;
27         }
28         else {
29             return;
30         }
31     }
32     if (password == MasterPassword){ //Er password lig MasterPassword?
33         password = ""; ///Tøm password strengen for karaktere.
34         StateAlarm = !StateAlarm; //Ændre status på alarm til modsat før.
35         UpdateAlarmStatus(); //Opdater alarm status på server.
36     }
37     if (key == 'C'){ //Er der trykket på knappen C?
38         password=""; //Tøm password strengen.
39     }
40 }
```

Bilag 8: kode Sensor.h

```
1  #include "Arduino.h"
2
3  //Indlæs variabler
4  int SensorPin = A15;
5  int SensorValue = 0;
6  int SensorCount = 0;
7  int SensorFirstIncident = 0;
8  unsigned long lastSensorTimer = 200;
9  unsigned long SensorTimerActive = 0;
10
11 /*Setup sensor*/
12 void Sensor_setup(){
13     pinMode(SensorPin, INPUT);
14 }
15
16 /*Registrering af bevægelser*/
17 void Sensor_reg() {
18     //Er der gået mere end 200ms siden sidste læsning?
19     if (PendingTimer >= lastSensorTimer) {
20         SensorValue = analogRead(SensorPin); //Læs sensorens analoge værdi
21         Serial.println(SensorValue);
22         //Gør klar til, at der skal gå 200ms før næste læsning.
23         lastSensorTimer = PendingTimer + 200;
24         if (SensorValue == 0) { //Hvis sensoren er afbrudt
25             return; //Afslut
26         }
27         //Er der registreret en bevægelse, og er det første gang?
28         if (SensorValue <= 220 && SensorFirstIncident == 0) {
29             SensorFirstIncident = 1; //Husk at der har været registrering før
30             //Husk tiden hvortil det sker + 3000ms, som reference.
31             SensorTimerActive = millis() + 3000;
32         }
33     }
34     //Så længe tiden er mindre end ref. tiden
35     while (PendingTimer <= SensorTimerActive) {
36         if (SensorValue <= 220) { //Er der registreret en bevægelse?
37             SensorCount++; //Tæl en op, for en registreret bevægelse
38         }
39         if (SensorCount >= 10) { //Er der lavet 10 eller flere registreringer?
40             SensorReg = 1; //Tolk det som en registrering.
41             SensorTimerActive = 0; //Nulstil ref. tiden.
42             SensorCount = 0; //Nulstil antal registreringer.
43             return;
44         }
45     }
46     //Er der gået længere tid end ref. tiden?
47     if (PendingTimer >= SensorTimerActive) {
48         //Glem registreringer:
49         SensorFirstIncident = 0;
50         SensorReg = 0;
51         SensorCount = 0;
52     }
53 }
```

Bilag 9: index.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link href="Style.css" rel="stylesheet">
    <title>Kontrolcenter</title>
  </head>
  <body id="Background">
    <?php
      require 'databaseconnect.php'; //Denne fil skal være inkluderet før at resten af php
      kan kører.
      session_start();
      if (isset($_POST['user']) && isset($_POST['pass'])) { //Hvis der indtastet noget i
      user og password feltet:
        $username = $_POST['user']; //Sæt det indtastede i userfeltet som variable
        $get_hash = mysqli_query($mysqlConnection, "SELECT salt FROM members WHERE user-
        name='$username'"); //Lav en forespørgsel på databasen, hvor den leder efter den salt
        hvor usernamet er ligmed det indtastede username.
        if ($row = $get_hash->fetch_assoc()) { //Hvis salt så findes:
          $pw = $_POST['pass']; //Definer variable, som indeholder det indtastede i pass-
          wordfeltet
          $hash_pw = crypt($pw, $row['salt']); //Hash password (BLOWFISH), m. den salt
          der findes for den unikke bruger fra databasen.
        }
        $query = "SELECT * from members WHERE username = '{$username}' AND password =
        '{$hash_pw}' LIMIT 1"; //Lav en forespørgsel på databasen, hvor den leder efter
        alt fra tabellen members hvor brugernavnet og passwordet er ens med det indta-
        stede.
        $result = mysqli_query($mysqlConnection, $query); //Lav forespørgslen.
        $queryname = "SELECT fornavn from members where username = '{$username}' LIMIT
        1";
        $queryresult = mysqli_query($mysqlConnection,$queryname);
        $result_first= mysqli_fetch_assoc($queryresult);
        if (!$result->num_rows == 1) { //Hvis brugernavn og password ikke er ens:
          echo "<script type='text/javascript'>alert('Forkert brugernavn eller adgangs-
          kode!');</script>";
        } else { //Hvis det er ens:
          $_SESSION["Login"] = "YES";
          $_SESSION['fornavn'] = $result_first['fornavn'];
          header('Location:login.php');
        }
      }
      mysqli_close($mysqlConnection); //Luk mysqli forbindelsen
    ?>
    <div class="Top">Kontrolcenter</div>
    <div class="Body">
      <center>
        <div class="TextOnLogin">
          Velkommen til Alarmduinos adgangsportal til dit alarmsystem. For at styre dit
          alarmsystem bedes du logge ind.
        </div>
        <form class="Login" name="form1" method="POST" action="index.php">
          Brugernavn:<br>
          <input type="text" id="Username" class="INPUTBOX" name="user" ><br>
          Adgangskode:<br>
          <input type="password" id="password" class="INPUTBOX" name="pass"><br><br>
          <input class="loginknap" type="submit" name="send" value="Login">
        </form>
      </center>
    </div>
  </body>
</html>
```


Bilag 10: databaseconnect.php

```
<?php
    $host = "localhost"; // Host navn
    $username = "15_c_el_glen0930"; // Mysql brugernavn
    $password = "ub4h3"; // Mysql password
    $db_name = "15_c_el_glen0930"; // Database navn
    $tbl_name = "members"; // Tabel navn
    $mysqlConnection = mysqli_connect($host, $username, $password); //Tilslut til databasen
    if (!$mysqlConnection){ //Hvis den ikke kan etablerer en forbindelse til databasen, så
        skal den printe:
        echo "<script type='text/javascript'>alert('Kan ikke skabe en sikker forbindelse til da-
        tabasen');</script>";
    } else { //Ellers, hvis den har skabt forbindelse, så skal den printe:
        mysqli_select_db($mysqlConnection,$db_name);
    }
    mysqli_report(MYSQLI_REPORT_ERROR);
?>
```

Bilag 11: Style.css

```
#Background{
    background-image: url("paper.gif");
}
body{
    background-image: url("paper.gif");
    z-index:-1;
}
.Top{
    position: relative;
    width: auto;
    height: 100px;
    text-align: center;
    text-height: font-size;
    font-size: 72pt;
    z-index: 1;
    margin-left: auto;
    margin-right: auto;
}
.Body{
    position:relative;
    width: auto;
    height: auto;
    z-index: 1;
}
.Login{
    position: relative;
    top: 100px;
    width: 200px;
    height: 150px;
    font-family: Cambria;
    font-size: 20pt;
    z-index: 2;
}
.TextOnLogin{
    position:relative;
    width: auto;
    z-index: 1;
}
.INPUTBOX{
    width: 200px;
    height: 14pt;
    font-size: 14pt;
}
.loginknap{
    position: relative;
    top:-35px;
}

.kontrolboks{
    height: auto;
    width:auto;
}
.fjernbruger{
    width:25px;
    margin: 10px;
}
.info{
    position: relative;
    text-align: left;
    z-index: 6;
}
.log,.log1,.log2,.log3,.log4{
    border: 1px solid black;
    border-collapse: collapse;
}
.log_time{
    border:1px solid black;
    border-collapse: collapse;
    position: fixed;
    left:1050px;
    top:180px;
    z-index: -1;
}
.log_time_window{
    border:1px solid black;
    border-collapse: collapse;
    position: fixed;
    left:1050px;
    top:180px;
    z-index: -1;
}
.log_time_sensor{
    border:1px solid black;
    border-collapse: collapse;
    position: fixed;
    left:1050px;
    top:180px;
    z-index: 1;
}
.plantegning{
    z-index:0;
    position: fixed;
    left:290px;
    top:185px;
}
```

```
.lightbulbdoor{
  z-index:1;
  position: fixed;
  left:465px;
  top:255px;
}
.lightbulbwindowone{
  z-index:1;
  position: fixed;
  left:367px;
  top:520px;
}
.lightbulbwindowtwo{
  z-index: 1;
  position: fixed;
  right:550px;
  top:250px;
}
.lightbulbsensor{
  z-index: 1;
  position: fixed;
  right:480px;
  top:490px;
}
.camera_log{
  height:100px;
  width:100px;
  opacity: 0.6;
  z-index:0;
}
.camera_log:hover{
  opacity: 1.0;
  cursor:pointer;
}
.overlay {
  position: fixed;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  background: rgba(0, 0, 0, 0.7);
  transition: opacity 500ms;
  visibility: hidden;
  opacity: 0;
  z-index:1;
}
.overlay:target {
  visibility: visible;
  opacity: 1;
}
```

```
}
.popup {
  margin: 70px auto;
  padding: 20px;
  background: #fff;
  border-radius: 5px;
  width: 60%;
  position: relative;
}
.popup .close {
  position: absolute;
  top: 20px;
  right: 30px;
  transition: all 200ms;
  font-size: 30px;
  font-weight: bold;
  text-decoration: none;
  color: #333;
}
.popup .close:hover {
  color: #06D85F;
}
.popup .content {
  max-height: 30%;
  overflow: auto;
}
#right{
  text-align: right;
  position: fixed;
}
.change_user{
  position: fixed;
  right:0px;
  z-index:5;
}
.back{
  z-index:5;
}
.logud {
  position: relative;
  right:0px;
  top:250px;
  z-index:5;
}
.change_status {
  position: relative;
  top:-100px;
}
a {
  z-index:5;
}
```

Bilag 12: login.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link href="Style.css" rel="stylesheet">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-
    awesome.min.css">
    <title>Oversigt over alarm</title>
  </head>
  <body id="Background">
    <?php
      require 'databaseconnect.php';
      session_start();
      if($_SESSION["Login"] != "YES"){
        header("Location: index.php");
      }
      $print = "";
      $query = "SELECT status FROM alarm_status";
      $result = mysqli_query($mysqlConnection, $query); //Lav forespørgslen.
      $row = mysqli_fetch_assoc($result);
      if ($row['status'] == 1){ //Hvis alarm er tændt:
        $status = '<span style="color:#0fe01a; font-weight:bold;text-align:center;">Alarm
        tændt</span>';
      } else if ($row['status'] == 0){ //Hvis alarm er slukket:
        $status = '<span style="color:#bb0505;font-weight:bold;text-align:center;">Alarm
        slukket</span>';
      }

    ?>
    <div class="Top">Kontrolcenter</div>
    <?php
      if(isset($_POST['tryk'])) { //Hold øje med ON/OFF knap
        $tjek = "";
        if($_POST['submit']=='ON' && $row['status']==0){ //Hvis knap er ON og tilstanden
        på alarm er slukket
          $tjek = 1;
          $print = '<span style="color:#0fe01a; font-weight:bold;text-align:cen-
          ter;">Tænder alarm...</span>';

          } else if($_POST['submit']=='OFF' && $row['status']==1) { //Hvis knap er OFF
          og tilstand på alarm er tændt
            $tjek = 0;
            $print = '<span style="color:#bb0505;font-weight:bold;text-align:cen-
            ter;">Slukker alarm...</span>';
          }
        /*Indsæt tid og opdateret status i databasen*/
        $mysqli = new mysqli("localhost", "15_c_el_glen0930", "ub4h3",
        "15_c_el_glen0930");
        date_default_timezone_set("Europe/Copenhagen");
        $tid = time();
        $stmt = $mysqli->prepare("UPDATE alarm_status SET status = ?");
        $stmt->bind_param("s", $tjek);
        $stmt->execute();
        $stmt = $mysqli->prepare("UPDATE alarm_status SET tidspunkt = ?");
        $stmt->bind_param("i", $tid);
        $stmt->execute();
      }
      header( "refresh:5;url=login.php" );
    ?>
    <div class="Body">
      <center>
        <h2>Du er nu logget ind på systemet: <?php echo $_SESSION["fornavn"]; ?></h2>
        <h3>Nuværende status: <?php echo $status ?></h3>
        <div class="kontrolboks">
          <div class="info">
            <a id="right" class="change_user" href="rediger.php">Rediger bruger:

```

```
<i class="fa fa-cog"></i></a><br><br>
<a class="opretknap" href="alarm_log.php">Log over alarm:
  <i class="fa fa-folder-open"></i>
</a><br><br>
<a class="opretknap" href="Camera_log.php">Kamera:
  <i class="fa fa-video-camera" aria-hidden="true"></i>
</a>
<p><a id="right" class="logud" href="logud.php">Log ud <i class="fa fa-
sign-out"></i></a></p>
</div>
<form class="change_status" action="" method="post">
  <input name="tryk" type="hidden" value="submit">
  <div id="status"></div>
  <p id="status"></p><br>
  <h3>Andre status:</h3>
  <input id="ON" type="submit" name="submit" value="ON"><input id="OFF"
name="submit" type="submit" value="OFF"><br>
  <?php echo $print ?>
</form>
</div>
</center>
</div>
</body>
</html>
```

Bilag 13: rediger bruger.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link href="Style.css" rel="stylesheet">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-
    awesome.min.css">
    <title>Rediger brugere</title>
  </head>
  <body id="Background">
    <div class="Top">Kontrolcenter</div>
    <p><a href="login.php" class="back">Tilbage <i class="fa fa-sign-out"></i></a></p>
    <div class="Body">
      <center>
        <h2>Register bruger på alarmsystemet: </h2>
        <div class="kontrolboks">
          <a class="opretknap" href="opret.php">Opret bruger <i class="fa fa-pencil-
          square-o"></i></a>
          <form action="rediger.php" method="POST"><p>Fjern bruger(ID):</p><input
          class="fjernbruger" type="text" name="removeuser"><input type="submit" va-
          lue="Fjern bruger"></form>
        </div>
        <h2>Liste over registrerede brugere:</h2>
        <?php
          session_start();
          if($_SESSION["Login"] != "YES") {
            header("Location: index.php");
          }
          require 'databaseconnect.php'; //Php-koden kan kun køre, hvis databaseconnect-
          filen er inkluderet.
          $sql = "SELECT * FROM members"; //Lav en query hvor den gemmes alt fra tabellen
          members
          $result = mysqli_query($mysqlConnection, $sql); //Tilslutter til databasen og
          spørger efter ovenstående query
          echo '<table class="log">'; //Opret tabel.
          echo '<tr><th class="log"> ID: </th><th class="log"> Brugernavn:</th><th
          class="log"> Fornavn:</th><th class="log"> Efternavn: </th></tr>'; //Skriv ID-
          brugernavn.
          while ($row = mysqli_fetch_array($result)) { //Så længe der er flere brugere
          i tabellen fra databasen:
            echo "<center>";
            echo '<tr><td class="log1">';
            echo $row['id']; //Skriv id'et på brugeren i rækken.
            echo '</td><td class="log2">';
            echo $row['username']; //Skriv brugernavnet.
            echo '</td><td class="log3">';
            echo $row['fornavn']; //Skriv fornavnet
            echo '</td><td class="log4">';
            echo $row['efternavn']; //Skriv efternavnet
          }
          echo "</table>"; //Luk tabellen.
          if (isset($_POST['removeuser'])) { //Hvis der er indtastet noget i removeuser-
          feltet:
            $id = $_POST['removeuser']; //Gem variabelen fra removeuser-feltet som $id.
            $query = "SELECT * from members WHERE id= '{$id}'"; //Lav en forespørgsel
            på databasen, hvor den henter alt fra tabellen members, hvor id'et er
            lig med id'et fra removeuser.
            $result = mysqli_query($mysqlConnection, $query); //Lav forespørgslen.
            if (!$result->num_rows == 1) { //Returnerer antallet af rækker, og hvis
            id'et ikke findes på systemet:
              echo "<script type='text/javascript'>alert('ID'et findes ikke på sy-
              stemet');</script>";
            } else { //Ellers hvis den findes på databasen:
              $mysqlConnection->query("DELETE FROM members WHERE id = '{$id}'");
              //Fortæl databasen at den skal slette alt i kolonnen med id'et fra
              removeuser.
            }
          }
        }
      </center>
    </div>
  </body>
</html>
```

```
        echo "<script type='text/javascript'>alert('Brugeren er nu fjernet fra  
        systemet!');</script>";  
        header('Location:rediger.php'); //Redirect til rediger siden.  
    }  
    ?>  
</center>  
</div>  
</body>  
</html>
```


Bilag 14: opret bruger.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link href="Style.css" rel="stylesheet">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-
awesome.min.css">
    <title></title>
    <script>
      /*Tjek at de to indtastet password er ens*/
      function tjek(){
        var password_one = document.getElementById("pass").value;
        var password_two = document.getElementById("pass2").value;
        if(password_one != password_two){
          alert("De to adgangskoden matcher ikke");
          return false;
        } else {
          return true;
        }
      }
    </script>
  </head>
  <body id="Background">
    <?php
      session_start();
      if($_SESSION["Login"] != "YES"){
        header("Location: index.php");
      }
      require 'databaseconnect.php'; //Php-koden kan kun køre, hvis databaseconnect-filen
er inkluderet.
      $salt = strtr(base64_encode(mcrypt_create_iv(16, MCRYPT_DEV_URANDOM)), '+', '.');
      //Genererer en salt værdi, som skal lægges til passwordet der bliver hashet.
      $value = 10;
      $salt = sprintf("$2a$%02d$", $value) . $salt; //Definer oplysninger for hashen, så
      PHP kan bekræfte den senere. "$2a$" betyder, at vi bruger en algoritme der hedder
      blowfish.
      if (isset($_POST['user']) && isset($_POST['pass'])) { //Hvis der er indtastet i user
      og password felt:
        $user = $_POST['user']; //Gem variable med det indtastede brugernavn
        $pass = $_POST['pass']; //Gem variable med det indtastede password
        $fornavn = $_POST['fornavn']; //Gem variable med det indtastet fornavn
        $efternavn = $_POST['efternavn']; //Gem variable med det indtastet efternavn
        $hash = crypt($pass, $salt); //Lav hash af det indtastede password hvor den bruger
        salten fra tidligere igennem blowfish algoritmen.
        $query = mysqli_query($mysqlConnection, "SELECT * FROM members WHERE user-
        name='$user'"); //Lav en query på brugernavn, for at tjekke om den findes.
        if (mysqli_num_rows($query) > 0) { //Hvis brugeren allerede findes på databasen:
          echo "<script type='text/javascript'>alert('Brugeren findes allere-
          de');</script>";
        } else { //Hvis ikke:
          mysqli_query($mysqlConnection, "INSERT INTO members (username, pass-
          word,salt,fornavn,efternavn) VALUES ('$user','$hash','$salt','$fornavn','$ef-
          ternavn')"); //Lav en ny bruger på databasen, med værdierne der er indtastede,
          dog bruger den det hashede password.
          header("location:index.php"); //Redirecter til index siden.
        }
      }
      mysqli_close($mysqlConnection);
    ?>
    <div class="Top">Kontrolcenter</div>
    <div class="Body">
      <p><a href="login.php" class="back">Tilbage <i class="fa fa-sign-out"></i></a></p>
      <center>
        <h2>Register bruger på alarmsystemet: </h2>
        <form action="opret.php" method="POST" onsubmit="return tjek()"><p>Bruger-
        navn:</p><input type="text" name="user" required>
      </center>
    </div>
  </body>
</html>
```

```
<p>Fornavn:</p><input type="text" name="fornavn" required>
<p>Efternavn:</p><input type="text" name="efternavn" required>
<p>Adgangskode:</p><input id="pass" type="password" name="pass" required>
<p>Adgangskode igen:</p><input id="pass2" type="password" name="pass2" requi-
red><br />
<input type="submit" value="Opret!" />
</form>
</center>
</div>
</body>
</html>
```

Bilag 15: Camera_log.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link href="Style.css" rel="stylesheet">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-
awesome.min.css">
    <title>Camera Log</title>
  </head>
  <body>
    <div class="Top">Kontrolcenter</div>
    <p><a href="login.php" class="back">Tilbage <i class="fa fa-sign-out"></i></a></p>
    <center>
      <h3>Seneste billeder:</h3>
      <?php
        session_start();
        if($_SESSION["Login"] != "YES"){
          header("Location: index.php");
        }
        require 'databaseconnect.php';
        $query_pic = "SELECT * FROM pictures";
        $result_pic = mysqli_query($mysqlConnection, $query_pic); //Lav forespørgsel
        $i=0; //Variabel der holder styr på hvilket overlay der skal åbnes
        while($row = mysqli_fetch_array($result_pic)) { //Så længe der er noget i rækken:
          /*Sætter filstien til billedet ud fra filnavnet på database*/
          $source = "http://htx-elev.ucholstebro.dk/HX-15-c-el/glen0930/Doorbell_IMG/"
            . $row['fileName'];
          /*Tidsformatering*/
          date_default_timezone_set("Europe/Copenhagen");
          $tid = date("m/d/Y H:i:s", $row['time']);
          echo '<div style="display: inline-block;">'; //Lav div til billedet
          echo "||". $tid ; //Skriver tiden for det pågældende billede
          echo "<br>";
          /*Laver billede med link, så når man trykker på den kommer der et overlay*/
          echo '      <a href="#popup'. $i .' "></a>';
          echo "</div>";
          /*Laver overlay der er skjult, indeholder stor version af billede samt tiden*/
          echo '<div id="popup'. $i .' " class="overlay">
            <div class="popup">
              <a class="close" href="#">&times;</a>
              <div class="content">
                
                <p>Tidspunkt billedet er taget: '. $tid .'</p>
              </div>
            </div>
          </div>';
          $i++; //Lægger én til
        }
      ?>
    </center>
  </body>
</html>
```

Bilag 16: Update_log.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link href="Style.css" rel="stylesheet">
    <title></title>
  </head>
  <body id="Background">
    <?php
      if (isset($_GET['State'])) {
        require 'databaseconnect.php';
        date_default_timezone_set("Europe/Copenhagen");
        $tid = time();
        $State = $_GET['State']; //Gem værdien fra State
        if ($State >= 0) {
          if ($State == 1) { //Hvis dør er brudt:
            $Stateinsert = "Dør";
            mysqli_query($mysqlConnection,"INSERT INTO log (Location,Tid, status_id)
            VALUES ('$Stateinsert','$tid', 3)"); //Indsæt Lokation, tid og status
            mysqli_query($mysqlConnection,"UPDATE log_status SET log_status = '1'
            WHERE sted='Door'"); //Opdaterer log_status med 1 ved døren
          } else if ($State == 2) { //Ellers hvis vindue 1 er brudt:
            $Stateinsert = "Vindue 1";
            mysqli_query($mysqlConnection,"INSERT INTO log (Location,Tid,status_id)
            VALUES ('$Stateinsert','$tid',1)"); //Indsæt Lokation, tid og status
            mysqli_query($mysqlConnection,"UPDATE log_status SET log_status = '1'
            WHERE sted='Window_one'"); //Opdaterer log_status med 1 ved vindue 1
          } else if ($State ==3) { //Ellers hvis vindue 2 er brudt:
            $Stateinsert = "Vindue 2";
            mysqli_query($mysqlConnection,"INSERT INTO log (Location,Tid,status_id)
            VALUES ('$Stateinsert','$tid',2)"); //Indsæt Lokation, tid og status
            mysqli_query($mysqlConnection,"UPDATE log_status SET log_status = '1'
            WHERE sted='Window_two'"); //Opdaterer log_status med 1 ved vindue 2
          } else if ($State ==4) { //Ellers hvis sensor har registreret bevægelse:
            $Stateinsert = "sensor";
            mysqli_query($mysqlConnection,"INSERT INTO log (Location,Tid,status_id)
            VALUES ('$Stateinsert','$tid',4)"); //Indsæt Lokation, tid og status
            mysqli_query($mysqlConnection,"UPDATE log_status SET log_status = '1'
            WHERE sted='Sensor'"); //Opdaterer log_status med 1 ved Sensor.
          }
        }
        echo $State;
      }
    ?>
  </body>
</html>
```

Bilag 17: Alarm_log.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link href="Style.css" rel="stylesheet">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-
    awesome.min.css">
    <title>Status & log over alarmer</title>
  </head>
  <body id="Background">
    <?php
      require 'databaseconnect.php';
      session_start();
      if($_SESSION["Login"] != "YES"){
        header("Location: index.php");
      }

      /*Lampe;udenfarve*/
      $lightbulb_door = "pics/Lightbulb.png";
      $lightbulb_window_one = "pics/Lightbulb.png";
      $lightbulb_window_two = "pics/Lightbulb.png";
      $lightbulb_sensor = "pics/lightbulb.png";

      /*Forespørgsel til database*/
      $query_time = "SELECT * FROM alarm_status";//Definer hvad der skal hentes fra tabel
      alarm_status
      $result_time = mysqli_query($mysqlConnection, $query_time); //Lav forespørgslen.
      $row = mysqli_fetch_assoc($result_time); //Definer hver række
      $alarm_tid = $row['tidspunkt']; //Sæt tidsvariabel til det som står i rækken tidspunkt.
      $query_log = "SELECT * FROM log WHERE Tid >= $alarm_tid"; //Hent kun tiden fra alarmer
      er blevet brudt, såfremt tiden er efter tidspunktet for alarmer er blevet tændt.
      Således fås ikke værdier, der er før alarmer blev tændt.
      $result_log = mysqli_query($mysqlConnection,$query_log); //Lav forespørgslen.

      //Tidsformatering
      date_default_timezone_set("Europe/Copenhagen");//Sætter standart tidzone til Europa

      if ($row['status'] == 1){ //Hvis alarm er tændt:
        /*Lampe;grønfarve*/
        $lightbulb_door = "pics/Lightbulbgreen.png";
        $lightbulb_window_one = "pics/Lightbulbgreen.png";
        $lightbulb_window_two = "pics/Lightbulbgreen.png";
        $lightbulb_sensor = "pics/lightbulbgreen.png";

        while ($row_log = mysqli_fetch_assoc($result_log)) { //Så længe der er noget i
          rækken:
            if($row_log['status_id'] == 1){ //Hvis vindue1 er brudt
              $lightbulb_window_one = "pics/lightbulbred.png"; //Gør lampe rød
            } else if ($row_log['status_id'] == 2){ //Ellers hvis vindue2 er brudt
              $lightbulb_window_two = "pics/lightbulbred.png"; //Gør lampe rød
            } else if ($row_log['status_id'] == 3){ //Ellers hvis dør er brudt
              $lightbulb_door = "pics/lightbulbred.png"; //Gør lampe rød
            } else if ($row_log['status_id'] == 4){ //Ellers hvis sensor er brudt
              $lightbulb_sensor = "pics/lightbulbred.png"; //Gør lampe rød
            }
          }
        } else if($row['status'] == 0) { //Hvis alarmer er slukket:
          //Opdater log_status til 0 på alle lokationer.
          mysqli_query($mysqlConnection,"UPDATE log_status SET log_status = '0' WHERE
          sted='Door'");
          mysqli_query($mysqlConnection,"UPDATE log_status SET log_status = '0' WHERE
          sted='Window_one'");
          mysqli_query($mysqlConnection,"UPDATE log_status SET log_status = '0' WHERE
          sted='Window_two'");
          mysqli_query($mysqlConnection,"UPDATE log_status SET log_status = '0' WHERE
          sted='Sensor'");
        }
      }
    }
  </body>
</html>
```

```
}
?>
<div class="Top">Kontrolcenter</div>
<p><a href="login.php" class="opretknap">Tilbage <i class="fa fa-sign-out"></i></a></p>
<center>
  <h3>Log over de seneste brud på alarm:</h3>
  <div class="Body">
    <?php
      if(isset($_GET['log_state'])){ //Hent status på alarm
        $log_state = $_GET['log_state']; //Definer variable med status på alarmen
        switch ($log_state) { //Gør switch funktionen.
          case "1": //Tilfælde 1; dør
            $log_query = "SELECT Tid FROM log WHERE status_id = '3'";
            $result_door = mysqli_query($mysqlConnection, $log_query);
            /*Lav tabel med tiden for brud*/
            echo '<table class="log_time">';
            echo '<tr><th class="log"> Tidspunkt for brud for døren: </th></tr>';
            while ($row = mysqli_fetch_array($result_door)) { //Så længe der er
              flere hændelser med brudt dør:
                echo '<tr><td class="log1">';
                $tid = date("m/d/Y H:i:s",$row['Tid']); //Tidsformatering
                echo $tid; //Indsæt i tabel
            }
            echo "</table>";
            break;
          case "2": //Tilfælde 2; vindue 1
            $log_query = "SELECT Tid FROM log WHERE status_id = '1'";
            $result_door = mysqli_query($mysqlConnection, $log_query);
            echo '<table class="log_time_window">';
            /*Lav tabel med tiden for brud*/
            echo '<tr><th class="log"> Tidspunkt for brud for vindue 1:
            </th></tr>';
            while ($row = mysqli_fetch_array($result_door)) { //Så længe der er
              flere hændelser med brudt vindue 1:
                echo '<tr><td class="log1">';
                $tid = date("m/d/Y H:i:s",$row['Tid']); //Tidsformatering
                echo $tid; //Indsæt i tabel
            }
            echo "</table>";
            break;
          case "3": //Tilfælde 3; vindue 2
            $log_query = "SELECT Tid FROM log WHERE status_id = '2'";
            $result_door = mysqli_query($mysqlConnection, $log_query);
            echo '<table class="log_time_window">';
            /*Lav tabel med tiden for brud*/
            echo '<tr><th class="log"> Tidspunkt for brud for vindue 2:
            </th></tr>';
            while ($row = mysqli_fetch_array($result_door)) { //Så længe der er
              flere hændelser med brudt vindue 2:
                echo '<tr><td class="log1">';
                $tid = date("m/d/Y H:i:s",$row['Tid']); //Tidsformatering
                echo $tid; //Indsæt i tabel
            }
            echo "</table>";
            break;
          case "4": //Tilfælde 4; sensor
            $log_query = "SELECT Tid FROM log WHERE status_id = '4'";
            $result_door = mysqli_query($mysqlConnection, $log_query);
            echo '<table class="log_time_sensor">';
            /*Lav tabel med tiden for brud*/
            echo '<tr><th class="log"> Tidspunkt for brud for PIR-sensor:
            </th></tr>';
            while ($row = mysqli_fetch_array($result_door)) { //Så længe der er
              flere hændelser med sensor:
                echo '<tr><td class="log1">';
                $tid = date("m/d/Y H:i:s",$row['Tid']); //Tidsformatering
                echo $tid; //Indsæt i tabel
```



```
        }
        echo "</table>";
        break;
    default:
    }
}
?>
<!-- Indsæt plantegning med lamper -->

<a href="alarm_log.php?log_state=1"> </a>
<a href="alarm_log.php?log_state=2"> </a>
<a href="alarm_log.php?log_state=3"></a>
<a href="alarm_log.php?log_state=4"></a>
</div>
</center>
</body>
</html>
```

Bilag 18: Read_Alarm.php

```
<?php
    /*Bliv kaldt fra Arduinoen, når der ændres på tjek værdien og når Arduinoen skal tjekke om
    der er ændret status gennem hjemmesiden*/
    if (isset($_GET['tjek'])) {
        $mysqli = new mysqli("localhost", "15_c_el_glen0930", "ub4h3", "15_c_el_glen0930"); //Op-
        ret forbindelse
        $tjek = $_GET['tjek']; //Ændre værdien 'tjek'
        /*Tid*/
        date_default_timezone_set("Europe/Copenhagen");
        $tid = time();
        /*Opdater status og tidspunktet på serveren*/
        //Status:
        $stmt = $mysqli->prepare("UPDATE alarm_status SET status = ?");
        $stmt->bind_param("i", $tjek);
        $stmt->execute();
        //Tidspunkt:
        $stmt = $mysqli->prepare("UPDATE alarm_status SET tidspunkt = ?");
        $stmt->bind_param("i", $tid);
        $stmt->execute();
    }
    //Følgende skriver $ hvis status er 1 på siden, og € hvis status er 0.
    require 'databaseconnect.php';
    $query = "SELECT status FROM alarm_status";
    $result = mysqli_query($mysqliConnection, $query); //Lav forespørgslen.
    $row = mysqli_fetch_assoc($result);
    if ($row['status'] == 1){ //Hvis status er 1, fra database
        echo '$'; //print $ som arduinoen kan tolke
    }
    if ($row['status'] == 0){ //Hvis den er 0,
        echo '€'; //print € som arduinoen kan tolke
    }
}
```

Bilag 19: ChangeNameFTP.php

```
<?php
require 'databaseconnect.php';
$query_pic = "SELECT * FROM pictures"; //Gem alt fra tabellen pictures.
$result_pic = mysqli_query($mysqlConnection, $query_pic); //Hent alt fra tabellen pictures.
$antal = mysqli_num_rows($result_pic); //Gem antal rækker fra tabellen
$antal++; //Tæl en op
$new_file = "door" . $antal . ".png"; //Definer det nye navn for filen
$sold_file = 'Ring1.png'; //Definer det gamle navn for filen
$conn_id = ftp_connect('htx-elev.ucholstebro.dk'); //Tilslut til FTP-serveren
$login_result = ftp_login($conn_id, 'glen0930', 'seapencosea'); //Login på FTP
if (ftp_rename($conn_id, "/HX-15-c-el/glen0930/Public/Doorbell_IMG/" . $sold_file, "/HX-15-c-el/glen0930/Public/Doorbell_IMG/" . $new_file)){ //Hvis det lykkes at omdøbe filen:
    echo "Det gik jo fint";
    /*Tidsbehandling for at huske hvornår filen blev uploadet*/
    date_default_timezone_set("Europe/Copenhagen");
    $tid = time();
    $query_pic = "INSERT INTO pictures (fileName, time) VALUES ('$new_file', $tid)";
    $result_pic = mysqli_query($mysqlConnection, $query_pic);
}
ftp_close($conn_id);
?>
```

Bilag 20: logud.php

```
<?php
    session_start(); // Begynder session
    session_unset(); // Uddeklarer alle sessioner
    session_destroy(); // Sletter session
    header( "refresh:2;url=index.php" ); //Redirect til forside
?>
<html>
    <head>
        <meta charset="UTF-8">
        <link href="Style.css" rel="stylesheet">
        <title>Logget ud</title>
    </head>
    <body id="Background">
        <center>
            <h1>Du er nu logget ud</h1>
        </center>
    </body>
</html>
```

Bilag 21: Godkendte projektbeskrivelse



HTX, 3.a
Teknikfag - EL
Eksamensprojekt

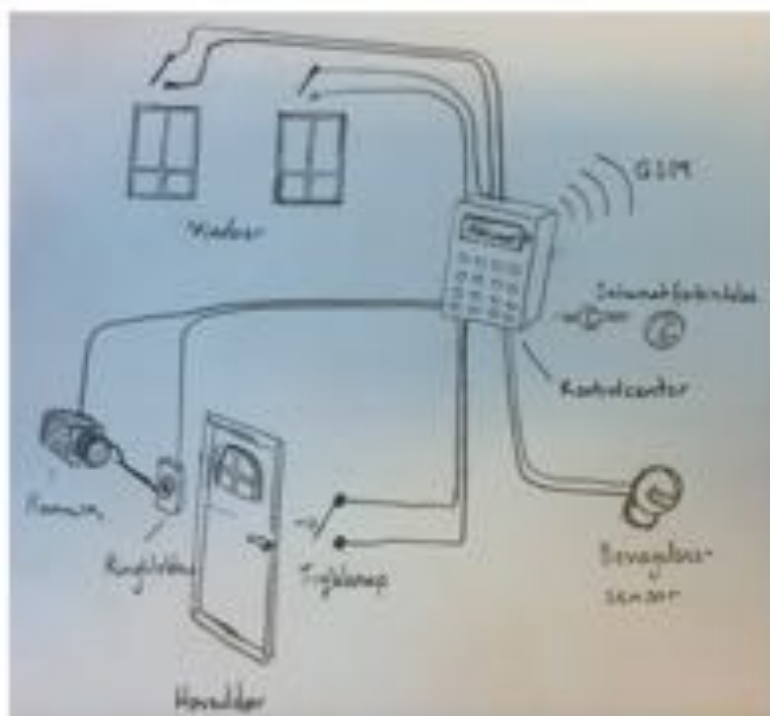
Tobias Sørensen,
Christian Henriksen
& Glenn Herping

PROJEKTBEKRIVELSE - ALARMSYSTEM

Generelt

I forbindelse med eksamensprojektet har vi valgt temaet "velfærds-elektronik i hjemmet", der omfatter udvikling og fremstilling af et stykke velfærds-elektronik. Den skal altså kunne løse en håndgribelig opgave og evt. med henblik på en bestemt målgruppe. Det kan f.eks. være et robot-støvsugere, SONOS, styring af lyd og lys osv. I dette eksamensprojekt har derfor valgt at arbejde med fremstillingen af et intelligent alarmsystem, der ved hjælp af elektriske kredsløb, sensorer, programmer og et integreret adgangssystem kan sikre det normale hjem. Alarmsystemet skal være brugervenligt og anvendeligt fra apparater med internetadgang, således at det kan styres fra et webbaseret kontrolcenter. Med andre ord skal der laves en hjemmeside med bruger-interface, hvorfra styring af systemet er muligt, såfremt brugeren opfylder de adgangsgivende krav. På den måde kan man holde sig opdateret på huset, når man ikke er hjemme og eksempelvis på arbejde eller ferie. Her er det også meningen, at alarmsystemets "intelligens" skal kunne give besked og opdatere om eventuelle ændringer og besøgende, såvel som at den eventuelt skal kunne simulere normal beboelsesadfærd.

For demonstrationens skyld vil systemet opstilles således, at det beskytter 1 hoveddør, 2 vinduer og et område med en bevægelsessensor. Dørene og vinduerne vil aktiveres vha. trykknapper, men der vil også kobles en ekstra tryknap til som dørklokke. Meningen med dørklokken er, at der ved uventet besøg, når alarmsystemet er tændt, vil være mulighed for at se hvem der ringer på. Dette sker vha. et skjult kamera ved døren, som aktiveres, når dørklokken bruges af uvedkommende. Der vil altså blive sendt et tidsstempelt billede til serveren og en besked til brugeren, når alarmen er aktiveret, så man kan logge ind på hjemmesiden og se, hvem der ringer på døren. Desuden skal der opstilles en sensor, som kan registrere bevægelser, når alarmen er aktiveret. Her vil der evt. blive udviklet på, hvornår sensoren skal slå alarm, alt efter hvor meget aktivitet den registrerer.



Hardwaremæssigt vil kontrolcenteret være samlet omkring én Arduino, som ved siden af diverse trykknapper og sensorer, skal arbejde sammen med et GSM-modul, således at brugeren får direkte feedback om evt. aktivitet. Arduinoen vil ligeledes være koblet til internettet, så der kan sendes feedback til hjemmesiden. Derudover skal man kunne betjene centeret direkte vha. et matrix-tastatur og et LCD-display, hvor en bruger med adgang kan slå alarmen til eller fra.

Tværfaglighed

I projektet vil der indgå forskellige elementer fra andre fag, som matematik, programmering og kommunikation og IT. Fra matematik vil der indgå statistik omkring hvor lang tid, eller hvor stor sandsynligheden er, for at man kan få en maskine til at knække en kode og dermed komme ind i alarmsystemet via kontrolpanelet på hjemmesiden.

Desuden vil der indgå overvejelser om, hvor brugervenligt systemet er og dermed hvor sikkert det er overfor fejl fra brugerens side, dvs. inden for en målgruppe.

Slutteligt er der også inddraget programmering, med hensyn til database, som ikke er en del af teknikfaget el.

Blokdiagrammer

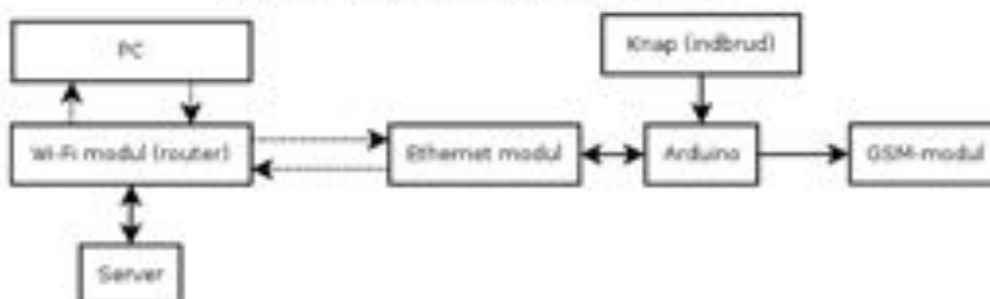
Herunder er skitseret de forskellige funktions- og blokdiagrammer for de tre detekteringstyper, som er hhv. døre og vinduer, ringklokke og bevægelsessensor.

Funktionsdiagram - Sikring af døre og vinduer



Sikringen af døre og vinduer sker i Arduinoens software. Er alarmer aktivert, holdes der øje med om "låsen", repræsenteret af en tryknap, er brudt. Hvis den brydes vil alarmeringen ske og serveren vil få besked samtidigt med at der udsendes en SMS til både bruger og alarmcentral.

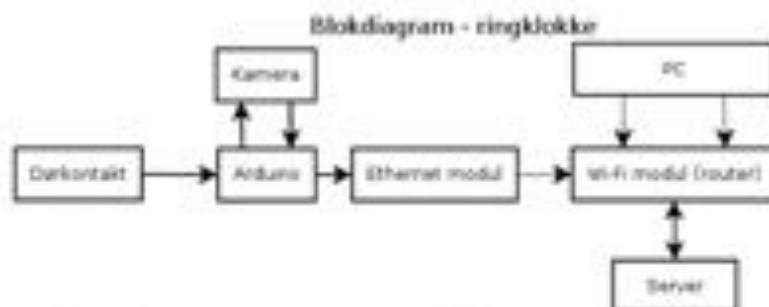
Blokdiagram - Sikring af døre og vinduer



Ringklokken vil være forbundet med Arduinoen, så der ved brug af denne, vil sætte gang i softwaren, som tager et billede og sender til serveren med tidsangivelse. Brugeren vil herefter få mulighed for at se billedet på hjemmesiden.

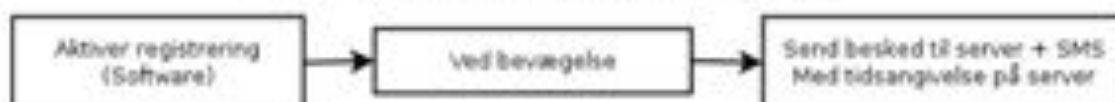
Funktionsdiagram - ringklokke



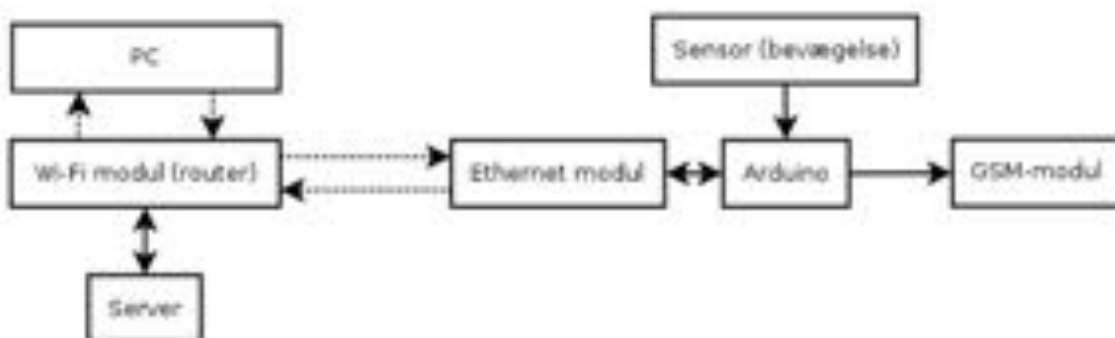


Sensoren virker efter samme princip som trykknapperne ved døren og vinduerne. Her er det blot en bevægelsesfølsom sensor, der aktivere alarmen, som så sender SMS og data til serveren.

Funktionsdiagram - Registrer bevægelse



Blokdiagram - Registrer bevægelse



Kravspecifikation

For at have overblik over projektet, samt være sikker på hvilke dele af tyverialarm der skal udføres er forskellige krav blevet opstillet. Disse krav er som følgende:

- Kontrolcenter, være tilgængeligt hvor end man er, så længe der er internet. Herunder selve kontrolpanelet, hvor Arduinoen sidder og styrer resten.



HTX, 3.a
Teknikfag - EL
Eksamensprojekt

Tobias Sørensen,
Christian Henriksen
& Glenn Herping

- Login på hjemmesiden påkrævet for at styre alarmer og se en log over tidligere aktiveret alarmer.
- Ved oprettelse af en ny bruger skal en two-steps-verification bruges. Eventuelt hvor man opretter brugeren og derefter skal man indtaste en unik kode (tilfældig kode der generes hver gang en bruger oprettes) på selve alarmer. For at oprette en ny bruger skal man først være logget ind.
- Man skal have mulighed for at tænde og slukke for alarmer på kontrolpanelet vha. en kode man har fra kontrolcenteret.
- Der skal være et display, hvor man kan se diverse statusser.
- Der skal eventuelt være en batteribackup på kontrolpanelet.
- Ringklokke
 - Når der trykkes på ringklokken skal der tages et billede af personen, der står udenfor.
 - Billedet uploades til kontrolcenteret og der tilføjes et tidsstempel.
 - Skal virke uanset om alarmer er aktiveret eller ej.
- Sikring af døre og vinduer
 - En kontakt der bliver aktiveret hvis en dør/vindue åbnes mens alarmer er tændt. Såfremt alarmer ikke er tændt skal der selvfølgelig ikke ske noget.
 - Upload en log med tidspunktet og hvilken dør/vindue der blev aktiveret til kontrolcenteret, hvis det sker når alarmer er aktiveret.
 - Send en SMS til administratoren om det mulige indbrud.
- Sensor
 - En sensor skal, når alarmer er aktiveret, registrere bevægelser.
 - Upload en log med tidspunktet til kontrolcenteret.
 - Send en SMS til administratoren om det mulige indbrud.

Tidsplan

På næstfølgende side ses et overblik over tidsplanen for projektet. Projektet bør følge tidsplanen, men det er muligt, at der vil forekomme afvigelser, da der er områder i projektet, som vi ikke før har beskæftiget os med.

Hvis en del af tidsplanen kræver at det bliver kodet er det en del af det element.

